Handbook of Statistics

Series Editor

C.R. Rao
C.R. Rao AIMSCS, University of Hyderabad Campus,
Hyderabad, India

North-Holland is an imprint of Elsevier
Radarweg 29, PO Box 211, 1000 AE Amsterdam, Netherlands
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1 GB, UK

Notices
Knowledge and best practice in this field are constantly changing. As new research and experience broaden our understanding, changes in research methods, professional practices, or medical treatment may become necessary.

Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information, methods, compounds, or experiments described herein. In using such information or methods they should be mindful of their own safety and the safety of others, including parties for whom they have a professional responsibility.

To the fullest extent of the law, neither the Publisher nor the authors, contributors, or editors, assume any liability for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions, or ideas contained in the material herein.

ISBN: 978-0-444-63492-4
ISSN: 0169-7161

Contributors

Numbers in Parentheses indicate the pages on which the author's contributions begin.

Ricardo Baeza-Yates (303), Yahoo Labs, Sunnyvale, California, USA

Gino Biondini (29), Department of Mathematics, State University of New York at Buffalo, Buffalo, New York, USA

Simon Chan (107), Department of Computer Science, University College London, London, United Kingdom

Varun Chandola (239), Computer Science and Engineering, State University of New York at Buffalo, Buffalo, New York, USA

Nemanja Djuric (145), Yahoo Labs, Sunnyvale, California, USA

John Feo (339), Context Relevant, Seattle, Washington, USA

Michal Galas (259), Department of Computer Science, University College London, London, United Kingdom

Auroop Ganguly (239), Civil and Environmental Engineering, Northeastern University, Boston, Massachusetts, USA

Vito Giovanni Castellana (339), Pacific Northwest National Laboratory, Richland, Washington, USA

Venu Govindaraju (3, 69), Department of Computer Science and Engineering, University at Buffalo, State University of New York, Buffalo, New York, USA

Mihajlo Grbovic (145), Yahoo Labs, Sunnyvale, California, USA

Venkat N. Gudivada (203), East Carolina University, Greenville, North Carolina, USA

David Haglin (339), Pacific Northwest National Laboratory, Richland, Washington, USA

Devashish Kumar (239), Civil and Environmental Engineering, Northeastern University, Boston, Massachusetts, USA

Jungsuk Kwac (89), Stanford Sustainable Systems Lab, Stanford University, Stanford, California, USA

Mounia Lalmas (303), Yahoo Labs, London, United Kingdom

Janette Lehmann (303), Universitat Pompeu Fabra, Barcelona, Spain

Madhav V. Marathe (171), Department of Computer Science, Virginia Tech, Blacksburg, Virginia; and Network Dynamics and Simulation Science Laboratory, Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, Virginia, USA

Diana Moise (279), Cray Inc., Switzerland, and This work was carried out as part of the Post-Doctoral Researcher position at INRIA Rennes, France

Alessandro Morari (339), Pacific Northwest National Laboratory, Richland, Washington, USA

Ifeoma Nwogu (3, 69), Department of Computer Science and Engineering, University at Buffalo, State University of New York, Buffalo, New York, USA

Neeti Pokhriyal (69), Department of Computer Science and Engineering, University at Buffalo, State University of New York, NY-14221

Saumyadipta Pyne (171), Bioinformatics, CR Rao Advanced Institute of Mathematics, Statistics and Computer Science, University of Hyderabad Campus, Hyderabad, and Public Health Foundation of India, New Delhi, India

Vijay V. Raghavan (203), University of Louisiana at Lafayette, Louisiana, USA

Ram Rajagopal (89), Stanford Sustainable Systems Lab, Stanford University, Stanford, California, USA

Ranga Raju Vatsavai (239), Computer Science, North Carolina State University, Raleigh, North Carolina, USA

Dhana Rao (203), Marshall University, Huntington, West Virginia, USA

Srirangaraj Setlur (3), University at Buffalo, SUNY, Buffalo, New York, USA

Denis Shestakov (279), Bright Computing, Netherlands, and This work was carried out as part of the Post-Doctoral Researcher position at INRIA Rennes, France

Maryam Shoaran (125), Department of Mechatronics, School of Engineering Emerging Technologies, University of Tabriz, Tabriz, Iran

Alex Thomo (125), Department of Computer Science, University of Victoria, Victoria, Canada

Philip Treleaven (107), Department of Computer Science, University College London, London, United Kingdom

Antonino Tumeo (339), Pacific Northwest National Laboratory, Richland, Washington, USA

Oreste Villa (339), NVIDIA Research, Santa Clara, California, USA

Slobodan Vucetic (145), Temple University, Philadelphia, Pennsylvania, USA

Anile Kumar S. Vullikanti (171), Computer Science and Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, Virginia, USA

Jesse Weaver (339), Pacific Northwest National Laboratory, Richland, Washington, USA

Jens H. Weber (125), Department of Computer Science, University of Victoria, Victoria, Canada

Preface

While the term "Big Data" is open to varying interpretation, it is quite clear that the Volume, Velocity, and Variety (3Vs) of data have impacted every aspect of computational science and its applications. The volume of data is increasing at a phenomenal rate and a majority of it is unstructured. With big data, the volume is so large that processing it using traditional database and software techniques is difficult, if not impossible. The drivers are the ubiquitous sensors, devices, social networks and the all-pervasive web. Scientists are increasingly looking to derive insights from the massive quantity of data to create new knowledge. In common usage, Big Data has come to refer simply to the use of predictive analytics or other certain advanced methods to extract value from data, without any required magnitude thereon. Challenges include analysis, capture, curation, search, sharing, storage, transfer, visualization, and information privacy. While there are challenges, there are also huge opportunities emerging in the fields of Machine Learning, Data Mining, Statistics, Human-Computer Interfaces and Distributed Systems to address ways to analyze and reason with this data. Among other things, Big Data has the potential to help companies improve operations and make faster, more intelligent decisions. This edited volume focuses on the challenges and opportunities posed by Big Data in a variety of domains and how statistical techniques and innovative algorithms can help glean insights and accelerate knowledge discovery.

Venu Govindaraju
Vijay V. Raghavan
C.R. Rao

Chapter 1

Document Informatics for Scientific Learning and Accelerated Discovery

Venu Govindaraju[*,1], **Ifeoma Nwogu**[*,1], **Srirangaraj Setlur**[*,1]
[*] *University at Buffalo, SUNY, Buffalo, New York, USA*
[1] *Corresponding authors: e-mail: govind@buffalo.edu; inwogu@buffalo.edu; setlur@buffalo.edu*

ABSTRACT
This chapter presents a concept paper that describes methods to accelerate new materials discovery and optimization, by enabling faster recognition and use of important theoretical, computational, and experimental information aggregated from peer-reviewed and published materials-related scientific documents online. To obtain insights for the discovery of new materials and to study about existing materials, research and development scientists and engineers rely heavily on an ever-growing number of materials research publications, mostly available online, and that date back many decades. So, the major thrust of this concept paper is the use of technology to (i) extract "deep" meaning from a large corpus of relevant materials science documents; (ii) navigate, cluster, and present documents in a meaningful way; and (iii) evaluate and revise the materials-related query responses until the researchers are guided to their information destination. While the proposed methodology targets the interdisciplinary field of materials research, the tools to be developed can be generalized to enhance scientific discoveries and learning across a broad swathe of disciplines. The research will advance the machine-learning area of developing hierarchical, dynamic topic models to investigate trends in materials discovery over user-specified time periods. Also, the field of image-based document analysis will benefit tremendously from machine learning tools such as the use of deep belief networks for classification and text separation from document images. Developing an interactive visualization tool that can display modeling results from a large materials network perspective as well as a time-based perspective is an advancement in visualization studies.

Keywords: Accelerated discovery, Digital document analysis, Probabilistic topic models, Scientific learning, Visualization

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00001-0

1 INTRODUCTION

In June 2011, the White House announced the Materials Genome Initiative (MGI), as a critical effort to enhance America's global competitiveness, by bolstering the U.S. advanced manufacturing enterprise (White House Materials Genome Initiative (MGI), 2011). MGI was launched as a presidential initiative to aid businesses discover, develop, and deploy new materials twice as fast. "The invention of silicon circuits and lithium ion batteries made computers and iPods and iPads possible, but it took years to get those technologies from the drawing board to the market place," said the President as he announced the Initiative. "We can do it faster." Accelerating the pace of discovery and deployment of advanced material systems will be crucial to achieving global competitiveness in the twenty-first century. Figure 1 shows phases of how materials move through the development continuum.

Most new materials innovation has not advanced much beyond Thomas Edison's trial-and-error approach to creating the light bulb in the late 1800s—it takes years, sometime decades to develop a new material. There are potentially high costs involved in development, high failure rate, and strong dependency on experts to manufacture and deploy the "successful" materials. For example, according to research published by the Tufts Center for the Study of Drug Development (DiMasi et al., 2003), the average cost of developing a new biotechnology material or drug can be as high as $1.2 billion[1] over a period

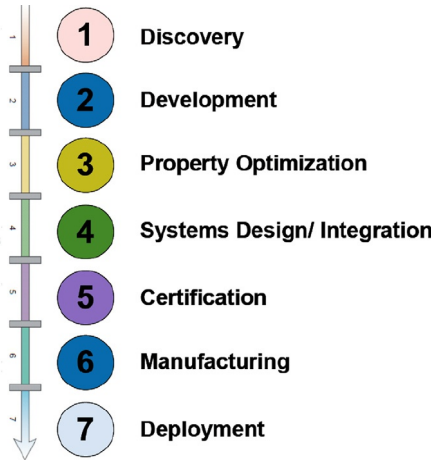


FIGURE 1 The materials innovation development continuum. *Source: White House Materials Genome Initiative (MGI), 2011.*

1. The biggest contributing factor is failure. The Tufts estimate is for the costs racked up by one product making it through. Other researchers have argued against this value, but their arguments are not substantiated in the literature.

of about 12–15 years, having a fallout rate of 90+%. Similarly, the lithium ion battery took approximately 20 years from discovery to deployment. Because this pathway from discovery to commercialization can take decades, one of the main thrusts of MGI is to reduce development time by providing the infrastructure and training that American innovators need in order to discover, develop, manufacture, and deploy advanced materials in a more expeditious and economical way (White House Materials Genome Initiative (MGI), 2011).

The University at Buffalo (UB) research team, led by Venu Govindaraju, has mobilized to work as part of a broader effort to realize the President's vision for advanced materials, by focusing on accelerating the early R&D stages of the materials continuum. This will be accomplished by giving researchers access to a very large data set as well as facilitation tools, upon which to base their models, and to provide a more complete picture of a material's characteristics. One of the factors limiting a scientist's ability to model materials behavior and invent new materials is their limited knowledge of the underlying physical and chemical mechanisms of the material system, and also limited knowledge of what has already been done by other researchers.

The work being undertaken by the UB researchers intends to accelerate materials discovery by significantly speeding-up the upfront learning processes undertaken by researchers during early development phases of new materials. This research involves a *meta-learning* approach where the large volumes of information obtained from peer-reviewed scientific articles are intelligently aggregated, clustered, and presented back to a researcher in a timely fashion, thereby facilitating the opportunity to leverage available information for modeling and simulation (from the large corpus of articles). Experimentalists will benefit from such summaries and deep data analytics for their own materials discovery research.

1.1 Sample Use Case

1.1.1 Description

An experimental materials physicist is interested in discovering new materials that have a temperature-driven metal–insulator transition (MIT). There are only a few known materials that act like an insulator at low temperatures but like a metal at warmer temperatures, but these MIT sources have not been studied extensively and even fewer experimental works exist in this area. The researcher must consider that titanium, vanadium, chromium, cobalt, iron, nickel, and ruthenium all exhibit rich phenomena of anomalous metallicity related to MIT. She therefore wants to know if in the last 20 years, any oxides of these metals have been measured under the conditions of. . .a certain temperature range, or under a certain magnetic field, or under a certain electrical field, or measured with a certain spectroscopy? At a global scale, she would find it extremely useful not only to find out what metals or their oxides exhibit the MIT phenomenon, but also see the trend of how materials with this property have been discovered

and tested over the years. And as new measurement tools are been developed, what new tests are being performed to expose this phenomenon? What new measurements are being taken and what is the range of recorded data? Who are the specialists/experts in this area? What journals carry the most informative experimental articles about this topic?

Unfortunately, the data set with which she must interact is extremely large and complex, comprising published scientific journal articles over a 20 year span. Also, the problem of "studying about the conditions under which certain metals exhibit MIT properties" is ill-defined and broadly scoped, especially because many researchers do not even necessarily refer to this phenomenon in their articles as MIT.

1.1.2 Current Research Process

The following steps provide an overview of the typical current process that the researcher engages in as she begins her quest:

1. She starts out entering a few related keywords such as the metal names into "Google" or Google Scholar, to get high-level information from *Wikipedia*® and similar online sources.
2. She begins to build her intuition and perform trial-and-error-based searches to get some basic information about MIT-exhibiting materials.
3. Now, armed with some basic information about her task, she begins to perform more specific journal searches over authors, titles, and abstracts. The focused journal search process:
   - She searches for articles using the metal names and interesting phenomena she believes might be related to MIT, *orbital occupations*, *lattice changes*, etc.
   - Iteratively, she constantly revises the specificity of her search (too specific returns too few articles and too general returns too many).
   - She cursorily reviews the returned lists, checking whether the articles are experimental as she is only interested in these for now. She then saves "potentially useful articles" to a private collection (this collection cannot be too large as it will quickly become unwieldy to manage).
   - At a more convenient time, she *serially* scans through all the articles in the collection, reviewing their properties data and deciding whether to discard or keep for more in-depth study at a later time.
   - She is constantly taking notes and maintaining a type of logbook as she gleans more knowledge about these phenomena and the related materials, from the articles.
4. Now, she has narrowed her search to only three metals but will repeat a more focused version of the entire process, as the information obtained so far is still not sufficient begin to design her own series of experiments for a new discovery.

1.1.3 Problems with the Current Process

Although search engines like Google Scholar are useful for keyword-based searches, their articles are typically indexed online by their meta-data such as *authors, title, abstract, keywords, year, journal/book-title*, and other bibliographic information (Davis and Lagoze, 1995), and paper prominence is driven solely by its citations. Beyond keywords in the title, abstract, descriptor, and/or fulltext, the researcher still has to include the author names, journal title, and/or publication year in the specifying query (Jacso, 2010), in order to retrieve relevant articles. These online searches are especially inappropriate for bibliometric searches, for evaluating publishing performances or the impact of researchers and/or journals. For this reason, the intuition-building exercise which needs to happen early in the research process can quickly lead down dead-end trails. Performing trial-and-error searches online can be laborious and time consuming.

Currently, the process of scanning through the contents of an article to figure out if it contains sufficiently relevant material is a completely manual activity performed serially. It is therefore easily prone to many human errors including hit-or-miss errors, and is again also very laborious and time consuming. Currently, it is not possible to search for a journal article based on a desired range of a key property of one or more materials-of-interest. Lastly, there is no automated way to figure out in advance if an article is experimental, theoretical, or computational without scanning at least the first page or two of the article to check for (i) equations, analytical models of materials and materials behavior (thus a theoretical paper); (ii) computing properties of materials, numerical solutions of the predictive equations of the models (thus a computational paper); or (iii) descriptions of equipment or measurement and characterization processes (thus an experimental paper). The matter is complicated because some majority experimental papers contain some theory, and vice versa for theory papers.

1.1.4 The Future Process

The researcher interacts with the our engine via an interactive questions/answering session at the front-end. The research engine validates the formulated queries via natural language, at the start of the process, to ensure the system and user are both referring to the same meaning. An extensive documents-driven, indexing system is prebuilt at the back-end from tables, figures, range of data, etc. Data in the engine is organized primarily by materials, material-related phenomena, and by key properties of the materials. There is also a time component to all the data. Based on the nature of the query, the system can either access the prebuilt hierarchical statistical model or dynamically develop a statistical model for the specific request. Results can be summarized in a Wikipedia-like format and presented to the user. The results are also presented to the user via the interactive visualization tool to browse over time, over related materials, to launch original journals, etc. To perform direct query searches using material properties and data ranges, results can be displayed as tables, figures, and plots from the articles.

Although we are not currently attempting to solve the end-to-end process described here, we are developing many of the core pieces to realize this vision, including (i) the hierarchical, time-based, clustering statistical model; (ii) the ability to search on texts in plots and tables and display the query results in form of the original figures; and (iii) the interactive visualization tool.

1.1.5 Benefits of the Future Process

Trial-and-error searches are reduced significantly via the interactive query session upfront. The process of manually scanning through journal contents in a serial fashion is now extensively automated, reducing human errors including hit-or-miss errors, while also saving time and labor spent opening and scanning through journal contents. It will now be possible to search for articles based on a desired range of key properties and the results can be presented in different forms as indicated by the user. Different query result views (with the option to select one) are presented to the user. Lastly, one of the most important aspects of the tool is its ability to display materials in a network graphical structure, showing proximity of materials and links based on material phenomena. These can also be viewed over time so that the most recently discovered materials and/or material properties are displayed. For example, a recently discovered material, exhibiting MIT properties, associated with vanadium (vanadium dioxide VO_2) can be shown on a timeline along with other similar vanadium-linked materials. It is now possible to know in advance if an article is experimental, theoretical, or computational, and in addition what materials science phenomena relate this article to another (potential link in a network graph). Deep searches into table contents, figure legends, etc. can now be performed and done quickly.

It would be of significant benefit to researchers to extract x–y pairs from the plots in the figures analytically/automatically. Currently, it is done very laboriously with a special tool, where one lays the journal page down, runs some software, defines the corners of the figure, and then uses the mouse to one-by-one click on each data point. It is extremely slow and also liable to human error, if one does not put the mouse right on top of the data point. It would be of benefit to many scientists to have a visualization tool with an image analysis component which could extract the data in numerical format so that the data can be examined for range, maximum values, etc. In summary, in addition to its other benefits, the development of this system including its front-end tools would greatly enhance the efficiency of material researchers focused on new materials discovery.

2 HOW DOCUMENT INFORMATICS WILL AID MATERIALS DISCOVERY

For generations, published scientific documents have been the primary sources for advanced scientific discovery and knowledge, yet the process by which scientists and engineers interact with these documents has largely remained

the same. And although recent years have seen rapid growth in parallel and distributed computing systems, developed in large part to serve as the backbone of the modern Internet-based information ecosystem (Frontiers in Massive Data Analysis—NIST Big Data Working Group, n.d.), the scientific learning community has not taken sufficient advantage of this growth in systems architecture to accelerate scientific discoveries and knowledge acquisition.

Specifically, the tools, practices, and architectural framework we propose in this research work will provide the means to structure and visualize large collections of documents and concepts, extract "deep knowledge" from scholarly scientific documents online, and accelerate the dispersion of new knowledge, thus encouraging science-based innovation. Our proposed approach aims to enhance the traditional methods (either computational, experimental, or theoretical) of discovering new materials or new materials-related phenomena, thereby advancing the fundamental understanding of materials more generally.

In a world of ever-increasing publications, there is no dispute over the fact that many articles are not getting proper attention (Brooks, 2009; Haque and Ginsparg, 2009; King, 2006); for example, the average number of citations per article in mathematics is below 1 (Adler et al., 2012). By offering a wider variety of ways to interact with articles, we anticipate significant improvements in the availability of research methods and results for scholars and for applications in society.

2.1 Motivation

> *Advanced materials are essential to economic security and human well-being, with applications in multiple industries, including those aimed at addressing challenges in clean energy, national security, and human welfare. Accelerating the pace of discovery and deployment of advanced material systems will therefore be crucial to achieving global competitiveness in the 21st century.*
>
> From the MGI Vision statement (White House Materials Genome Initiative (MGI), 2011)

It is apparent that we are at the crossroads of scientific development where innovative approaches for integrated comprehension of digitally available resources are vital. This is possible only by coordinated cross-cutting research that crosses the traditional boundaries of academic disciplines to develop effective integrated solutions for learning and discovery. Thus, to keep up with the pace of scientific progress and help mankind address our current problems and the new challenges that keep confronting us, it is necessary for the scientific community to have efficient means to assess the current state of the art, thorough research to develop good scientific intuition as to what is likely to work and which course to pursue, and in a way that will lead to accelerated discoveries with the use of meta-machine learning methods. Development of such technologies is also likely to have significant broader societal impact.

And specifically, for materials science, the proposed infrastructure work will accelerate new materials discovery (of significant benefit to society), by enabling faster recognition and use of important theoretical, computational, and experimental information aggregated from peer-reviewed and published materials-related scientific documents online.

Scientifically, given the rate at which our digital knowledge bases are growing, it will take an immense number of and an amalgam of diverse technologies to prepare our upcoming generations of students and researchers to navigate the path of scientific documents for new discovery purposes. The automated techniques available today for scientific research are very rudimentary, involving technologies such as keyword- and parser-based searches. These are not adequate at all to enable a satisfactory experience from the perspective of a researcher who is looking to quickly identify landmark papers, and focus on deeper aspects of the paper such as the analysis of flow charts, graphs, diagrams, tables, and scientific notation such as mathematical formulae, chemical symbols, and other structures. While machine-print text processing is a mature field, interpretation of elements such as graphs, figures, and tables still presents a lot of unsolved challenges in Document Image Analysis (DIA). Reasoning with and understanding the output of the DIA and representing and indexing this knowledge such that it is amenable to more nuanced search will entail new research in Ontologies, Natural Language Processing, and Information Retrieval (IR).

Technically, the research will advance the machine-learning subarea of developing hierarchical, dynamic topic models to investigate the trends in materials discovery over user-specified time periods. Also, the field of image-based document analysis will benefit tremendously from machine learning tools such as the use of deep belief networks (DBNs) for classification and text separation from document images. Developing an interactive visualization tool that can display modeling results from a large materials network perspective and a time-based perspective is an advancement in visualization. Lastly, high-performance computing techniques will be implemented in the visualization module to allow for any scalability issues that might result from interacting with the large size of the documents corpus. Unlike traditional methods of text-based document clustering, our approach will index document data over text, images, and number units, where the data for modeling would be obtained from the standard text in the article, table contents, image captions, and data plots.

2.2 Big Data Justification

How many scholarly research articles are there in existence? According to Jinha (2010), scholarly journal articles first appeared in 1665, and the cumulative total is estimated at over 50 million as of 2009, resulting in well over a billion pages of scholarship data. PubMed, a freely available index of biomedical

abstracts published by the National Center for Biotechnology Information, has a collection of more than 19 million citations (Dogan et al., 2009). But even this enormous database excludes large swathes of previously published articles in physics, mathematics, chemistry, engineering, and computer science not indexed by the United States National Library of Medicine. Scopus, a subscription-only database of journals, covers a wider range of literature than PubMed and currently claims to have indexed over 40 million records (Burnham, 2006). A rival of Scopus, the ISI Web of Knowledge (WOK), claims to be a similar size with 40 million items and about a third of these are scientific articles. Google Scholar also indexes all this data, but the size of their index is not publicly available (Jacso, 2010). PubMed, Scopus, WOK, and Google Scholar typically only index peer-reviewed articles. According to Larsen and von Ins (2010), the rate of growth of *scientific* journals is about 4.7–7% per year exponential growth, where the number doubles over a 10-year period. These vast collections of publications offer an excellent opportunity for the automatic discovery of knowledge by discovering unapparent logical connections among existing scholarly information (Jin et al., 2007; Konrad, 2007).

The goal of this research work, therefore, is to devise a method to aid materials researchers in *learning about how to learn* from a growing number of large-scale materials research documents online, hence the term *meta-learning*. The major thrust of this research work involves using technology to aid researchers in (i) extracting "deep" meaning from a large number of related documents; (ii) navigating, clustering, and summarizing response documents in a meaningful way; and (iii) evaluating and revising the materials-related query responses until the researchers are guided to their information destination. Researchers would not only have accomplished their specific research tasks with the aid of these tools but would have also learned about "how to learn" about new and previously existing materials or new materials-related phenomena from the large collection of scientific articles online. Although the methodology we propose in this project is geared toward the interdisciplinary field of materials research, these methods can be generalized to enhance the process of general scientific discoveries and science learning online. Figure 2 shows the rate of data growth.

2.3 Challenges of Meta-Learning in Materials Research

The large amounts of data necessary to represent digital text, along with the subjectivity of classification, make the problem of learning from textual data extremely challenging. Intelligent text classification methods, which rely heavily on machine learning algorithms, have the potential to supersede existing IR techniques and provide superior facilities that will save time and money for users and companies.

The interdisciplinary nature of the field of materials research presents a unique challenge in meta-learning because it is closely related to many areas of

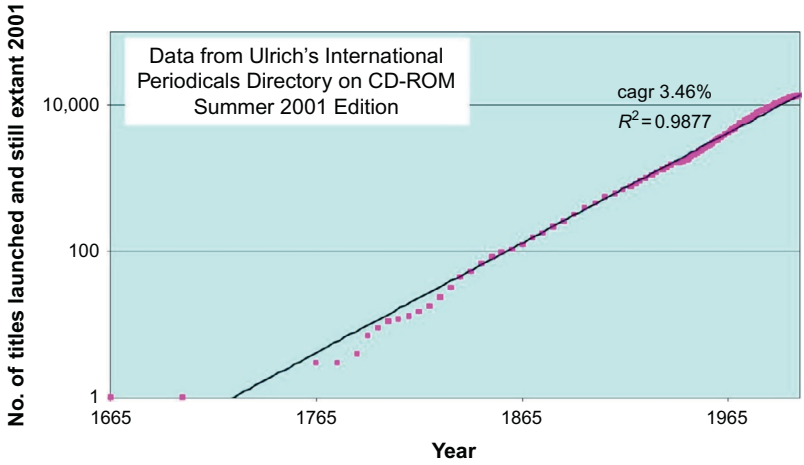


FIGURE 2 Data growth rate estimated from Ulrich's International Periodicals in 2001 (Jinha, 2010).

science and engineering, incorporating elements of applied physics, biology, and chemistry and applying methodologies from almost every area of engineering.

This method of learning about materials from existing literature presents a plethora of multi-faceted challenges. These include (i) designing and implementing big data systems architecture; (ii) developing computational methods for aggregating text data over time, dating as far back as 200 years; (iii) developing computational models for clustering materials based on the connectivity of their underlying structures, their material properties, the processes that change the materials, and their structures; and (iv) documenting the functions and performance of a large number of materials.

With the visualization tool, there are challenges especially involving usability issues. These include how well users understand the role of low-level components of the tool, the extent of domain knowledge of users in order to interpret the contents presented by the tool, learning and sharing various principles and skills of visual communication and semiotics among collaborators potentially using the tool, and scalability as the quantity of data being interacted with continues to grow exponentially (Chen, 2005).

3 THE GENERAL RESEARCH FRAMEWORK

In this section, we describe a general framework for comprehensive scientific learning and accelerated discovery in the digital age, beyond material science alone. The proposed four-layer learning and discovery architecture is illustrated in Fig. 3, where all four layers contribute in distinct ways to the overall process of acceleration of discovery.

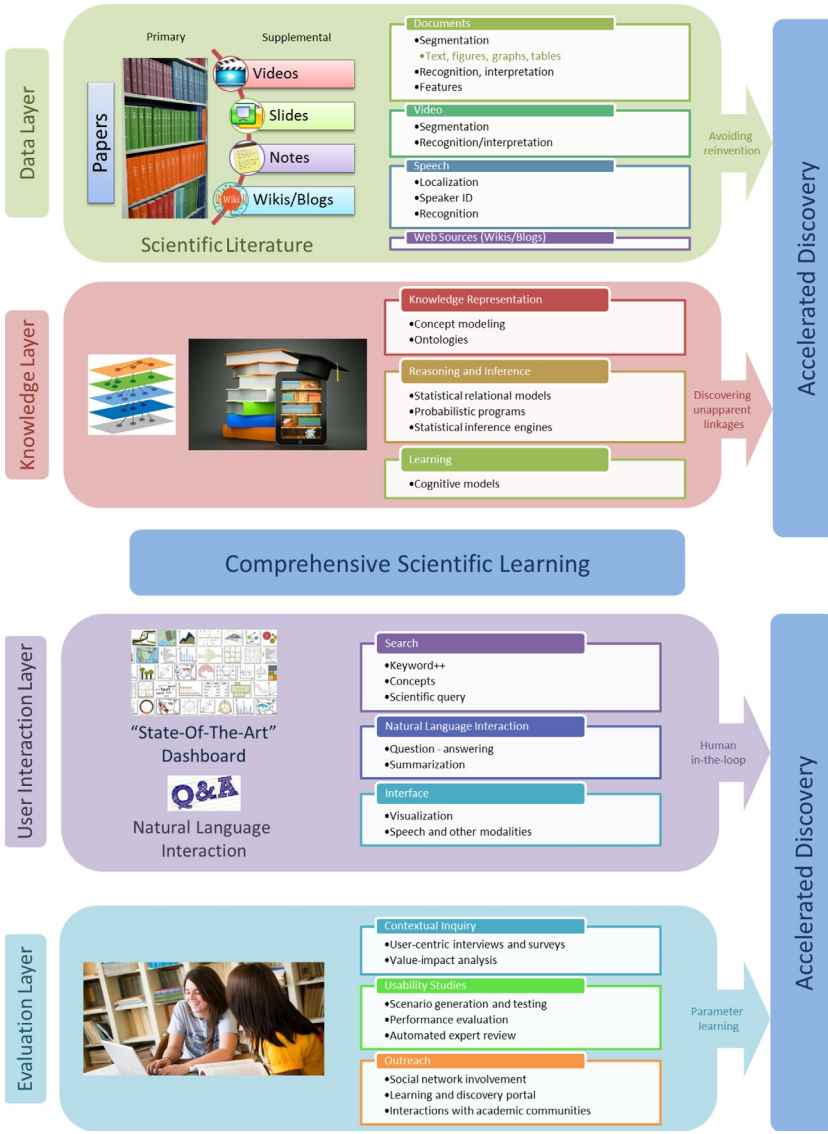


FIGURE 3 Overview of proposed accelerated discovery framework.

- The data layer involves the low-level and mid-level processing (segmentation, recognition, interpretation) of disparate data from multiple sources. The data types include documents, videos, slides, blogs, etc. An additional objective is to aggregate the different representations of the same concept at the data level. The data layer embodies all scientific material on a topic and

contributes to the process of discovery by highlighting the methods and ideas already explored and overlaps between items so that reinvention of the wheel can be avoided.

- The knowledge layer encompasses the tasks pertaining to enabling knowledge capture for scientific search by developing fundamental theories and methodologies to organize the output of the data layer. This layer focuses on (i) collaboratively utilizing human intuition and machine-based algorithms to specify how the data is to be represented conceptually; (ii) predictions based on known concepts learned from training data; and (iii) discoveries of previously unknown concepts hidden in the data. The consolidated knowledge base which represents the digest of the data is essential preparation to enable IR, Q&A, and summarization. It is instrumental in highlighting otherwise unapparent linkages across the scientific items which are often the source of major discoveries.
- The user interaction layer addresses the research needs for developing human–computer interfaces that can support the vision of this learning platform. Some of these technologies include advances in search (text, voice) beyond keywords such as natural language queries and question answering for interacting with the knowledge base, advances in text summarization such as the automatic transformation of the results of a complex query into a *wiki* format that will in turn lead to insightful visualization designs. The user interface layer will also look at human-in-the-loop approaches, studying continuous interactions between machine-prompted knowledge nuggets and the human researcher consuming the data, in order to learn and predict user intent and personalize the visualization interface.
- The evaluation layer is focused on studying how potential users of the expedition will interact with and be impacted by the learning framework, thus addressing the issues of value and impact in the societal context. This research area especially focuses on the predesign and postimplementation evaluation of the technologies developed, to ensure that they result in improved usability and access. This layer has the dual role of evaluating the effectiveness of the framework by comparing multiple learning methods and discovery paradigms using control groups of students and researchers. Additionally, we expect this framework to be beneficial to practitioners also—e.g., this can help an engineer looking to apply methods from literature, to wade through the complex alchemy of algorithms and parameters quickly and speed up a process that usually take years to master. Given that the products of scientific research are themselves data in our framework, we expect our data-driven modeling to highlight the choice of the best techniques and associated parameters that have proven successful in similar problem domains.

This general framework therefore addresses the development of a discovery environment that facilitates rapid and comprehensive scientific learning, and enables accelerated discovery. The framework incorporates the following attributes:

1. Efficient querying and search of the current state of the art
2. Development of a representation framework that captures the gist of knowledge embedded in the scientific literature
3. Consolidation of the knowledge base by integration of scientific publications with corresponding online resources such as video lectures, tutorials, blogs, and wikis
4. Acceleration of discovery by avoiding "reinvention of the wheel" and by making explicit unapparent linkages in the scientific body of literature.

Typically, we have looked to scientific literature for an authoritative, verifiable assessment of the current status of our knowledge for a given problem or field. While full-text search of the document is feasible for most literature today, for scientific discovery it is in many cases not linear text but other elements, such as the hypothesis/argument/evidence structure, the tables, and the graphs, that are of greater importance. The nature of the questions that a researcher poses and the type of answers desired involve understanding the deeper meaning in scientific literature, and we believe that this requires a paradigm shift in the approach to search technologies (the framework for reasoning, clustering, indexing, and retrieval) that will enable fundamentally different types of queries.

Today, the advances in computing, storage, and machine learning algorithms make it possible for the entire scientific literature of any given field to be examined in its totality, so that papers across topics, years, authors, disciplines, and institutions can reveal linkages that have been thus far unapparent and which could lead to transformational discoveries. Our proposal takes on this grand challenge by developing a framework that would scale across disciplines and accelerate the entire process of scientific endeavor and discovery. The proof of concept will be established on the basis of three disparate scientific domains: (i) materials science, (ii) life sciences, and (iii) computer science. The testing and evaluation of the outcomes of this proposal will be facilitated by support from publishers of academic journals, and our experimental test bed will consist of a million articles from a broad range of fields. We propose to target the journal family of the American Physical Society (APS) (such as *Physical Review B*) for the area of materials science and full participation, NIH Portfolio journals from PubMed for the life sciences, and Open Access journals for computer science. An online learning and discovery portal will be developed to demonstrate the outcomes.

4 PILOT IMPLEMENTATION

We will develop a *proof-of-concept* application, specifically narrowing down the field of scientific articles to archives provided by the APS for four major physics journals: (i) *Physical Review* (all years), (ii) *Physical Review Letters* (all years), (iii) *Physical Review X* (all years), and *Reviews of Modern Physics* (all years). And in spite of its substantial commercial value, the APS has freely provided this data set to the UB for research-only purposes.

4.1 Objective 1: To Design and Develop a Time-Based, Hierarchical Topic Model

4.1.1 Problem Description

Currently, working with online documents typically involves using two tools—search and links—we type keywords into a *search engine* and examine the returned results in the form of *ranked list of links to the documents*. But we propose a method for interacting in a different manner with scholarly articles online, by searching, exploring, and visualizing the documents based on themes. Scientific articles can now be hierarchically clustered with respect to their relevance to identified themes, and users could explore how these themes are related to each other. Also, the relationships between specific themes and the associated subthemes, between meta-data such as authors and themes, and how themes change over time can be explored. Currently, we do not interact with large collections of scholarly documents in this manner, primarily because human annotations of thematic structures are virtually impossible at this scale. But the similarity in structure of scientific articles across seemingly different interdisciplinary subject areas of material research can be explored deeply to discover thematic commonalities, the use of analogous methods/algorithms across fields, and other semantic content-based knowledge (Huang, 2008). Hence, the goal of the proposed work is to develop machine learning-based mechanism for this type of deep exploration within large collections.

4.1.2 Prior Work

Studying the trend of scientific ideas over time was earlier studied by Kuhn in 1962 (Kuhn, 1962). In Kuhn's model, science is viewed as shifting from one paradigm to another; since researchers' ideas and vocabulary are constrained by their paradigm, successive less compatible paradigms will have a different vocabulary and framing. Although Kuhn's model was intended to apply only to very large shifts in scientific thought, rather than at the microlevel of trends in research, this insight that vocabulary and vocabulary shift is a crucial indicator of ideas and shifts in ideas has been explored by several researchers in the machine learning and information engineering communities (Hall et al., 2008). A related issue is that of analyzing culture changes using millions of digitized books (Michel et al., 2011).

Latent Dirichlet allocation (LDA) (Blei et al., 2003) also known as unsupervised topic modeling was first published in 2003 and is the most basic idea of probabilistic topic (or theme) modeling. It is assumed that a fixed number of "topics" are distributions over words in a fixed vocabulary, in the entire document collection, so that LDA provides a method for automatically discovering topics that the documents collectively contain. Other more advanced methods of discovering latent hierarchies based on unsupervised learning of densities and

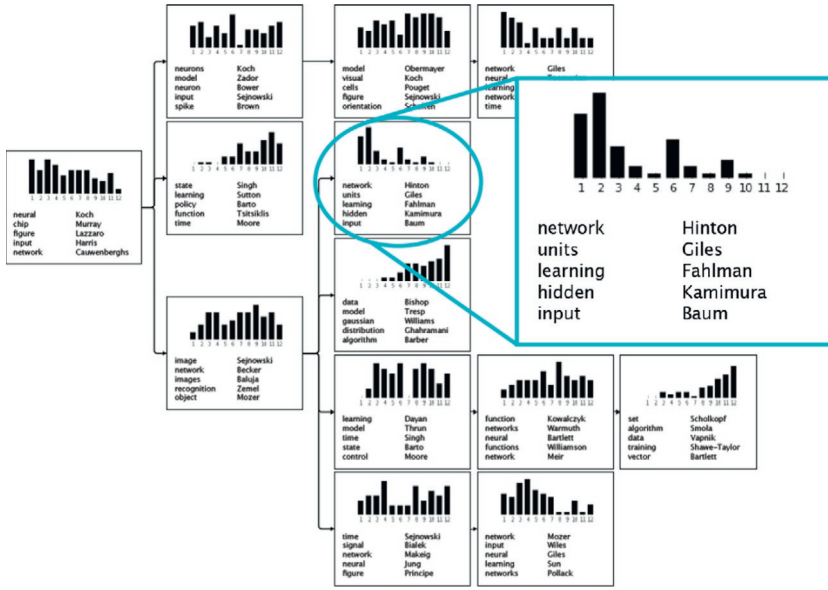
nested mixtures include finite-depth trees (Williams, 1999), diffusive branching processes (Neal, 2003), and hierarchical clustering (Heller and Ghahramani, 2005; Teh et al., 2008). Other latent hierarchical Bayesian approaches include semi-supervised learning (Kemp et al., 2003), relational learning (Roy et al., 2006), and multi-task learning (Daumé, 2009). Most recently, evolutionary diffusion processes have been proposed to capture the tree-like, hierarchical structure of natural data (Adams et al., 2010; Meeds et al., 2008; Paisley et al., 2012).

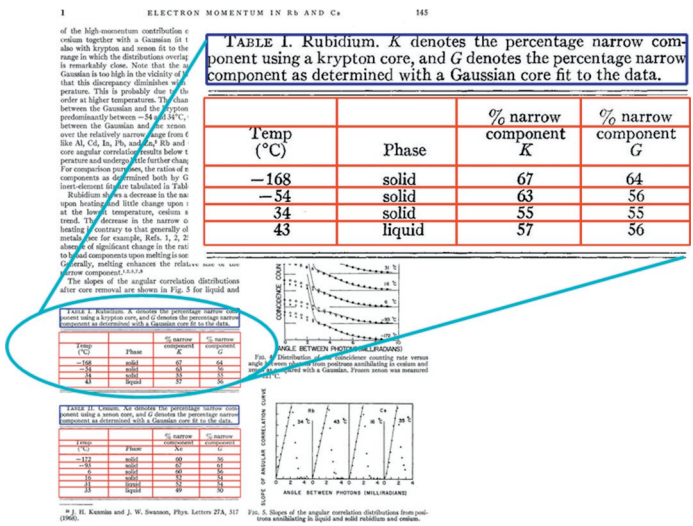
The Dynamic Topic Model (Blei and Lafferty, 2006) is an example of how to model temporal relationships by extending the standard LDA, where each year's documents are assumed to be generated from a normal distribution centroid over topics, and the following year's centroid is generated from the preceding year's, with a Markov chain type of relationship. The Topics over Time Model (Wang and McCallum, 2006) assumes that each document chooses its own time stamp based on a topic-specific beta distribution. These two models however impose strong constraints on the time periods. Along these line, we also implemented a dynamic topic model published in the *Journal of Machine Learning* (Malgireddy et al., 2013). In this model, we learned the relationships between the input observables also as a Markov chain type of relationship and used this model to cluster and classify human activities in large collections of videos. An example of a subtree of documents inferred using 20 topics is presented in Fig. 4 (left), where only the nodes with at least 50 documents are shown.

4.1.3 Research Contributions

Based on the assumption that materials research scholarly data can be naturally modeled by an unobserved hierarchical structure, we build upon the unsupervised tree-structured, hierarchical nonparametric Bayesian model originally proposed by (Adams et al., 2010). A flexible nonparametric prior is placed over unknown data hierarchies, and nested stick-breaking processes are implemented to allow for the generation of trees of unbounded width and depth. This method was applied to the hierarchical topic modeling of documents from the publicly available data set, NIPS 1-12[2], and an example of a subtree of documents inferred using 20 topics is shown in Fig. 4 (left), where only nodes with at least 50 documents are shown. Each node shows the five most common author names, the five most common words, and a histogram over the years of proceedings. We will implement such a model over a significantly larger and more diverse collection of scholarly material research articles, for a distribution over random measures that also construct a natural hierarchy on the thematic data. As in LDA, we will use the bag-of-words topic model in our approach. Where in LDA, each document has a unique topic distribution, in this model, each document lives at a node and that node has a unique topic distribution. Thus, multiple documents share a distribution over topics if they exist at the same node. Each node's

2. http://cs.nyu.edu/~roweis/data.html.





1 ELECTRON MOMENTUM IN Rb AND Cs 145

of the high-momentum contribution c
cesium together with a Gaussian fit t
also with krypton and xenon fit to the
range in which the distributions overlap
is remarkably close. Note that the
Gaussian is too high in the vicinity of
that this discrepancy diminishes wi
perature. This is probably due to th
order at higher temperatures. The chan
between the Gaussian and the krypton
predominantly between −54 and 34°C,
between the Gaussian and the xenon
over the relatively narrow range from
like Al, Cd, In, Pb, and Ag.[5] Rb and
core angular correlation results below t
perature and undergo little further chan
For comparison purposes, the ratios of
components as determined both by G
inert-element fits are tabulated in Tabl
Rubidium shows a decrease in the na
upon heating and little change upon
at the lowest temperature, cesium s
trend. The decrease in the narrow c
heating is contrary to that generally ob
metals (see for example, Refs. 1, 2, 2
absence of significant change in the rati
to liquid components upon melting is sur
Generally, melting enhances the relative
narrow component.[4,6,7,8]
The slopes of the angular correlation distributions
after core removal are shown in Fig. 5 for liquid and

TABLE 1. Rubidium. K denotes the percentage narrow component using a krypton core, and G denotes the percentage narrow component as determined with a Gaussian core fit to the data.

Temp (°C)	Phase	% narrow component K	% narrow component G
−168	solid	67	64
−54	solid	63	56
34	solid	55	55
43	liquid	57	56

TABLE II. Cesium. K denotes the percentage narrow component using a xenon core, and G denotes the percentage narrow component as determined with a Gaussian core fit to the data.

Temp (°C)	Phase	% narrow component K	% narrow component G
−172	solid	60	56
−93	solid	67	61
6	solid	60	56
16	solid	53	54
31	liquid	52	54
35	liquid	49	50

[6] J. H. Kunsman and J. W. Swanson, Phys. Letters 27A, 317 (1968).

FIG. 5. Slopes of the angular correlation distributions from positrons annihilating in liquid and solid rubidium and cesium.

FIGURE 4 Top: A subtree of documents from NIPS 1-12 (modified from Adams et al., 2010) showing the hierarchy on thematic documents data. Bottom: A sample of our table and caption detector (a current work in progress) on a physics article published in 1968. Image best viewed in color.

topic distribution is sampled from a chained Dirichlet-multinomial and each word is drawn from a distribution over depths that is given by a stick-breaking model.

An implementation such as this can capture natural events such as the time period when certain materials begin to take a lion's share of attention in the materials research world. Studies of these materials spawn new areas of research and cause new materials to be discovered. Hence, based on the data obtained from our corpus of scholarly articles, such materials would be the natural ancestors of more specialized materials that followed on from the new ideas on these major materials. An example of such an event would be the isolation of graphene (a crystalline form of carbon) in 2004 which has led to an explosion in graphene research. This desirable feature of having a distribution over hierarchies naturally allows us to have access to the notion of ancestor materials data and their descendants. Correspondingly, we also gain access through our model to the seminal papers leading to the discoveries of the ancestor materials as well as the papers that have followed from such a research boom.

It is important to note that the current way of tracking this type of explosion in an area is the traditional way of following the references. One takes a seminal paper in the past, finds out which articles reference it by doing a so-called *reverse literature search*, and then determine which other articles reference the second set, and so forth. This is the usual manner in which early-stage literature searches are done by researchers in materials, when starting out in an area. Our proposed method using advanced topic modeling goes beyond the scope of just a reverse literature search, so that even without a large citation base, the first papers containing the material names or certain associated keywords become the ancestors on the hierarchical tree structure. This will provide a more complete set of "pioneering papers" in the area than those linked by references, thus enabling researchers to more readily access these pioneering works that might be "hidden" in the large document corpus.

Our goal is to obtain prototype materials (from scholarly documents) at different dimensions of interest and create phenomena-based links between the prototypes and other materials within the hierarchy. A major attraction for this proposed statistical modeling method is the practical inference approach based on Markov chain Monte Carlo which has been demonstrated on several real-world data sets in different domains by Adams et al. (2010).

The key intellectual contribution of this work is the machine learning-based extensions we propose on the class of topic models, by extending the flat, single-level LDA model to a hierarchical tree-structure to suit our predefined concept types, and also to simultaneously incorporate a dynamic element into this hierarchy in order to capture the trends in materials discovery over the years, linked across subfields or topical areas in related materials.

4.2 Objective 2: To Implement Algorithms for Extracting Text from *x–y* Plots and Tables

4.2.1 Problem Description

The objective of DIA is to recognize the text and graphics components in images and to extract the intended information as a human would (Blostein et al., 2000; Kasturi et al., 2002; Lu et al., 2009). Although there are several categories of document image processing (Balasubramanian et al., 2006), we are dealing primarily with textual processing in this research work. Textual processing involves extracting and generating meta-data from the text components of a document image. Textual processing involves a significant amount of graphics processing, which is necessary to separate graphical components from textual ones (Futrelle et al., 2003; You et al., 2011). The two classes of document images that we are concerned with in this work are (i) *x–y* plots in experimental physics articles and (ii) tabular structures. The tasks involved in their analysis include recognizing their location on a page, and further analysis to separate the text in the graphics or tables from the background, surrounding lines, and curves. We will be employing image processing and machine learning techniques for this type of processing. After applying these text-from-graphics analysis techniques, meta-data such as material properties, the metrics for the properties, and the range of values are also culled from the processed documents, to yield a much more concise and complete semantic description of the articles in the corpus.

4.2.2 Prior Work

Content-based image retrieval (Datta et al., 2008; Smeulders et al., 2000) is a very mature, yet on-going, open area of research. Specifically, many of the DIA algorithms found in use today were developed in the 1990s and early 2000s (Feng, 2009; Kim and Govindaraju, 1997; Madhvanath et al., 2009; Shi et al., 2005), when document analysis was a very active research area. Some of these works included text extraction historical documents, recognition of U.S. census forms, bank check recognition, etc. Many of these were heuristic-based algorithms that have done well on small volumes of data with clean well-defined data structures, but are not at all expected to scale to the large data set we are working with. Currently existing open source processes of turning PDF documents into text are incapable of handling equations and tables. Pande (2002) ran a few experiments of table understanding for IR purposes, but on clean table data with known structure. Some machine learning-based models that have shown promising results when applied to image categorization include Barnard et al. (2003) and Li et al. (2009). Similarly, by extracting features of patches from images of printed documents and using a Bayesian generative model, document segmentation can be accomplished without attempting text extraction (Burns and Corso, 2009) or using optical character recognizers (OCRs). A discrete patch-based codebook was learned over regions in the document and the

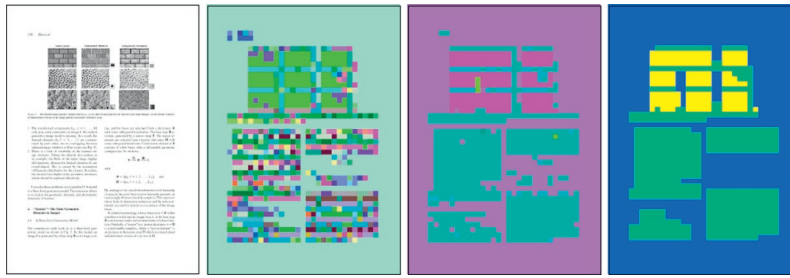


FIGURE 5 From left to right are the original image, the encoded document, the topic map, and the final segmentation. *Source: Image obtained from Burns and Corso (2009).*

latent variables presenting the region were measured as a distribution over the patch indices. Figure 5 is an example of the results of the automated document separation, without OCR.

4.2.3 Research Contributions

The main thrusts for this objective are

A. To extract plots, search on figure captions and axis labels, and analyze plotted numerical data

- Enable search on figure captions
- Classify plots/figures into x–y plots and those not (typically diagrams, photos, etc.)
- Extract axis labels and data from a plot into a table of x and y values
- Handle plots which are inset within other plots and plots which have multiple curves, especially curves which cross each other. Curves will be distinguished usually by color or symbol shape, but not always. Use slopes of curves to estimate which curve is which
- Comparatives: ask the question: "which graphs are the same or similar, among different papers, based on the shape of the curve and what is being plotted?" (e.g., the maximum resistivity for Co compounds)

B. To extract data from tables

- Search on table headers and collect properties data
- Identify the data range values for the properties collected
- Extract data from bar graphs
- Separate columns, rows, and cells in tables and associate them with the correct materials and properties
- Associate tables with paper section headings and perform advanced analytics of surrounding text to get associations

C. To enable usability

- For each document, or group of documents, make a list of all the tables (with the extracted tables) and a list of all the figures (with extracted figures), with their associated figure/table captions
- Index the *x*–*y* plots, index the key words in the figure caption. Search and return figures which contain those key words. Link the PDF of the paper to the figures/tables

The main implementation tool for separating printed text from surrounding printed material is the DBN, to be used for extracting the text in figures (specifically *x*–*y* plots) and tables. DBNs are generative models with multiple, densely connected layers of nonlinear latent variables and have exponentially more representational power than mixture models because many variables in a layer can simultaneously contribute when generating data. Figure 6 shows a two-layer DBN, where layers are shown by dashed boxes. The observed variables v are the inputs (from scanned images of articles) to the first layer and are used to learn weights W_1. The reconstructed hidden variables h_1 from the first layer become the inputs to the second layer and W_2 is learned similarly, while keeping W_1 fixed.

We propose the use of such a deep architecture for modeling the features that represent the different parts of a scientific article. DBNs have been used extensively in the last few years for feature extraction, especially for large data sets (Hinton, 2007). Their efficient extension to 2D images was presented by Ranzato et al. (2010, 2011). The technical challenges primarily include (i) how to structure the scanned article data and present them to the deep network as observed variables. The options include the use of convolutional kernels, unfiltered image patches, whole static images downsampled to very small sizes, etc.; (ii) the choice of the optimization criteria when reconstructing the inputs to a layer; an appropriate energy function to minimize during reconstruction has to be designed and proven theoretically; (iii) the choice of a classifier that will

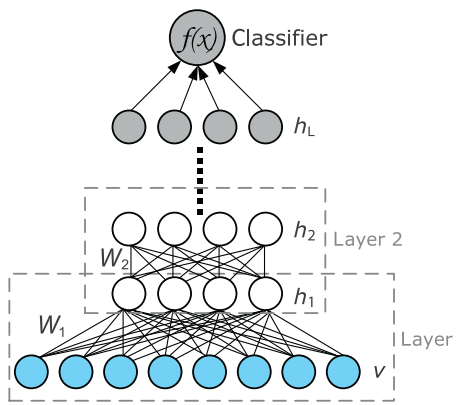


FIGURE 6 The DBN architecture.

perform optimal optical character recognition of the extracted items; and lastly, (iv) the tuning of the design parameters (e.g., number of layers, full or partial connectivity between layers) to obtain the best features for classification. We currently have very promising results from isolating *x*–*y* plots in articles and separating text from the surrounding printed materials in the plots, as shown in Fig. 4 (right).

4.3 Objective 3: To Develop an Interactive, Materials Network Visualization Tool

4.3.1 Problem Description

The proposed visualization tool has several different purposes (i) to display the results of advanced queries involving material properties and data ranges-of-interest, presenting the query results in the form of figures, tables, and other informative objects in the papers; (ii) to display a graphical histogram-like structure of the frequency of publications occurring for a material hierarchy and each of its associated processes; and (iii) lastly, to provide an easy, visual graphical network-like mechanism for the researcher to review the relationships between different materials over time, across subtopical areas. Ultimately, the tool will provide the researcher with an interface to interact with the large, complex scholarly data and to access the necessary scientific papers related to the specific research task at-hand.

The goal of this visualization tool is to close the loop of information selection, preparation, and visualization. The tool will accelerate the process of getting from large, complex, unwieldy data to general understanding and back to more specific, relevant data, in a way that can be easily understood and interacted with. For example, a scientist studying a more common material wants to know, over all the decades (which in physics can be upward of 100 years) of experiments, whether this material has ever been measured under the conditions of. . .a certain temperature range, or a certain pressure, or under a certain magnetic field, or under a certain electrical field, or measured with a certain spectroscopy, or a certain thermodynamic measurement (specific heat, susceptibility, thermopower, etc.). If the material is a common one, that information might exist in an obscure paper published so long in the past, that it will be hard to review all the various suggestions, say from a *Google Scholar*® search, looking for the right range of the property. A visualization tool such as we propose can display the processes that have been associated with materials and that links materials together. For example, the existence of the relationship between silicon carbide (SiC) and graphene—when SiC is heated to high temperatures ($> 1100\,^{\circ}\mathrm{C}$) under low pressures ($\approx 10^{-6}$ torr), it can be reduced to graphene. The example in Fig. 7 shows a set of associations across a broad range of papers and time. It demonstrates the sort of hierarchies of relationships created not just by linking papers which mention each other in their list of references, but are linked thematically by subject matter and approach.

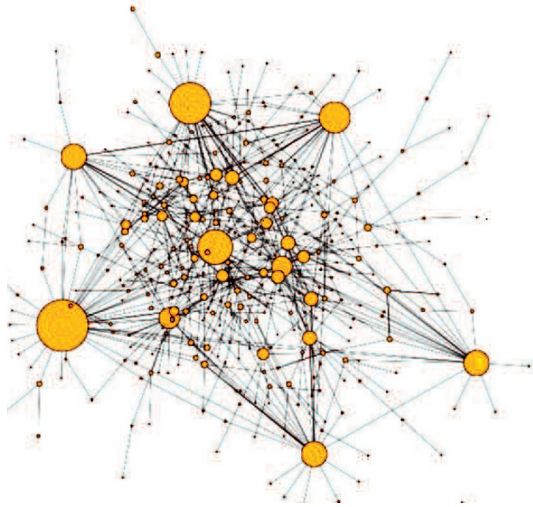


FIGURE 7 An example of a graphical structure potentially showing a network of the clustered hierarchies of materials. The size of the cluster represents its frequency in the data set and its connections represent the associations with other materials.

4.3.2 Prior Work

There has been a large body of work over the past 60 years to uncover patterns in large, complex data sets (Perer and Shneiderman, 2009; Shneiderman, 2002), and Wasserman and Faust (1994) is perhaps the most widely used reference book for many of the earlier methods (Shneiderman and Plaisant, 2006). There have also been several projects focusing on improving interactive exploration with networks. From a review in Shneiderman and Plaisant (2006), GUESS is a graph exploration system that combines an interpreted language with a graphical front-end (Adar, 2006). TreePlus allows users to explore graphs using more comprehensible enhanced tree layouts (Lee et al., 2006). JUNG (the Java Universal Network/Graph Framework) is an open source graph modeling and visualization framework written in Java to provide users with a framework to build their own social network analysis tools (Madadhain et al., 2005). Processing (Fry, 2008) is a programming language and development environment aimed specifically at implementing many different flavors of these visualization programs.

Building the temporal element into visualization techniques over large data sets is not quite as active an area of visualization research, but is also starting to get some attention especially with geo-based spatiotemporal data (Compieta et al., 2007). The Java3D visualization tool (Java3D Web site, n.d.) provides user interaction in a non-geo-referenced space, making it more oriented to data-mining experts.

4.3.3 Research Contributions

Our goal is to build a multi-faceted visualization tool using the existing platform provided by Processing (Fry, 2008). Processing enables us to write code which is in turn compiled into Java. There is also a Processing javascript which will be very useful for deploying this tool online, as Web sites choosing to use the Processing-based tool can be deployed without Java applets. As an added bonus, Processing has been available open source, for several years; hence, there is a large number of examples and added code from the community, to accelerate our development process. The current research items for the visualization tool include

1. Developing tight integration between the automatic model computations implemented from Sections 4.1 and 4.2 and the visualization tool;
2. Incorporating a significant amount of properties learned from the scientific articles (meta-data) into the tool;
3. Implementing perceptual, cognitive, and graphical principles in the tool;
4. Optimizing the visualization techniques to allow researchers interactively explore interaction techniques, such as focus and context;
5. Learning adaptive algorithms for users' intent, in order to adapt the visualization tool parameters based on the user's preferences and the data selected.

4.4 Testing and Validation

We are developing a *proof-of-concept* materials research application, whose front-end is the visualization tool and its back-end includes the output of the time-based, hierarchical topic model. Our *proof-of-concept* targets only the scientific articles in the archives provided to us by the APS for four major physics journals. Their first papers were published between 1968 and 1970 so that we have at least 40+ years of publications per journal. The archive includes all volumes and issues within the publication year.

The significantly large number of documents (and clusters of documents) being analyzed strongly suggests that many of the traditional algorithms for evaluating topic models will need to be replaced by ones that scale better. The primary author of the topic modeling paradigm, David Blei, has done extensive work on running topic models on large data sets of publication data, where he ran a 100-topic algorithm on all the *Science* articles from 1882 till 2001 (Blei and Lafferty, 2007). The code for this work has been tested and optimized for large data sets and is made publicly available at http://topics.cs.princeton.edu/Science/. Although we will be extending this initial model significantly, we will begin our implementation by building upon tried-and-tested codebases such as this.

Duchi et al. (2012) presented an approach to supervised ranking based on aggregation of partial preferences, using statistic-based empirical risk minimization procedures. This approach showed very consistent results in large

data sets for a large-scale Web-ranking task. Due to the large scale of this project, we will implement similar risk minimization procedures for evaluating the validity of our query results. High-performance computing techniques will be employed in the visualization module to allow for any scalability issues that might result from interacting with the large size of the documents corpus.

REFERENCES

Adams, R.P., Ghahramani, Z., Jordan, M.I., 2010. Tree-structured stick breaking for hierarchical data. In: NIPS. pp. 19–27.

Adar, E., 2006. GUESS: a language and interface for graph exploration. In: Proceedings of the ACM Conference on Human Factors in Computing Systems.

Adler, R., Ewing, J., Taylor, P., 2012. Citation statistics. Report from the International Mathematical Union. URL http://www.mathunion.org/fileadmin/IMU/Report/CitationStatistics.pdf.

Balasubramanian, A., Meshesha, M., Jawahar, C.V., 2006. Retrieval from document image collections. In: Document Analysis Systems (DAS). pp. 1–12.

Barnard, K., Duygulu, P., de Freitas, N., Forsyth, D.A., Blei, D.M., Jordan, M.I., 2003. Matching words and pictures. J. Mach. Learn. Res. 3, 1107–1135.

Blei, D., Lafferty, J., 2007. A correlated topic model of science. Ann. Appl. Stat. 1, 17–35.

Blei, D., Lafferty, J.D., 2006. Dynamic topic models. In: Proceedings of the 23rd International Conference on Machine Learning, ICML. ACM.

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Blostein, D., Lank, E., Zanibbi, R., 2000. Treatment of diagrams in document image analysis. In: Proceedings of the International Conference on Theory and Application of Diagrams, vol. 2. pp. 330–344.

Brooks, T., 2009. Timing and location count when announcing particle physics results. Symmetry Magazine. URL http://www.symmetrymagazine.org/breaking/category/spires/.

Burnham, J., 2006. Scopus database: a review. Biomed. Digit. Libr. 3 (1).

Burns, T.J., Corso, J.J., 2009. Robust unsupervised segmentation of degraded document images with topic models. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition.

Chen, C., 2005. Top 10 unsolved information visualization problems. IEEE Comput. Graph. Appl. 25 (4), 12–16.

Compieta, P., Martino, S.D., Bertolotto, M., Ferrucci, F., Kechadi, T., 2007. Exploratory spatio-temporal data mining and visualization. J. Vis. Lang. Comput. 18 (3), 255–279.

Datta, R., Joshi, D., Li, J., Wang, J.Z., 2008. Image retrieval: ideas, influences, and trends of the new age. ACM Comput. Surv. 40 (2), 1–60.

Daumé III, H., 2009. Bayesian multitask learning with latent hierarchies. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. AUAI Press, pp. 135–142.

Davis, J.R., Lagoze, C., 1995. Dienst: an architecture for distributed document libraries. Commun. ACM 38, 38.

DiMasi, J., Hansen, R., Grabowski, H., 2003. The price of innovation: new estimates of drug development costs. J. Health Econ. 22, 151–185.

Dogan, R.I., Murray, G.C., Névéol, A., Lu, Z., 2009. Understanding pubmed® user search behavior through log analysis. Database 2009. http://dx.doi.org/10.1093/database/bap018.

Duchi, J.C., Mackey, L., Jordan, M.I., 2012. The Asymptotics of Ranking Algorithms. arXiv e-prints

Feng, S., 2009. Statistical Models for Text Query-Based Image Retrieval. PhD thesis, University of Massachusetts.

Frontiers in Massive Data Analysis—NIST Big Data Working Group, n.d. URL http://bigdatawg.nist.gov/FrontiersInMassiveDataAnalysisPrepub.pdf.

Fry, B., 2008. Visualizing Data—Exploring and Explaining Data with the Processing Environment. O'Reilly Media, Sebastopol, CA, pp. I–XIII, 1–366 .

Futrelle, R.P., Shao, M., Cieslik, C., Grimes, A.E., 2003. Extraction, layout analysis and classification of diagrams in pdf documents. In: Proceedings of the International Conference on Document Analysis and Recognition. p. 1007.

Hall, D.L.W., Jurafsky, D., Manning, C.D., 2008. Studying the history of ideas using topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP'08. Association for Computational Linguistics, pp. 363–371.

Haque, A., Ginsparg, P., 2009. Positional effects on citation and readership in arXiv. arXiv:0907.4740

Heller, K.A., Ghahramani, Z., 2005. Bayesian hierarchical clustering. In: Proceedings of the 22nd International Conference on Machine Learning, ICML. ACM, pp. 297–304.

Hinton, G.E., 2007. Boltzmann machine. Scholarpedia 2 (5), 1668.

Huang, A., 2008. Similarity measures for text document clustering. In: NZCSRSC. pp. 49–56.

Jacso, P., 2010. Metadata mega mess in Google Scholar. Online Inf. Rev. 34, 175–191.

Java3D Web site, n.d. URL http://java.sun.com/products/java-media/3D/S.

Jin, W., Srihari, R., Wu, X., 2007. Mining concept associations for knowledge discovery through concept chain queries. Adv. Knowl. Discov. Data Min. 4426, 555–562.

Jinha, A., 2010. Article 50 million: an estimate of the number of scholarly articles in existence. Learn. Publ. 23, 258–263.

Kasturi, R., O'Gorman, L., Govindaraju, V., 2002. Document image analysis: a primer. Sadhana 27 (1), 3–22.

Kemp, C.C., Griffiths, T.L., Stromsten, S., Tenenbaum, J.B., 2003. Semi-supervised learning with trees. In: NIPS. MIT Press.

Kim, G., Govindaraju, V., 1997. Bank check recognition using cross validation between legal and courtesy amounts. Int. J. Pattern Recognit. Artif. Intell. 11 (4), 657–674.

King, D.W., 2006. Measuring total reading of journal articles. D-Lib Magazine 12 (10), 71–122. URL http://www.dlib.org/dlib/october06/king/10king.html.

Konrad, A., 2007. On inquiry: human concept formation and construction of meaning through library and information science intermediation. PhD thesis, University of California, Berkeley.

Kuhn, T.S., 1962. The Structure of Scientific Revolutions. University of Chicago Press, Chicago, IL.

Larsen, P.O., von Ins, M., 2010. The rate of growth in scientific publication and the decline in coverage provided by science. Scientometrics 84, 575–603.

Lee, B., Parr, C.S., Plaisant, C., Bederson, B.B., Veksler, V.D., Gray, W.D., et al., 2006. TreePlus: interactive exploration of networks with enhanced tree layouts. IEEE Trans. Vis. Comput. Graph. 12 (6), 1414–1426.

Li, L.-J., Socher, R., Li, F.-F., 2009. Towards total scene understanding: classification, annotation and segmentation in an automatic framework. In: CVPR. pp. 2036–2043.

Lu, X., Kataria, S., Brouwer, W.J., Wang, J.Z., Mitra, P., Giles, C.L., 2009. Automated analysis of images in documents for intelligent document search. Int. J. Doc. Anal. Recognit. 12, 65–81.

Madadhain, J., Fisher, D., Smyth, P., White, S., Boey, Y.B., 2005. Analysis and visualization of network data using JUNG. J. Stat. Softw. 10, 1–25.

Madhvanath, S., Govindaraju, V., Srihari, S., 2009. Recognition of US census forms. Int. J. Imaging Sci. Technol. 7, 312–319.

Malgireddy, M.R., Nwogu, I., Govindaraju, V., 2013. Language-motivated approaches to action recognition. J. Mach. Learn. Res. 14, 2189–2212. URL http://jmlr.org/papers/v14/malgireddy13a.html.

Meeds, E., Ross, D., Zemel, R., Roweis, S., 2008. Learning stick-figure models using nonparametric Bayesian priors over trees. In: Computer Vision and Pattern Recognition (CVPR), 2008 IEEE Conference on. pp. 1–8.

Michel, J.B., Shen, Y., Aiden, A., Veres, A., Gray, M., Pickett, J., et al., 2011. Quantitative analysis of culture using millions of digitized books. Science 331, 176–182. URL http://www.sciencemag.org/content/331/6014/176.full.html.

Neal, R., 2003. Density modeling and clustering using Dirichlet diffusion trees. Bayesian Statistics. pp. 619–629.

Paisley, J.W., Blei, D.M., Jordan, M.I., 2012. Stick-breaking beta processes and the Poisson process. J. Mach. Learn. Res. Proc. Track 22, 850–858.

Pande, A., 2002. Table understanding for information retrieval. MSc thesis, Virginia Technical Institute.

Perer, A., Shneiderman, B., 2009. The importance of integrating statistics and visualization: long-term case studies supporting exploratory data analysis of social networks. IEEE Comput. Graph. Appl. 29, 39–51.

Ranzato, M., Krizhevsky, A., Hinton, G.E., 2010. Factored 3-way restricted Boltzmann machines for modeling natural images. J. Mach. Learn. Res. Proc. Track 9, 621–628.

Ranzato, M., Susskind, J., Mnih, V., Hinton, G.E., 2011. On deep generative models with applications to recognition. In: CVPR. pp. 2857–2864.

Roy, D.M., Kemp, C., Mansinghka, V.K., Tenenbaum, J.B., 2006. Learning annotated hierarchies from relational data. In: NIPS.

Shi, Z., Setlur, S., Govindaraju, V., 2005. Text extraction from gray scale historical document images using adaptive local connectivity map. In: Proceedings of the International Conference on Document Analysis and Recognition. pp. 794–798.

Shneiderman, B., 2002. Inventing discovery tools: combining information visualization with data mining. Inf. Vis. 1 (1), 5–12.

Shneiderman, B., Plaisant, C., 2006. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In: Proceedings of the 2006 Advanced Visual Interfaces Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization. Association for Computing Machinery, pp. 1–7.

Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R., 2000. Content-based image retrieval at the end of the early years. IEEE Trans. Pattern Anal. Mach. Intell. 22 (12), 1349–1380.

Teh, Y.W., Daumé III, H., Roy, D.M., 2008. Bayesian agglomerative clustering with coalescents. In: Advances in Neural Information Processing Systems, vol. 20.

Wang, X., McCallum, A., 2006. Topics over time: a non-Markov continuous-time model of topical trends. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD. ACM, pp. 424–433.

Wasserman, S., Faust, K., 1994. Social Network Analysis: Methods and Applications. Cambridge University Press.

White House Materials Genome Initiative (MGI), 2011. Materials Genome Initiative for Global Competitiveness. URL http://www.whitehouse.gov/sites/default/files/microsites/ostp/materials_genome_initiative-final.pdf.

Williams, C.K.I., 1999. An MCMC approach to hierarchical mixture modelling. In: NIPS. pp. 680–686.

You, D., Antani, S., Demner-Fushman, D., Govindaraju, V., Thoma, G.R., 2011. Detecting figure–panel labels in medical journal articles using MRF. In: ICDAR. pp. 967–971.

Chapter 2

An Introduction to Rare Event Simulation and Importance Sampling

Gino Biondini*,[1]
*Department of Mathematics, State University of New York at Buffalo, Buffalo, New York, USA
[1]Corresponding author: e-mail: biondini@buffalo.edu

ABSTRACT
This chapter provides a relatively low-level introduction to the problem of rare event simulation with Monte Carlo methods and to a class of methods known as variance reduction techniques that have been devised to deal with this problem. Special emphasis is given to importance sampling, but several other techniques are also presented, including the cross-entropy method, rejection sampling, and Markov chain Monte Carlo methods such as the Metropolis method and Gibbs sampling. A brief discussion is also given about asymptotic efficiency and the connections with large deviations theory.

Keywords: Monte Carlo methods, Rare event simulation, Variance reduction techniques, Importance sampling, Cross-entropy 2000 *MSC:* 65C05, 65B99

1 INTRODUCTION: MONTE CARLO METHODS, RARE EVENT SIMULATION, AND VARIANCE REDUCTION TECHNIQUES

Since its introduction almost 70 years ago (Metropolis and Ulam, 1949) (see Metropolis, 1987 for a historical review), the Monte Carlo (MC) method has been extensively used in engineering and scientific computing. In their most general interpretation, MC methods are a way to compute integrals. They comprise a collection of techniques for generating random samples on a computer as well as their application to solve a variety of problems. In essence, they involve drawing random or pseudo-random samples from a specific distribution and using them to estimate one or more quantities of interest. Such methods are especially

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00002-2

advantageous over numerical quadrature methods when the dimensionality of the problem is large. As a result, and thanks to their flexibility, such methods have found a wide range of applications (e.g., see Fishman, 1996; Fishman, 2006; Kroese et al., 2011; Landau and Binder, 2000).

A common challenge in MC simulation is that of *rare event simulation*, also referred to as the problem of rare events, where very small probabilities need to be accurately estimated—for example, in reliability analysis, or performance analysis of telecommunication systems. In a nutshell, the problem is that if one needs to quantify the probability of one or more events that occur very rarely, an exceedingly large number of samples are needed even to just produce the desired events, and an even larger number of samples are required to obtain accurate estimates. Other applications that call for rare event simulation are queueing systems (to avoid excessively long waiting times), nuclear physics (avoiding catastrophic accident), security systems (false alarms in radar), material science (technical defects), mathematical science, and insurance.

One approach to overcome the problem of rare events is the use of *variance reduction techniques* (VRTs) (e.g., see the monographs: Bucklew, 2004; Fishman, 1996; Kroese et al., 2011 for general reviews). The general idea behind all of these techniques is to modify the selection of the random samples in such a way that the desired events occur more frequently than they would normally, while simultaneously taking these changes into account in order to obtain unbiased estimates.

Perhaps the most famous VRT is the *importance sampling* (IS) (Fishman, 1996; Kroese et al., 2011; Srinivasan, 2002). The main idea behind IS is to select an appropriate *biasing distribution* (i.e., a change of probability measure) from which to draw the MC samples so that most of the distribution mass falls on the regions of interest. This ensures that many of the MC samples will produce the rare events sought. At the same time, the contribution from each sample is weighted according to the *likelihood ratio*, which ensures that unbiased estimates are obtained.

Of course, for IS to be effective, a good biasing distribution must be chosen. This requires knowledge of which system configurations are likely to produce the rare events of interest. Even though such knowledge is not always available, in many cases it is enough to leverage what is known about the system's behavior in order to guide the choice of biasing distribution, and indeed IS has been used with success in a variety of applications (Biondini et al., 2004; Li et al., 2007; Moore et al., 2008; Smith et al., 1997). (Note that, often, exact knowledge of the most likely failure configurations may not be needed, and an approximate knowledge may be sufficient, since the statistical nature of the MC sampling allows one to take into account the contributions of nearby points in sample space.)

Many other VRTs have also been used with success in various applications, such as multicanonical MC methods (Yevick, 2002), Markov chain Monte Carlo (MCMC) methods (Secondini and Forestieri, 2005), and Gibbs sampling. See

Fishman (1996), Landau and Binder (2000), and MacKay (2003) for a general overview of these methods. The common thread among those VRTs is that they are adaptive. In essence, such methods attempt to find the important regions of sample space numerically. These methods can be applied to problems for which no good choice of biasing distribution is known. When IS is available, however, it is generally advantageous over other methods, because: (i) IS allows one to compute precise error estimates, if desired; (ii) adaptive methods typically require tweaking certain parameters, on which IS is less dependent; and (iii) IS is usually faster than adaptive methods, since, in adaptive methods, a certain portion of numerical simulations needs to be used to look for the most important regions in state space. Indeed, the speed advantage of IS was verified directly in a few cases by a detailed comparison between different methods (Biondini and Kath, 2005; Lima et al., 2005). We should also mention that it is not always necessary to choose between IS and adaptive VRTs. Indeed, yet another technique which has proven to be especially useful in recent years is the cross-entropy method (de Boer et al., 2005; Rubinstein and Kroese, 2004). While it is a useful VRT on its own right, in some cases, the cross-entropy method can also be combined with IS to afford the user the advantages of both IS and those of adaptive techniques.

The remainder of this chapter aims to put the above discussion on a more precise mathematical setting.

2 MC METHODS AND THE PROBLEM OF RARE EVENTS

2.1 MC Estimators

We start with a simple one-dimensional (1D) example. Let X be a random variable (RV) with probability density function (pdf) $p_x(x)$ (Papoulis, 1991). If one defines $Y = y(X)$, where $y(x) = \int_{-\infty}^{x} p_x(x)\,\mathrm{d}x$, it is easy to show that Y is uniform in [0,1]. [To see this, note that $p_y(y)\,\mathrm{d}y = p_x(x)\,\mathrm{d}x$, with $\mathrm{d}y = (\mathrm{d}y/\mathrm{d}x)\,\mathrm{d}x$. But $\mathrm{d}y/\mathrm{d}x = p_x(x)$, so $p_y(y) = 1$.]

Now suppose that we wish to calculate the probability Q that X falls in a range R of interest, namely $Q = \mathbb{P}[\,X \in R\,]$, where $R \subset \mathbb{R}$. We can write Q as

$$Q = \int I_R(x) p_x(x)\,\mathrm{d}x\,. \tag{1}$$

The function $I_R(x)$ is the so-called *indicator function* (or characteristic function) of the set R: namely, $I_R(x) = 1$ if $x \in R$ and $I_R(x) = 0$ otherwise. (Hereafter we will drop the subscript R on I whenever that will not cause ambiguity. Also, integrals without limits are always intended as complete—i.e., over all of sample space—unless specifically noted otherwise.) In particular, we are interested in situations in which it is difficult to compute the above integral analytically.

Making the substitution $x \mapsto y$, we can express Q as $Q = \int I(x(y))\,\mathrm{d}y$. It is therefore natural to try to estimate Q using a frequency count. That is, we draw N independent identically distributed (i.i.d.) random samples $Y_1, \ldots, Y_N$ from a uniform distribution and we write the estimate $\hat{Q}_N = F/N$, where F is the number of samples which fall in the region of interest. More specifically, the above MC estimator is $\hat{Q}_N = (1/N)\sum_{n=1}^{N} I(x(Y_n))$. Equivalently, we can forget about Y and write the above estimator as

$$\hat{Q}_N = \frac{1}{N}\sum_{n=1}^{N} I(X_n)\,, \tag{2}$$

where the i.i.d. random samples $X_1, \ldots, X_N$ are drawn according to the distribution $p_x(x)$. Note that, while Q is a deterministic quantity, $\hat{Q}_N$ is itself a RV. In fact, it is easy to show that

$$\mathbb{E}[\hat{Q}_N] = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}[I(X_n)] = \mathbb{E}[I(X)] = Q\,,$$

where $\mathbb{E}[Z] = \int Z(x)p_x(x)\,\mathrm{d}x$ denotes the expectation value with respect to the pdf $p_x(\cdot)$, which shows that the expectation value of our estimator is indeed the quantity of interest, and

$$\begin{aligned}\mathrm{var}[\hat{Q}_N] = \mathbb{E}[\hat{Q}_N^2] - \mathbb{E}[\hat{Q}_N]^2 &= \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N} \mathbb{E}[I(X_n)I(X_m)] - Q^2 \\ &= \frac{1}{N}\mathrm{var}[I(X)]\,,\end{aligned}$$

where we used the fact that $\mathbb{E}[I(X_n)^2] = \mathrm{var}[I(X)] + Q^2$ and $\mathbb{E}[I(X_n)I(X_m)] = Q^2$ when $n \neq m$ (because X_n and X_m are statistically independent). Note that the above two results are true more generally, i.e., independently of $I(\cdot)$ being an indicator. For an indicator function, in particular, it is $\mathbb{E}[I(X_n)^2] = Q$ (because $I^2(x) = I(x)$) and therefore

$$\mathrm{var}[\hat{Q}_N] = \frac{1}{N}(Q - Q^2)\,.$$

The above results are easily extended to the multidimensional case. Let $\mathbf{X} = (X_1, \ldots, X_D)^T$ be a vector of RVs with joint pdf $p_{\mathbf{X}}(\mathbf{x})$, and suppose that we are interested in the probability $Q = \mathbb{P}[\, y(\mathbf{X}) \in R\,]$, where $y(\mathbf{x})$ is some real-valued function:

$$Q = \int I_R(y(\mathbf{x}))p_{\mathbf{X}}(\mathbf{x})(\mathrm{d}\mathbf{x})\,, \tag{3}$$

where $(\mathrm{d}\mathbf{x}) = \mathrm{d}x_1 \cdots \mathrm{d}x_D$ is the volume element in $\mathbb{R}^D$. More generally, consider integrals of the type

$$Q = \int f(\mathbf{x}) p_{\mathbf{x}}(\mathbf{x})(\mathrm{d}\mathbf{x}) \,, \tag{4}$$

where $f(\mathbf{x})$ is a generic real-valued function. Situations for which calculating the above integral analytically is practically impossible are very common: The dimensionality of the system might be very large, the function $f(\cdot)$ might be complicated, and/or the region R might be complicated.

By analogy with the 1D case, we can define the MC estimator

$$\hat{Q}_N = \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{X}_n) \,. \tag{5}$$

As in the 1D case, we have

$$\mathbb{E}[\hat{Q}_N] = \mathbb{E}[f(\mathbf{X})] = Q \,, \quad \mathrm{var}[\hat{Q}_N] = \frac{1}{N} \mathrm{var}[f(\mathbf{X})] \,. \tag{6}$$

Then, if in particular $f(\mathbf{X}) = I(y(\mathbf{x}))$, we have $\mathrm{var}[\hat{Q}_N] = (Q - Q^2)/N$. The above result implies that the accuracy of a MC estimator is simply proportional to $1/\sqrt{N}$ *independently of the number of dimensions*. This is one of the main advantages of MC methods to compute multidimensional integrals compared to deterministic integration methods.

In passing, we note that

$$\mathrm{var}[f(\mathbf{X})] = \int \big(f(\mathbf{x}) - Q\big)^2 p_{\mathbf{x}}(\mathbf{x})(\mathrm{d}\mathbf{x}) \,.$$

But since in practice we do not know the theoretical variance, we can define a MC estimator for it:

$$\hat{\sigma}_N^2 = \frac{1}{N-1} \sum_{n=1}^{N} \big(f(\mathbf{X}_n) - \hat{Q}_N\big)^2 \,.$$

(The $N-1$ in the denominator is necessary for $\hat{\sigma}_N^2$ to be an unbiased estimator, i.e., so that $\mathbb{E}[\hat{\sigma}_N^2] = \mathrm{var}[f(\mathbf{x})]$.) Note also that an efficient way to compute $\hat{\sigma}_n^2$ is to use the recursion relation

$$(n-1)\,\hat{\sigma}_n^2 = (n-2)\,\hat{\sigma}_{n-1}^2 + \left(1 - \frac{1}{n}\right)\big(f(\mathbf{X}_n) - \hat{Q}_{n-1}\big)^2 \,.$$

Using this formula, one can compute both the sample mean and variance in a single iteration.

As a simple example of an application of MC methods, one can approximate the value of π as follows. The area of the portion of the unit disk in the first quadrant is $\pi/4$. We can write this area as an integral of the form (3), where $\mathbf{x} = (x_1, x_2)$ and $p_{\mathbf{x}}(\mathbf{x}) \equiv 1$ (i.e., x_1 and x_2 are independent uniform RVs in [0,1]), and with $y(\mathbf{x}) = \|\mathbf{x}\|^2 = x_1^2 + x_2^2$ and $R = \{y \in \mathbb{R} : 0 \le y \le 1\}$. We can then estimate this integral with MC methods by simply taking random samples and counting the fraction of samples that fall inside the disk.

2.2 The Problem of Rare Events

While the variance of a MC estimator provides an absolute measure of how widely distributed it is around its mean value, in most cases a relative measure of the relative accuracy of the MC estimator is more useful. Such a measure is provided by the *coefficient of variation* (cv) of a RV Z, which is defined as

$$\mathrm{cv}[Z] = \mathrm{stdev}[Z] \,/\mathbb{E}[Z]\,,$$

where as usual $\mathrm{stdev}[Z] = \sqrt{\mathrm{var}[Z]}$ is the standard deviation. More precisely, the cv gives the number of samples that are necessary on average to achieve a given accuracy.

To apply this concept in our case, suppose $Z = \hat{Q}_N$. Since $\mathrm{var}[\hat{Q}_N] = \mathrm{var}[f(X)]/N$, we have $\mathrm{cv}[\hat{Q}_N] = \mathrm{cv}_Q/\sqrt{N}$, where (with some abuse of notation) we denoted $\mathrm{cv}_Q = \mathrm{stdev}[f]/Q$. Therefore, if we want $\mathrm{cv}[\hat{Q}]$ to be below a target value cv_o, on average we will need $N > (\mathrm{cv}_Q/\mathrm{cv}_o)^2$. In particular, for an indicator function the above calculations yield $\mathrm{cv}[\hat{Q}] = \sqrt{(1-Q)/(NQ)}$.

We can now see the problem of rare event simulation in a more quantitative way: If $Q \ll 1$, the number of samples needed on average to obtain a given value of cv is $N \sim 1/(Q\mathrm{cv}_o^2)$. For example, if $Q \sim 10^{-6}$ and we want a cv of 0.1, we need $N = 10^8$ samples. In other words, the problem is that, if $Q \ll 1$, the events that make $I(y(\mathbf{x})) = 1$ have a very low probability of occurring, and therefore, a large number of samples is needed even to observe one such event, and an even larger number of samples is needed to obtain a reliable estimate.

As mentioned in Section 1, VRTs are a collection of methods aimed at overcoming (or at least alleviating) this problem. In the following, we will look in some detail at two of them, namely IS and the cross-entropy method.

3 IMPORTANCE SAMPLING

3.1 Importance-Sampled MC Estimators

As mentioned earlier, the idea behind IS is simple: We want to improve the efficiency of MC methods by pushing (biasing) the simulations to favor the rare events of interest so that they will occur more frequently than they would otherwise. Of course, we must do this in a proper way in order to still have an unbiased estimator (i.e., an estimator whose expectation value is still the quantity of interest).

We do so by introducing a modified density $p_*(\mathbf{x})$, called the *biasing distribution*, and by rewriting the integral in Eq. (4) that defines the quantity of interest as

$$Q = \int f(\mathbf{x}) L(\mathbf{x}) p_*(\mathbf{x}) \,(\mathrm{d}\mathbf{x})\,. \tag{7}$$

The ratio $L(\mathbf{x}) = p_{\mathbf{x}}(\mathbf{x})/p_*(\mathbf{x})$ is called the *importance function* or *likelihood ratio* (or even weight function in some works). An equivalent way to write the integral in Eq. (7) is

$$Q = \mathbb{E}_*[f(\mathbf{X})L(\mathbf{X})]\,, \tag{8}$$

where $\mathbb{E}_*[\cdot]$ denotes expectation values with respect to the density $p_*(\mathbf{x})$. We then define an importance-sampled MC estimator as

$$\hat{Q}_N^* = \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{X}_n^*)L(\mathbf{X}_n^*)\,,$$

where the samples $\mathbf{X}_n^*$ are now drawn from the biasing distribution $p_*(\mathbf{x})$. Importantly, note that a necessary requirement in order to carry out the change of measure from (4) to (7) is that the support of $f(\mathbf{x})p_*(\mathbf{x})$ includes that of $f(\mathbf{x})p(\mathbf{x})$ [i.e., $f(\mathbf{x})p_*(\mathbf{x}) \neq 0$ whenever $f(\mathbf{x})p(\mathbf{x}) \neq 0$]. Otherwise $\hat{Q}_N^*$ will not converge to the correct value in general. Conversely, it should be obvious that, as long as this condition is satisfied, $\mathbb{E}_*[\hat{Q}_N^*] = Q$, thanks to Eq. (8). That is, we still have an unbiased estimator for the quantity of interest.

The reason for biasing the sampling distribution can be seen by looking at the variance of our new estimator, namely

$$\mathrm{var}_*[f(\mathbf{X})L(\mathbf{X})] = \int \big(f(\mathbf{x})L(\mathbf{x}) - Q\big)^2 p_*(\mathbf{x})(\mathrm{d}\mathbf{x}) = \int f^2(\mathbf{x})L(\mathbf{x})p_{\mathbf{x}}(\mathbf{x})(\mathrm{d}\mathbf{x}) - Q^2. \tag{9}$$

Thus,

$$\mathrm{var}[f(\mathbf{X})] - \mathrm{var}_*[f(\mathbf{X})L(\mathbf{X})] = \int f^2(\mathbf{x})\big(1 - L(\mathbf{x})\big)\, p_{\mathbf{x}}(\mathbf{x})(\mathrm{d}\mathbf{x})\,. \tag{10}$$

Looking at the integrand in Eq. (9) we see that if $p_*(\mathbf{x}) = f(\mathbf{x})p_{\mathbf{x}}(\mathbf{x})/Q$, we would have $\mathrm{var}_*[f(\mathbf{X})L(\mathbf{X})] = 0$. Thus, in this case our importance-sampled estimator would have zero variance: every sample would always yield the same result, namely the exact value of the quantity Q of interest!

Of course, the above choice of biasing distribution is not practical, because it requires the advance knowledge of the value of Q (which is the desired result). On the other hand, Eq. (10) implies if we can choose $p_*(\mathbf{x})$ so that $p_*(\mathbf{x}) > p_{\mathbf{x}}(\mathbf{x})$ wherever $f^2(\mathbf{x})p_{\mathbf{x}}(\mathbf{x})$ is large and $p_*(\mathbf{x}) < p_{\mathbf{x}}(\mathbf{x})$ wherever $f^2(\mathbf{x})p_{\mathbf{x}}(\mathbf{x})$ is small, the variance of our importance-sampled estimator will be much smaller than the original variance. This corresponds to redistributing the probability mass in accordance with its relative importance as measured by the weight $f^2(\mathbf{x})p_{\mathbf{x}}(\mathbf{x})$. The zero-variance choice is just an ultimate case of this redistribution.

An estimator of the importance-sampled variance can be written using the same methods as before:

$$\hat{\sigma}_N^{*2} = \frac{1}{N-1}\sum_{n=1}^{N}\big(f(\mathbf{X}_n^*)L(\mathbf{X}_n^*) - \hat{Q}_N^*\big)^2\,,$$

which again can be computed recursively as

$$(n-1)\,\hat{\sigma}_n^{*2} = (n-2)\,\hat{\sigma}_{n-1}^{*2} + (1-1/n)\big(f(\mathbf{X}_n^*)L(\mathbf{X}_n^*) - \hat{Q}_{n-1}^*\big)^2 .$$

A case in which the likelihood ratio can be computed particularly easily is the common situation in which the components of both $\mathbf{X}$ and $\mathbf{X}^*$ are statistically independent, for in this case it is $p_{\mathbf{x}}(\mathbf{x}) = \prod_{j=1}^{D} p_{x_j}(x_j)$ and similarly for $p_*(\mathbf{x})$, yielding the likelihood ratio simply as $L(\mathbf{x}) = \prod_{j=1}^{D} p_{x_j}(x_j)/p_{*j}(x_j)$.

Of course, a key question is how to make the choice of a biasing distribution in practice. We will see shortly how IS works in a simple example, but unfortunately there are no general rules that work in all cases, and the task of choosing good biasing distributions is the most difficult step in applying IS. Also note that choosing a bad biasing distribution can make the problem worse and make the variance of the importance-sampled estimator much bigger than the original one. This is why it is occasionally said that IS (like all of computer simulation; Knuth, 2011) is an art. Nonetheless, *there are* general principles that one can follow to select a biasing distribution, and indeed IS has been used with success in a large variety of problems.

3.2 A Simple Example

As an illustration of the concepts discussed above, it will be useful to consider a specific example: a 1D symmetric random walk (RW). Let $\mathbf{X} = (X_1, \ldots, X_D)$ and

$$y(\mathbf{X}) = \sum_{j=1}^{D} X_j ,$$

where, for $j = 1, \ldots, D$,

$$X_j = \begin{cases} +1 & \text{with probability } 1/2 , \\ -1 & \text{with probability } 1/2 . \end{cases}$$

That is, we consider a sequence of D random steps, each one unit to the right or to the left with probability 1/2, and we are interested in computing the final position. In particular, suppose we want to compute the probability that the final position will be to the right of some given threshold:

$$Q = \mathbb{P}[\, y(\mathbf{X}) \geq C \,] .$$

To make things more concrete, suppose $D = 100$ and $C = 70$. This is equivalent to asking the probability that by flipping a coin 100 times we get at least 85 heads.

We can try to estimate the pdf of the final position (and therefore our desired probability) by performing MC simulations. That is, we use Eq. (5) with

$$f(\mathbf{X}) = H(y(\mathbf{X}) - C) ,$$

where $H(\cdot)$ is the Heaviside step function: $H(s) = 1$ for $s > 0$ and $H(s) = 0$ for $s < 0$. The histogram of the final position in a simulation with $N = 100,000$ samples is shown in Fig. 1. The problem is that no samples occurred with final position greater than 50. That, of course, is because our desired event is extremely rare, and therefore, we are very unlikely to see it with a reasonable number of samples.

To obviate this problem, we can simulate a biased RW: given $0 < q < 1$, for $j = 1, \ldots, D$ we take

$$X_j = \begin{cases} +1 & \text{with probability } q\,, \\ -1 & \text{with probability } 1 - q\,. \end{cases}$$

The value $q = 1/2$ reproduces the unbiased case. If $q > 1/2$, however, steps to the right will be more prevalent, which means that we are pushing the final position to the right (which is what we want). The histogram of the final position in a biased simulation with $q = 0.7$ is shown in Fig. 2. The results show we now get a lot more samples with final positions to the right. But of course now we cannot simply take the relative frequency of our event as an estimator of the desired probability, and we need to use the likelihood ratios instead. The individual likelihood ratio for a single step is given by

$$\ell(X_j) = \begin{cases} 1/(2q) & \text{if } X_j = 1\,, \\ 1/[2(1-q)] & \text{if } X_j = -1\,, \end{cases}$$

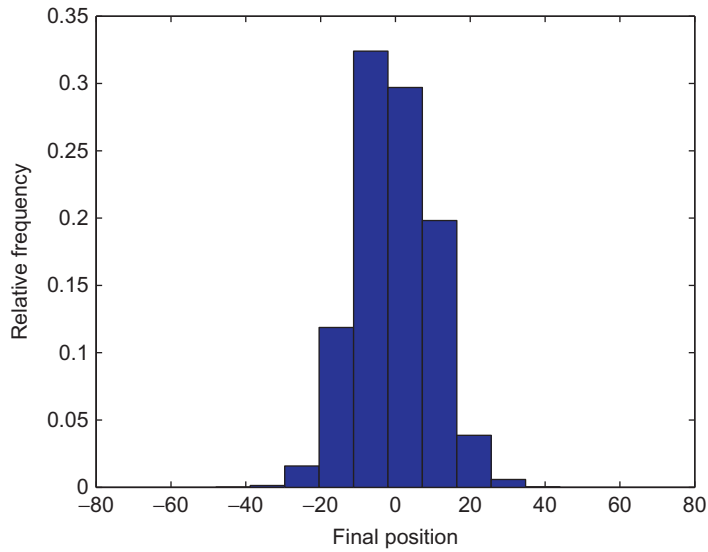


FIGURE 1 Histogram of the final position in an unbiased MC simulation of a symmetric 1D random walk with $N = 100,000$ samples.

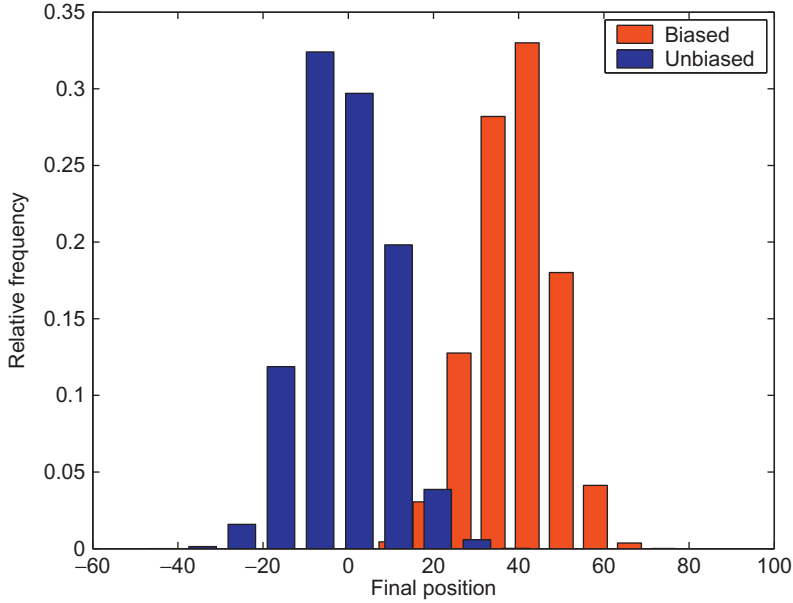


FIGURE 2 Histogram of the final position (in red (light gray in the print version)) in a biased MC random walk with $q = 0.7$, with $N = 100,000$ samples. For comparison, the blue (dark gray in the print version) histogram shows the unbiased case.

and the overall likelihood ratio for a single sample RW is $L(\mathbf{X}) = \prod_{j=1}^{D} \ell(X_j)$. Now recall that if $q > 1/2$, on average there will be more samples for which the final position is to the right. Since $1/(2q) < 1$, we can therefore expect the overall likelihood ratio for those samples to be less than 1 as well. In fact, we will see that the likelihood ratios can get quite small.

The results in Fig. 2 should already demonstrate that use of an appropriate biasing distribution can yield a distinct advantage over standard MC simulations. Roughly speaking, the reason why IS is effective is that, generically, it is much better to estimate a quantity by using a large number of samples each of which contributes a little to the final result (by virtue of the likelihood ratios), rather than using a very small number of result which gives a binary contribution (one or zero). (This is true as long as the contributions are not *too* small, as we will discuss later.) The results, however, also suggest that perhaps one could get an even better result by further increasing the value of q. The natural question is then: What is the optimal value of q? This question raises again the key issue in properly applying IS: How does one choose a good biasing distribution? Usually, this requires some analytical knowledge about the behavior of the system. We turn to this issue next.

3.3 The Optimal Biasing Distribution

The case of a symmetric 1D RW is simple enough that analytical expressions can be derived for the pdf of the final position. Comparison with these analytical results will then provide some insight into the issue of how to best choose a biasing distribution.

It is easy to see that if D is odd and m is even or vice versa, it is $\mathbb{P}[\, y(\mathbf{X}) = m\,] = 0$. If D and m are both even or both odd, instead,

$$\mathbb{P}[\, y(\mathbf{X}) = m\,] = \frac{1}{2^D}\binom{D}{(D+m)/2}.$$

The factor $1/2^D$ arises because we are taking D steps, each of which has a probability 1/2 of being either to the left or to the right. The binomial coefficient arises because, for the final position to equal m, we need a total of $(D+m)/2$ steps to the right and $(D-m)/2$ steps to the left, and there are exactly D choose $(D+m)/2$ ways to arrange this. Taking the sum over all possible results above the threshold, we then simply have

$$Q = \frac{1}{2^D}\sum_{m=C}^{D}{}' \binom{D}{(D+m)/2}, \tag{11}$$

where the prime indicates that the sum should be taken only on even values of m or only on odd values of m depending on whether D is even or odd.

In particular, for $D = 100$ and $C = 70$, we then get $\mathbb{P}[y(\mathbf{X}) \geq C] = 2.4 \times 10^{-13}$. Recalling the discussion about the cv in Section 2.2, we then see that, even in a simple example as this, it would be almost hopeless to try to accurately estimate the desired probability numerically, except perhaps on the fastest supercomputers.

The above discussion, however, does not answer our question of what is the optimal choice of biasing. To answer that question, we need to dig a little deeper. Fortunately, our example is simple enough that we can actually calculate analytically the variance of the biased estimator. Note first that

$$\begin{aligned}\mathrm{var}_*[f(\mathbf{X})L(\mathbf{X})] &= \mathbb{E}_*[f^2(\mathbf{X})L^2(\mathbf{X})] - \left(\mathbb{E}_*[f(\mathbf{X})L(\mathbf{X})]\right)^2 \\ &= \mathbb{E}[f(\mathbf{X})L(\mathbf{X})] - \left(\mathbb{E}[f(\mathbf{X})]\right)^2,\end{aligned}$$

where we used that $f(\cdot)$ is an indicator and we rewrote expectations with respect to $p_*(\mathbf{x})$ as expectations with respect to $p_{\mathbf{x}}(\mathbf{x})$. We then have

$$\mathrm{var}_*[f(\mathbf{X})L(\mathbf{X})] = -Q^2 + \frac{1}{2^D}\sum_{m=C}^{D}{}' \binom{D}{(D+m)/2} \frac{1}{(2q)^{(D+m)/2}\,[2(1-q)]^{(D-m)/2}},$$

where Q is given by Eq. (11). Note that the last part of the sum is precisely the likelihood ratio of a sample with final position m.

We can now look at this variance as a function of q. Even better, we can plot the biased cv; i.e., the ratio $\text{cv}_* = \text{stdev}_*[fL]/Q$. The corresponding results are given in Fig. 3. (Note that these results agree very well with numerical estimates of the variance as a function of q.)

Figure 4 shows that the likelihood ratios when $q > 1/2$ are indeed much smaller than unity, as anticipated. Thus, each sample that ends past the threshold will only give a small contribution to the estimator. (It should be noted that, in our example, the value of the likelihood ratio is the same for all paths that lead to the same value of final position, but this is not true in more general situations.)

From Fig. 3, we see that optimal value of q is 0.85. At that value, the cv is just 2.32, whereas for the unbiased RW ($q = 0.5$) it is 2.04×10^6. Now recall that the number of MC samples needed on average to get a given value of the cv is $N = (\text{cv}_j/\text{cv}_o)^2$ or, for importance-sampled MC, $N = (\text{cv}_*/\text{cv}_o)^2$. Using the optimal value $q = 0.85$, one can therefore obtain a cv of 0.1 using just a few hundred samples. On the other hand, to obtain the same level of accuracy with unbiased MC simulations, one would need over 10^{14} samples. So, in our example IS increases the efficiency of the MC simulations by 10 orders of magnitude! Such a huge increase in efficiency is not a fluke, but has been realized in practical applications (e.g., see Biondini et al., 2004; Marzec et al., 2013; Moore et al., 2008).

The general message that we should take from this example is that the optimal biasing choice is to concentrate the MC samples around the *most likely*

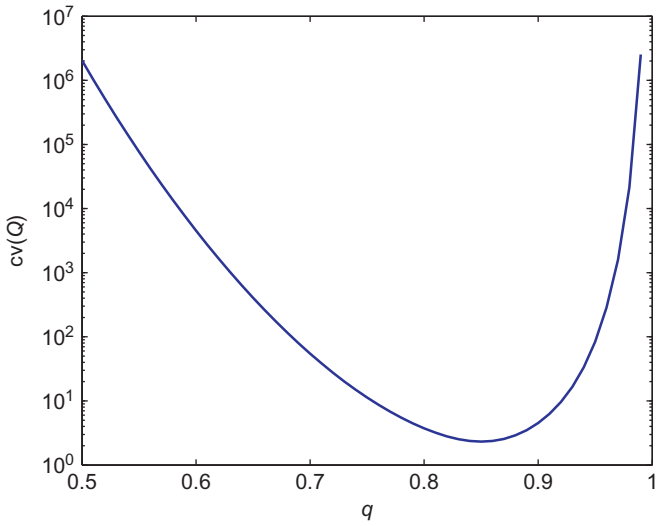


FIGURE 3 The ratio $\text{cv}_* = \text{stdev}_*[Q]/Q$ as a function of q for the 1D random walk.

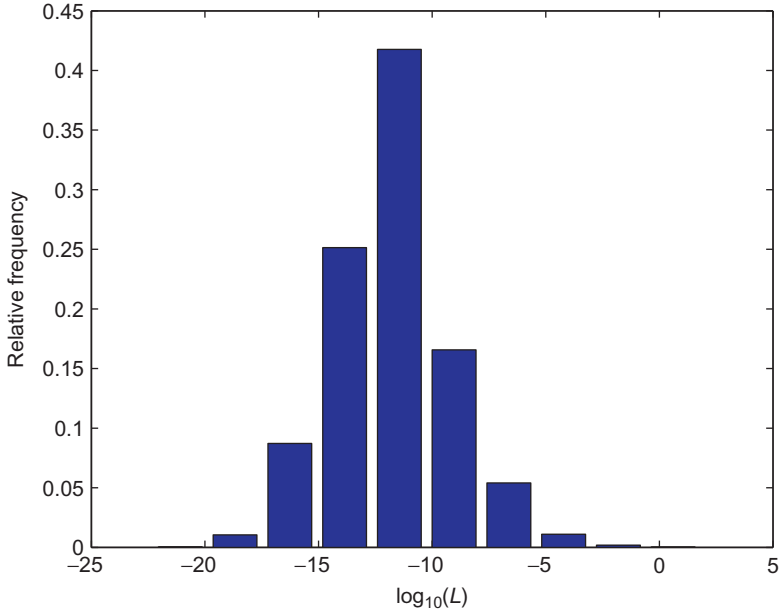


FIGURE 4 Relative frequency of the likelihood ratios in a simulation with $q = 0.85$ and $N = 100,000$.

path that leads to the desired event. The reason why this is so is that the event with the largest value of $p(\mathbf{x})$ among all those for which $I(y(\mathbf{x})) \neq 0$ is the one that provides the dominant contribution to the integral that defines Q. (Note that in many cases of interest, $p(\mathbf{x})$ decays exponentially away from its maximum.) In our example, the most likely way to obtain *at least* 85 heads (the desired event) is to obtain *exactly* 85 heads. So, the optimal biasing choice is to bias the simulations around this value.

3.4 Common Biasing Choices and Their Drawbacks

We next discuss two simple and commonly mentioned approaches to selecting a biasing distribution: variance scaling and mean translation.

With *variance scaling*, one raises the variance of the input RVs in order to increase the chance that some samples will hit the desired event. For example, if the input RVs $\mathbf{X} \in \mathbb{R}^D$ are normal, i.e., $p(\mathbf{x}) = p_\sigma(\mathbf{x})$, with $p_\sigma(\mathbf{x}) = \mathrm{e}^{-\mathbf{x}\cdot\mathbf{x}/2\sigma^2}/(\sqrt{2\pi}\sigma)^D$, one may try to choose $p_*(\mathbf{x}) = p_{\sigma_*}(\mathbf{x})$, with $\sigma_* > \sigma$. In simple situations (such as the 1D case, i.e., $D = 1$), variance scaling can be quite effective. The applicability of the method is rather limited, however, because of its well-known dimensionality problem. Generally speaking, the problem is that, in many situations, the area over which the samples can "spread" grows faster

than that of the region of interest with the number of dimensions increases. Therefore, while it may intuitively seem that increasing the variance would increase the probability of reaching the desired region compared to the unbiased distribution, this probability will in fact decrease. The end result is that, in dimensions larger than one, the best variance is typically the unscaled one—i.e., the unbiased distribution—and all other biasing choices yield worse results than unbiased MC simulations (i.e., the variance of the importance-sampled estimator is larger than that of the standard MC estimator). For this reason, variance scaling has largely been superseded by the mean translation method.

With *mean translation*, one adds a mean to the input RVs in order to increase the chance that some samples will hit the desired event, e.g., with normal RVs, one would choose $p_*(\mathbf{x}) = p_\sigma(\mathbf{x} - \mathbf{m})$, with the vector $\mathbf{m}$ being the mean shift. If $\mathbf{m}$ is chosen correctly, mean translation can be very effective in many situations. This method also has some drawbacks, however. When the dimensionality of the problem is large and/or the indicator function of the desired event has a nontrivial geometry in sample space, the optimal translation point might be impossible to find analytically. In this case, one must resort to hybrid or adaptive methods. Also, problems can arise when the symmetry of the problem leads to degeneracy, e.g., suppose one is interested in the total norm of the sum of the RVs. In this case, there is no single choice of translation point that can lead to the correct result. (In the parlance of large deviations theory, which will be briefly discussed in Section 8, this is an example of a situation in which there are multiple—in this case an infinity of—minimum rate points, and no single dominating point; e.g., see Bucklew, 2004 for a discussion of this issue).

We will return to the problem of selecting a good biasing point in Sections 5 and 8.

4 MULTIPLE IS

In some cases of interest, no single choice of biasing distribution can efficiently capture all the regions of sample space that give rise to the events of interest. In these cases, it is necessary to use IS with more than one biasing distribution. The simultaneous use of different biasing methods (which is similar to the use of a mixture density) is called *multiple importance sampling*.

4.1 Multiple IS: General Formulation

Suppose we want to use J biasing distributions $p_{*1}(\mathbf{x}), \ldots, p_{*J}(\mathbf{x})$, each of which allows us to efficiently reach a given region of sample space. The issue arises of how to correctly weight the results coming from these different distributions. One possible solution to this problem is to assign a weight $w_j(\mathbf{x})$ to each distribution and rewrite Q as:

$$Q = \sum_{j=1}^{J} Q_j = \sum_{j=1}^{J} \int w_j(\mathbf{x}) f(\mathbf{x}) L_j(\mathbf{x}) p_j^*(\mathbf{x})(\mathrm{d}\mathbf{x}) \,, \tag{12}$$

where $L_j(\mathbf{x}) = p(\mathbf{x})/p_{*j}(\mathbf{x})$ is the likelihood ratio for the jth distribution. Note that the right-hand side of Eq. (12) equals Q for any choice of weights such that $\sum_{j=1}^{J} w_j(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathbb{R}^D$. Each choice of weights corresponds to a different way of partitioning of the total probability.

From (12), a multiply-importance-sampled MC estimator for Q can now be written as

$$\hat{Q} = \sum_{j=1}^{J} \hat{Q}_j = \sum_{j=1}^{J} \frac{1}{N_j} \sum_{n=1}^{N_j} w_j(\mathbf{X}_{j,n}) f(\mathbf{X}_{j,n}) L_j(\mathbf{X}_{j,n}) \,, \tag{13}$$

where N_j is the number of samples drawn from the jth distribution $p_{*j}(\mathbf{x})$, and $\mathbf{X}_{j,n}$ is the nth such sample. Also, one can show that, similar to before, an unbiased estimator of its variance is

$$\hat{\sigma}^2_{\hat{Q}} = \sum_{j=1}^{J} \frac{1}{N_j(N_j-1)} \sum_{n=1}^{N_j} \left(w_j(\mathbf{X}_{j,n}) L_j(\mathbf{X}_{j,n}) f(\mathbf{X}_{j,n}) - \hat{Q}_j \right)^2 .$$

As before, recursion relations can also be written so that this quantity can be obtained without the need of storing all the individual samples until the end of the simulation:

$$\hat{\sigma}^2_{\hat{Q}} = \sum_{j=1}^{J} \frac{1}{N_j(N_j-1)} \hat{S}_{j,N_j} \,,$$

with $\hat{Q} = \sum_{j=1}^{J} \hat{Q}_{j,N_j}$ and [in the special case $f(\mathbf{x}) = I(y(\mathbf{x}))$]

$$\hat{Q}_{j,n} = \frac{n-1}{n} \hat{Q}_{j,n-1} + \frac{1}{n} w_j^2(\mathbf{X}_{j,n}) L_j^2(\mathbf{X}_{j,n}) I(y(\mathbf{X}_{j,n})) \,,$$
$$\hat{S}_{j,n} = \hat{S}_{j,n-1} + \frac{n-1}{n} \left(w_j^2(\mathbf{X}_{j,n}) L_j^2(\mathbf{X}_{j,n}) I(y(\mathbf{X}_{j,n})) - \hat{Q}_{j,n-1} \right)^2 .$$

4.2 The Balance Heuristics

Of course, several ways exist to choose the weights $w_j(\mathbf{x})$ when using multiple IS. And the choice of weights is almost as important as the choice of biasing distributions $p_j(\mathbf{x})$. Different weighting functions result in different values for the variance of the combined estimator. A poor choice of weights can result in a large variance, thus partially negating the gains obtained by IS. The best weighting strategies are of course the ones that yield the smallest variance.

The simplest possibility is just to set $w_j(\mathbf{x}) = 1/J$ for all $\mathbf{x}$, meaning that each distribution is assigned an equal weight in all regions of sample space. This choice is not advantageous, however, as we will see shortly. Another simple

choice is that in which the weighting functions are constant over the whole sample space. In this case, one would have

$$Q = \sum_{j=1}^{J} w_j \int I(y(\mathbf{x}))L_j(\mathbf{x})(\mathrm{d}\mathbf{x}) = \sum_{j=1}^{J} w_j \, \mathbb{E}_{*j}[I(y(\mathbf{x}))L_j(\mathbf{x})] \, .$$

The corresponding importance-sampled estimator is then simply a weighted combination of the estimators obtained by using each of the biasing distributions. Unfortunately, the variance of Q is also a weighted sum of the individual variances: $\sigma_j^2 = \sum_{j=1}^{J} w_j \sigma_j^2$, and if any of the sampling techniques is bad in a given region, then Q will also have a high variance. Then, one may be tempted to define the weights according to the actual number of samples from each distribution that fall in a given region. It is important to realize, however, that this is not a good choice, as it does not produce a unbiased estimator (i.e., one whose expectation value is the desired quantity).

On the other hand, there is a relatively simple and particularly useful choice of weights: the *balance heuristics* (Owen and Zhou, 2000; Veach, 1997). In this case, the weights $w_j(\mathbf{x})$ are assigned according to

$$w_j(\mathbf{x}) = \frac{N_j p_{*j}(\mathbf{x})}{\sum_{j'=1}^{J} N_{j'} p_{j'}^{*}(\mathbf{x})} \, . \tag{14}$$

Note that the quantity $N_j p_{*j}(\mathbf{x})$ is proportional to the *expected* number of hits from the jth distribution. Thus, the weight associated with a sample $\mathbf{x}$ with the balance heuristics is given by the relative likelihood of realizing that sample with the jth distribution relative to the total likelihood of realizing that same sample with all distributions. Hence, Eq. (14) weights each $p_{*j}(\mathbf{x})$ most heavily in those regions of sample space where $p_{*j}(\mathbf{x})$ is largest. [Note Eq. (14) can also be written in terms of likelihood ratios, a form which is particularly convenient in Eq. (13).]

The balance heuristics has been shown to be close to optimal in most situations (Veach, 1997). Of course, other strategies are possible, and some of these alternatives do perform better in specific cases (Veach, 1997). It is difficult to tell *a priori* which choice will be best in any given situation, however. Therefore, the balance heuristics is frequently used in practice because of its effectiveness and simplicity.

4.3 Application: Numerical Estimation of Probability Density Functions

In some cases, one is not just interested in one specific probability, but rather would like to numerically estimate the whole pdf of a quantity of interest which is a complicated function of the RVs. As an application of multiple IS, here we briefly discuss the strategy that can be used to set up the numerical simulations.

In our example of a 1D RW, suppose that we want to numerically estimate the pdf of the final position $y(\mathbf{X})$. (Of course, in this case we already did it analytically, but the example will serve to illustrate the procedure.)

The desired result now is more than a single number; instead, we are trying to simultaneously estimate all the integrals

$$p_k = \frac{1}{\Delta y_k} \int_{R_k} p_y(y)\, \mathrm{d}y = \frac{1}{\Delta y_k} \int I_{R_k}(y(\mathbf{x})) p_{\mathbf{x}}(\mathbf{x})\, (\mathrm{d}\mathbf{x})\,, \tag{15}$$

for $k = 1, \ldots, K$, where $y_k = y_o + \Delta y_{k-1}$, with $\Delta y_k = y_{k+1} - y_k$, and $R_k = [y_k, y_{k+1}]$. Note that the integrals in Eq. (15) are of the same type as that in Eq. (3). Thus, we can apply the IS techniques presented earlier. It should be clear, however, that no single biasing distribution can efficiently generate the whole range of possible values of y, and therefore, one needs to resort to multiple IS. The procedure is then to:

1. choose a set of J biasing distributions $p_{*1}(\mathbf{x}), \ldots, p_{*J}(\mathbf{x})$;
2. perform a predetermined number N_j of MC simulations for each distribution, keeping track of the likelihood ratio and the weights for each sample;
3. sort the results of all the MC samples into bins and combine the individual samples using one of the weighting strategies presented earlier.

Note that it is not necessary to fix the number of bins and the precise bin locations in advance of the simulations, and one can choose them *a posteriori* to optimize the results.

Figure 5 shows the results obtained from each of three individual importance-sampled MC simulations of the same 1D RW described earlier, together with the corresponding coefficient of variation. Note that, as is often the case in similar situations, one of the biasing distributions was chosen to be the unbiased one, to make sure that the simulations recover the main portion of the desired pdf. As expected, different values of the biasing parameter target different regions of the pdf. (Negative values of final position can obviously be targeted just as easily by choosing $q < 1/2$.) Note how the cvs for each simulations become large near the edges of the region targeted by each simulation, where the expected number of samples is small.

Figure 6 shows the corresponding pdf obtained when the results from the individual simulations are combined into a single multiply-importance-sampled estimator using the balance heuristics. One can see that indeed the combined results have a low cv throughout the range of values desired.

5 THE CROSS-ENTROPY METHOD

As we have seen earlier, in order for IS methods to be effective, it is crucial to choose a good biasing strategy, as poor biasing strategies can lead to incorrect results and/or performance that is even poorer than that of standard MC. In some

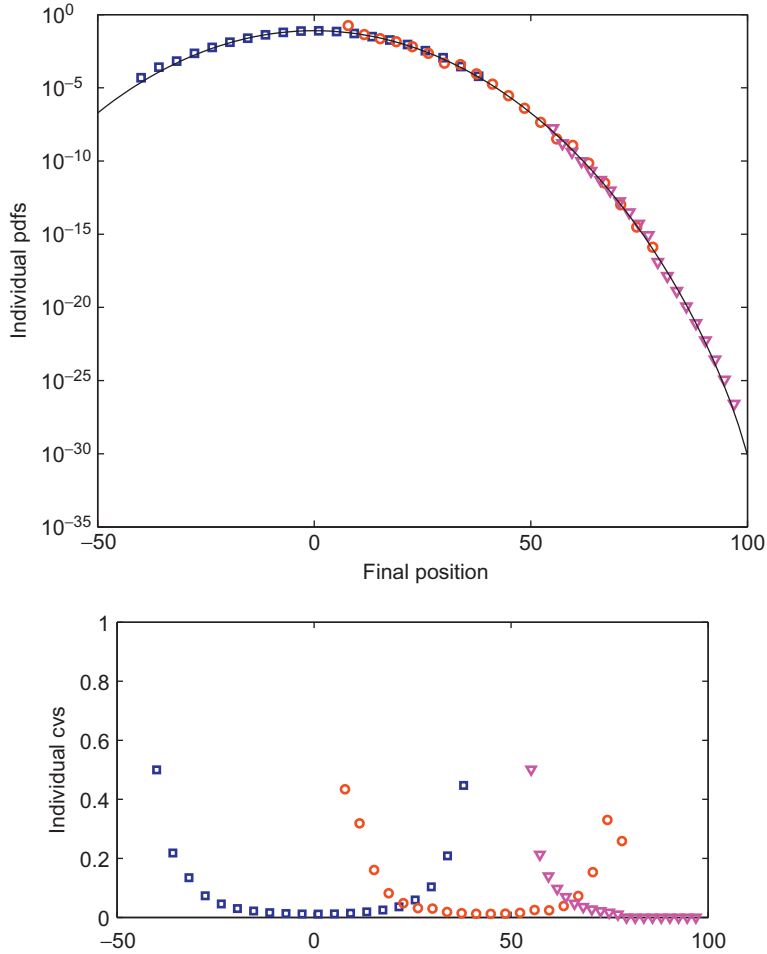


FIGURE 5 The portions of pdf of the 1D random walk as reconstructed from three IS-MC runs with $N = 10,000$ each. Blue (dark gray in the print version): $q = 0.5$; red (gray in the print version): $q = 0.72$; magenta (light gray in the print version): $q = 0.9$. Inset: The cv for each of the simulations.

cases, however, it may be difficult to find such a strategy. A possible alternative in such cases is the use of the *cross-entropy method* (de Boer et al., 2005; Rubinstein and Kroese, 2004).

Recall that the theoretical optimal biasing distribution, $p_{\text{opt}} = I_R(y(\mathbf{x}))\, p(\mathbf{x})/Q$, is not practical, as it requires knowledge of Q in advance. Often, however, one can find a good biasing distribution by requiring it to be "close" to the optimal biasing distribution. This can be accomplished by minimizing the Kullback–Leibler distance (Kullback and Leibler, 1951):

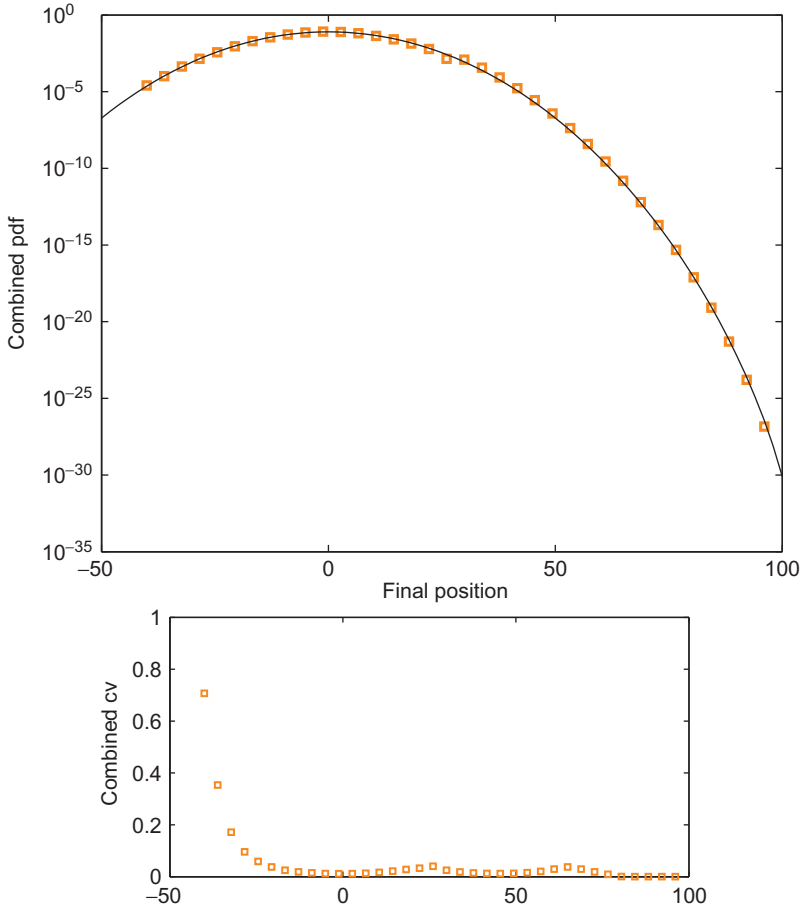


FIGURE 6 The pdf of the 1D random walk as reconstructed by combining the three simulations into a single multiply-importance-sampled run. Inset: The overall coefficient of variation.

$$\begin{aligned}\mathcal{D}(p_{\text{opt}}, p_*) &= \mathbb{E}_{p_{\text{opt}}}\left[\ln \frac{p_{\text{opt}}(\mathbf{x})}{p_*(\mathbf{x})}\right] \\ &= \int \ln(p_{\text{opt}}(\mathbf{x}))\, p_{\text{opt}}(\mathbf{x})(\mathrm{d}\mathbf{x}) - \int \ln(p_*(\mathbf{x}))\, p_{\text{opt}}(\mathbf{x})(\mathrm{d}\mathbf{x})\,, \end{aligned} \tag{16}$$

which is also known as the cross-entropy between two probability distributions. Minimizing $\mathcal{D}(p_{\text{opt}}, p_*)$ is equivalent to maximizing $\int \ln(p_*(\mathbf{x}))p_{\text{opt}}(\mathbf{x})\,(\mathrm{d}\mathbf{x})$. (Note that $\mathcal{D}$ is not a true "distance," as it is not symmetric in its two arguments.) In turn, recalling the expression for p_{opt}, this problem is equivalent to maximizing $\mathbb{E}[I_R(y(\mathbf{x}))\ \ln p_*(\mathbf{x})\,]$.

Suppose that, as is typically the case in practice, the biasing distributions are selected from a family $\{p_*(\mathbf{x};\mathbf{v})\}_{\mathbf{v}\in V}$ parametrized by a vector $\mathbf{v}$, where V is the corresponding parameter space, and suppose $p_*(\mathbf{x};\mathbf{u}) = p(\mathbf{x})$ is the unbiased distribution. Based on the above discussion, one must maximize the integral

$$D(\mathbf{v}) = \int I_R(y(\mathbf{x}))\ln(p_*(\mathbf{x};\mathbf{v}))p(\mathbf{x})\,(\mathrm{d}\mathbf{x})\,. \tag{17}$$

This is usually done numerically. Since the optimal biasing distribution is typically far from the unbiased distribution, however, the region R of interest is generally also far from the region in sample space where the unbiased distribution $p(\mathbf{x})$ is large. Thus, determining the best choice for $\mathbf{v}$ also becomes a rare event simulation.

The solution to this problem is to use a sequence of intermediate regions R_j that reach the desired region iteratively. (For an alternative approach, see Chan and Kroese, 2012.) Let $D_j(\mathbf{v})$ be the integral in Eq. (17) with R replaced by R_j. Starting with the unbiased distribution, one uses MC sampling to minimize the CE distance between the parametrized distribution and the optimal distribution that reaches R_1. This step, which is done by finding the maximum of $D_1(\mathbf{v})$ over this first set of samples, will give a parameter value $\mathbf{w}_2$. One then uses this value to define a biasing distribution and performs an MC simulation with this distribution to minimize the CE distance between the parametrized distribution and the optimal distribution that reaches R_2. Since a biasing distribution is being used, each step of the procedure is an IS simulation of a stochastic optimization. That is, at step j, one must compute

$$\mathbf{w}_{j+1} = \max_{\mathbf{v}\in V} \hat{D}_j(\mathbf{v})\,, \tag{18}$$

where

$$\hat{D}_j(\mathbf{v}) = \frac{1}{M}\sum_{m=1}^{M} I_{R_j}(y(\mathbf{x}^{(m)}))\ \ln(p_*(\mathbf{x}^{(m)};\mathbf{v}))\,L(\mathbf{x}^{(m)})\,, \tag{19}$$

and where $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(M)}$ are i.i.d. samples generated according to $p_*(\mathbf{x};\mathbf{w}_j)$. The optimal biasing distribution can then be adaptively determined by performing the following steps:

1. Set $j = 0$ and the initial parameter $\mathbf{w}_0 = \mathbf{u}$;
2. Generate MC samples according to $p_*(\mathbf{x};\mathbf{w}_j)$;
3. Solve Eq. (18) to find $\mathbf{w}_{j+1}$;
4. If the iteration has converged, stop; otherwise, increase j to $j+1$ and reiterate from step 2.

Once the iteration has converged, one can then perform IS-MC simulations using the biasing distribution $p_*(\mathbf{x};\mathbf{w}_{\text{final}})$.

The regions R_j can be defined in terms of sample quantiles of some quantity of interest (de Boer et al., 2005). A major issue associated with the above

algorithm, however, is how to accomplish step 3. Solving (18) is in general complicated. If $D(\mathbf{v})$ is convex and differentiable, however, the solutions of (18) can be obtained by solving a system of algebraic equations:

$$\frac{1}{M}\sum_{m=1}^{M} I_R(y(\mathbf{x}^{(m)}))\,\nabla_{\mathbf{v}}[\ln p(\mathbf{x}^{(m)};\mathbf{v})]\,L(\mathbf{x}^{(m)};\mathbf{u},\mathbf{w}) = 0\,. \tag{20}$$

In many applications, this equation can be solved analytically. If that is not possible, one can try to find a solution numerically.

The CE method enjoys desirable convergence properties. Specifically, for certain (static) models, under mild regularity conditions the CE method terminates with probability 1 in a finite number of iterations. Moreover, the CE method provides a consistent and asymptotically normal estimator for the optimal reference parameters (see Homem-de-Mello and Rubinstein, 2002). The CE method has been successfully applied to the estimation of rare event probabilities in dynamic models, in particular queueing models involving both light and heavy tail input distributions (de Boer et al., 2004; Kroese and Rubinstein, 2004). Recently, a method that combines IS with the CE method has been developed and used with success to study a specific model of birefringence-induced errors (Marzec et al., 2013; Schuster et al., 2014), and noise-induced perturbations (Donovan and Kath, 2011) of lightwave communication systems. We refer the reader to de Boer et al. (2005) and Rubinstein and Kroese (2004) for further details about the method and its applications.

6 MCMC: REJECTION SAMPLING, THE METROPOLIS METHOD, AND GIBBS SAMPLING

A related simulation problem is that in which the distribution $p_{\mathbf{X}}(\mathbf{x})$ of the RVs $\mathbf{X}$ is not easy to sample from. This might happen for various reasons, e.g., a typical situation is that in which the normalization constant in the distribution is difficult to compute. Another typical situation is that in which the RVs are not independent but are related by complicated nonlinear interdependencies, in which case $p_{\mathbf{X}}(\mathbf{x})$ is a derived density that may be very hard to compute. In these situations, a useful approach could be the use of rejection sampling, the Metropolis–Hastings method (Metropolis et al., 1953), and its variants such as Gibbs sampling. We next give a brief introduction to these methods, referring the reader to Fishman (1996) and MacKay (2003) for further details.

We start with the simplest among these methods: *rejection sampling*. Consider for simplicity a 1D case, namely a single RV X distributed according to $p_X(x)$. Suppose that $p_X(x) = \tilde{p}_X(x)/Z$, where $\tilde{p}_X(x)$ is known but Z is not. The idea behind rejection sampling is to use a proposal density $p_*(x) = \tilde{p}_*(x)/\tilde{Z}$ which is known (possibly up to the normalization constant $\tilde{Z}$) and from which we can easily draw samples. Suppose further that we can also find a constant C such

that $C\tilde{p}_*(x) > \tilde{p}_x(x)$ for all x. A single step of the rejection sampling method proceeds as follows:

(i) Generate a RV, X_*, from the proposal density $\tilde{p}_*(x)$.
(ii) Evaluate $C\tilde{p}_*(X_*)$ and generate a uniformly distributed RV u from the interval $[0, C\tilde{p}_*(X_*)]$.
(iii) Evaluate $p_x(X_*)$ and accept or reject the sample X_* by comparing the value of u with the value of $\tilde{p}_x(X_*)$. More precisely, if $u > \tilde{p}_x(X_*)$, then X_* is rejected; otherwise, it is accepted, in which case X_* is added to our set of samples. (The value of u is discarded no matter what.)

The obvious question is why should this procedure generate samples from $p_x(x)$. To answer this, note first that the pair (X_*, u) identifies a point in the two-dimensional xy plane. Moreover, (X_*, u) is selected with uniform probability from the area underneath the curve $y = C\tilde{p}_*(x)$. The above algorithm rejects all points that lie above the curve $y = \tilde{p}_x(x)$. Thus, points (x, u) that are accepted are uniformly distributed over the area under $y = \tilde{p}_x(x)$. This implies that the probability density of the x-coordinates of points that are accepted must be proportional to $\tilde{p}_x(x)$. In turn, this implies that the accepted samples amount to independent samples drawn from $p_x(x)$.

Rejection sampling can be generalized to several RVs in a straightforward way. In many cases, however, it is difficult to produce a proposal density $\tilde{p}_*(x)$ with the desired properties. In some of these cases, the problem can be obviated by the use of the *Metropolis method*. The main idea of the Metropolis method is to create a Markov chain whose transition matrix does not depend on the normalization term. One needs to make sure that the chain has a stationary distribution and such stationary distribution is equal to the target distribution. After a sufficient number of iterations, the chain will then converge to the stationary distribution.

To make these ideas more precise, recall that a (discrete time) Markov chain is a random process $\mathbf{X}_t \in \mathcal{S}$ (where $\mathcal{S}$ denotes sample space) that satisfies the Markov property: $\mathbb{P}[\mathbf{X}_{t+1} \mid \mathbf{X}_t, \ldots, \mathbf{X}_1] = \mathbb{P}[\mathbf{X}_{t+1} \mid \mathbf{X}_t]$. That is, the process has no memory: the future state of the system only depends on its present state, not on its past. A finite-state Markov chain (namely, one in which the cardinality of $\mathcal{S}$ is finite, $|\mathcal{S}| < \infty$) can be completely specified by the transition matrix $P = (p_{i,j})$ defined by the elements $p_{i,j} = \mathbb{P}[\mathbf{X}_{t+1} = j \mid \mathbf{X}_t = i]$. For irreducible chains, the stationary distribution π is the long-term proportion of time that the chain spends in each state. (Such a distribution can be computed noting that $\pi = \pi P$.) The Metropolis method makes use of a proposal density $p_*(\mathbf{X}; \mathbf{X}_t)$ that depends on the current state $\mathbf{X}_t$. More precisely, a single step of the Metropolis method proceeds as follows:

(i) Select a candidate move $\mathbf{X}_*$ generated from the current state $\mathbf{X}_t$ according to the proposal density $p_*(\mathbf{X}_*; \mathbf{X}_t)$.

(ii) Compute the ratio

$$r = \frac{p_{\mathbf{x}}(\mathbf{X}_*)\, p_*(\mathbf{X}_*; \mathbf{X}_t)}{p_{\mathbf{x}}(\mathbf{X}_t)\, p_*(\mathbf{X}_t; \mathbf{X}_*)}\,. \tag{21}$$

(iii) If $r \geq 1$, accept the move. Otherwise accept the move with probability r. (As in rejection sampling, this can be done by drawing a uniform RV u in $[0, 1]$ and accepting the move if $u < r$.)

(iv) If the move is accepted, set $\mathbf{X}_{t+1} = \mathbf{X}_*$. Otherwise remain in the current state (i.e., set $\mathbf{X}_{t+1} = \mathbf{X}_t$).

The approach is similar to rejection sampling, in that a candidate move is generated and then either accepted or rejected with a given probability. Two important differences, however, are that: (a) unlike rejection sampling, here the candidate move depends on the current state and (b) in rejection sampling, rejected points are discarded and have no influence on the list of samples collected, whereas in the Metropolis method a rejection causes the current state to be inserted again into the list of samples. We also note in passing that the original formulation of the method was done for the special case in which the proposal density is symmetric, i.e., $p_*(\mathbf{y}; \mathbf{x}) = p_*(\mathbf{x}; \mathbf{y})$, in which case (21) reduces simply to $r = p_{\mathbf{x}}(\mathbf{X}_*)/p_{\mathbf{x}}(\mathbf{X}_t)$. The more general version of the method described above should be more accurately called the Metropolis–Hastings method.

Unlike rejection sampling, the Metropolis method does not automatically generate samples from $p_{\mathbf{x}}(\mathbf{x})$. Rather, one can show that, for any positive proposal density $p_*(\mathbf{y}, \mathbf{x})$, the density of $\mathbf{X}_t$ tends asymptotically to $p_{\mathbf{x}}(\mathbf{x})$ in the limit $t \to \infty$. Nothing can be said in general about the rate of convergence, however, i.e., about how rapidly the convergence takes place. It is also important to realize that the samples generated by the Metropolis method are not statistically independent (which makes it difficult to compute variances). Indeed, the Metropolis method is our first example of *MCMC* methods, in which a Markov process is used to generate a sequence of states, each state having a probability distribution that depends on the previous state. Since successive samples are dependent, one may need to run the Markov chain for a considerable time in order to generate samples that are effectively independent. Finally, an important caveat is that the Metropolis method relies on diffusion to explore state space. This can be extremely slow and inefficient.

While rejection sampling and the Metropolis method can be used on 1D problems, *Gibbs sampling* (also known as the heat bath method or "Glauber dynamics") is a method for sampling from distributions in dimensions two or higher. The main idea of Gibbs sampling is to use conditional distributions. Consider for simplicity a two-dimensional example, with $\mathbf{X}_t = (X_{1,t}, X_{2,t})^T$. Suppose one has a situation where, while it is complicated to sample from the joint density $p_{\mathbf{x}}(\mathbf{x})$, it is feasible to draw samples from the two conditional

distributions $p_{x_2}(x_2 \mid x_1)$ and $p_{x_1}(x_1 \mid x_2)$. A single iteration of the Gibbs sampling method then proceeds as follows:

(i) Given the current state $\mathbf{X}^t$, generate a new value for X_1 using the conditional distribution $p_{x_1}(x_1 \mid X_{2,t})$.
(ii) Use the new X_1 to generate a new value for X_2 using the conditional distribution $p_{x_2}(x_2 \mid X_1)$ and set $\mathbf{X}_{t+1} = (X_1, X_2)^T$.

One can show that a single iteration of Gibbs sampling can be viewed as a Metropolis method with target density $p_{\mathbf{x}}(\mathbf{x})$, and that this particular implementation has the property that every candidate move is always accepted. Thus, as long as the joint density $p_{\mathbf{x}}(\mathbf{x})$ is reasonably nice, the probability distribution of the samples generated will tend to $p_{\mathbf{x}}(\mathbf{x})$ as $t \to \infty$.

Since Gibbs sampling is a special case of a Metropolis algorithm, it suffers from the same problems. Namely, samples are not independent, and generically speaking state space is explored by a slow RW. On the other hand, Gibbs sampling does not involve any adjustable parameters, and therefore, it is an attractive strategy when one wants to quickly test a new model. Also, various software packages are available that make it easy to set up and simulate a large class of probabilistic models by Gibbs sampling (Thomas et al., 1992).

7 APPLICATIONS OF VRTs TO ERROR ESTIMATION IN OPTICAL FIBER COMMUNICATION SYSTEMS

One of the areas in which IS and other VRTs have recently been applied with considerable success in recent years is the estimation of error probabilities in optical fiber communication systems (Agrawal, 2002; Kaminov and Koch, 1997). As an illustration of the methods discussed in this chapter, we devote this section to a brief review of the problem and of how the techniques that were presented in the previous sections were used in this context.

Errors in optical fiber communication systems are required to be extremely rare, e.g., the bit error ratio [that is, the probability of a transmission error] is required to be 10^{-12} or smaller after error correction. This stringent requirement imposes severe constraints on the design of these systems and creates a need for accurate design tools. On one hand, however, experiments are very expensive (the cost of setting up a fully equipped lab can exceed millions of dollars), and optimizing the system's performance involves selecting precise values for many independent parameters (such as input powers, pulse format, fiber types, and relative section lengths). Therefore, design engineers are in need of accurate mathematical and computational modeling. On the other hand, systems are large and complex, with many physical effects contributing to determine the overall system performance. Moreover, error probabilities are non-Gaussian due to nonlinear interactions. Hence, mathematical methods are alone not sufficient. But precisely because errors are required to be so rare, error probabilities cannot be estimated by standard MC simulations. An approach which has proved to be

successful in this situation is a hybrid one, in which the analytical knowledge of the dominant sources of error is used to design appropriate biasing strategies for IS.

There are two main sources of randomness that contribute to determine the overall system performance in optical fiber transmission systems: fiber disorder, manifesting itself in random birefringence, and amplified spontaneous emission noise from the optical amplifiers that are used to compensate for the attenuation of the signal due to fiber loss (Agrawal, 2002; Kaminov and Koch, 1997). We next briefly describe each of these two problems and the techniques that were brought to bear to study each of them. A further source of variability is the pseudo-randomness of the data stream, which can result in transmission errors through system nonlinearity. For brevity, however, we omit any discussion of this issue, and we refer the reader to Ablowitz et al. (1998), Mecozzi (1998), Sinkin et al. (2007), and references therein for details.

7.1 Polarization-Mode Dispersion

Birefringence arises when the speed of propagation of light in a medium depends on the polarization of the light itself. Although a great deal of effort is devoted to controlling all aspects of the manufacturing of optical fibers, a certain amount of fiber birefringence is always present. The presence of birefringence has the effect that an optical pulse will split into two components, propagating along what are called the fast and slow axes of birefringence. Moreover, the fiber's birefringence (including its strength and the birefringence axes) varies with wavelength, temperature, and time. The random, birefringence-induced perturbations on optical pulses are referred to as *polarization-mode dispersion* (PMD) (Kogelnik et al., 2002).

In most installed systems, PMD-induced impairments are completely determined by the real, three-component first- and second-order PMD vector, denoted, respectively, as $\vec{\tau}$ and $\vec{\tau}_\omega = \mathrm{d}\vec{\tau}/\mathrm{d}\omega$ (where ω is the optical frequency) (Kogelnik et al., 2002). In turn, the growth of PMD with distance is governed by the first- and second-order PMD concatenation equations (Gordon and Kogelnik, 2000), which describe how the first- and second-order PMD vectors of adjoined fiber sections combine with each other to produce the overall behavior of the system. In many cases, after performing an appropriate distance-dependent rotation of the reference frame can be written in the following simplified form (Biondini et al., 2004):

$$\vec{\tau}^{(n+1)} = \vec{\tau}^{(n)} + \Delta\vec{\tau}^{(n+1)}, \quad \vec{\tau}_\omega^{(n+1)} = \vec{\tau}_\omega^{(n)} + \Delta\vec{\tau}^{(n+1)} \times \vec{\tau}^{(n)}. \tag{22}$$

Here $\vec{\tau}^{(n)}$ and $\vec{\tau}_\omega^{(n)}$ are, respectively, the total first- and second-order PMD vectors after the nth fiber section, $\Delta\vec{\tau}^{(n)}$ is the first-order PMD vector of the nth fiber section, and $\Delta\vec{\tau}_\omega^{(n)}$ is the corresponding second-order PMD vector. The rescaled PMD vector $\Delta\vec{\tau}^{(n)}$ of each section can be assumed to be uniformly distributed on the Poincaré sphere; its magnitude $|\tau_n|$ obeys a Maxwellian

distribution with respect to wavelength. Also, for linearly birefringent sections, $\Delta\vec{\tau}_{\omega}^{(n)} = 0$.

The goal of system designers is to estimate the effects of PMD by performing numerical simulations, and in particular to quantify PMD-induced error probabilities. As mentioned before, however, the problem is events in which PMD takes on much larger than average values (resulting in transmission errors) are exceedingly rare. Thus, one would like to have a method to produce large first- and second-order PMD events more frequently than they would occur in practice, and weigh them with correct statistics. We next describe how this can be accomplished using IS. For simplicity, we describe the simplest case in which all fiber sections contribute the same amount of PMD to the total. We emphasize, however, that several other models of PMD generation have been considered in the literature, and a variety of IS and other VRTs have been used with success in all of these cases (Biondini and Kath, 2004, 2005; Biondini et al., 2004; Li et al., 2008, 2010; Lu and Yevick, 2005; Schuster et al., 2014; Secondini and Forestieri, 2005; Yevick, 2002).

It was shown in Biondini et al. (2004) that, when $|\Delta\vec{\tau}^{(n)}|$ is independent of n in (22), the appropriate variables to control in order to monitor the growth of PMD are the relative orientations of the individual sections, $\Delta\vec{\tau}^{(n)}$. To apply IS, one first needs to find the deterministic choices of $\Delta\vec{\tau}^{(n)}$ that maximize the desired combination of first- and second-order PMD. We will label these vectors $\vec{b}^{(n)}$. Once these vectors have been found, one can implement IS by biasing the random MC samples around them. To follow this idea, it is convenient to express the vectors $\vec{b}^{(n)}$ relative to a orthonormal frame of reference formed by the unit vectors $\{\hat{u}_1^{(n)}, \hat{u}_2^{(n)}, \hat{u}_3^{(n)}\}$, where

$$\hat{u}_1^{(n)} = \boldsymbol{\tau}^{(n)}/|\boldsymbol{\tau}^{(n)}|\,, \quad \hat{u}_2^{(n)} = \boldsymbol{\tau}_{\omega,\perp}^{(n)}/|\boldsymbol{\tau}_{\omega,\perp}^{(n)}|\,, \quad \hat{u}_3^{(n)} = \hat{u}_1^{(n)} \times \hat{u}_2^{(n)}\,. \tag{23}$$

Here $\boldsymbol{\tau}_{\omega,\perp}^{(n)}$ is the component of $\boldsymbol{\tau}_{\omega}^{(n)}$ perpendicular to $\boldsymbol{\tau}^{(n)}$. The first of Eq. (22) thus describes a simple 3D RW. Thus, if one only wants to maximize the length of the total first-order PMD vector τ, the best option is to choose $\vec{b}^{(n+1)}$ to be parallel to $\vec{\tau}^{(n)}$ (i.e., to align $\vec{b}^{(n+1)}$ along $\hat{u}_1^{(n)}$). On the other hand, the second of Eq. (22) couples the growth of second-order PMD to that of first-order PMD. Thus, if a nontrivial amount of second-order PMD is desired, one must also take into account the growth of first-order PMD.

When the number of sections is not too small (i.e., larger than 4 or 5), it was found convenient to employ a continuum approximation to find the deterministic biasing directions. Specifically, let $\lim_{\Delta z\to 0} \Delta\vec{\tau}_{n+1}/\Delta z = \vec{b}(z)$. The magnitude of $\vec{b}(z)$ describes the rate at which PMD is added by the birefringent sections. In this limit, one obtains

$$\frac{\mathrm{d}\vec{\tau}}{\mathrm{d}z} = \vec{b}\,, \quad \frac{\mathrm{d}\vec{\tau}_{\omega}}{\mathrm{d}z} = \vec{b} \times \vec{\tau}\,, \tag{24}$$

where z is the longitudinal direction along the fiber. Or, in the frame of reference $\{\hat{u}_1, \hat{u}_2, \hat{u}_3\}$ defined as above,

$$\frac{\mathrm{d}\tau}{\mathrm{d}z} = b_1\,, \quad \frac{\mathrm{d}\tau_{\omega,\parallel}}{\mathrm{d}z} = b_2\,\frac{\tau_{\omega,\perp}}{\tau}\,, \quad \frac{\mathrm{d}\tau_{\omega,\perp}}{\mathrm{d}z} = b_3\tau - b_2\,\frac{\tau_{\omega,\parallel}}{\tau}\,, \tag{25}$$

where (b_1, b_2, b_3) are now the components of $\vec{b}$ with respect to $\{\hat{u}_1, \hat{u}_2, \hat{u}_3\}$. The goal is now to find the function $\vec{b}(z)$ that maximizes second-order PMD or a linear combination of first- and second-order PMD. Fortunately, Eqs. (25) can be solved exactly for any $\vec{b}(z)$:

$$\tau(z) = \int_0^z b_1(\zeta)\,\mathrm{d}\zeta\,, \quad \tau_{\omega,\parallel}(z) = \int_0^z b_3(\zeta)\tau(\zeta)\sin[\beta(z,\zeta)]\,\mathrm{d}\zeta\,,$$

$$\tau_{\omega,\perp}(z) = \int_0^z b_3(\zeta)\tau(\zeta)\cos[\beta(z,\zeta)]\,\mathrm{d}\zeta\,, \tag{26a}$$

$$\beta(z,\zeta) = \int_\zeta^z \frac{b_2(\xi)}{\tau(\xi)}\,\mathrm{d}\xi\,. \tag{26b}$$

The choice of $\vec{b}(z)$ that maximizes the magnitude of second-order PMD (or any combination of first- and second-order PMD) can now be found using calculus of variations. (Detailed calculations can be found in Biondini et al., 2004.) The result is that the maximum growth of second-order PMD is obtained for "in-plane" contributions, namely, $(b_1, b_2, b_3) = b\,(\cos\alpha(z), 0, \sin\alpha(z))$ where the $\alpha(z)$ gradually interpolates between an initial value of 0 (producing pure first-order PMD at first) and a final value of $\pi/2$ (producing pure second-order PMD at the end). In particular, in the case of equal-length sections (namely, for $|\vec{b}(z)| = b$), the angle $\alpha(z)$ has a linearly varying profile: that is, $\alpha(z) = \alpha_{\max}\, z/z_{\max}$, with $\alpha_{\max} = \pi/2$. (The case of nonequal-length sections can be easily obtained from this one by rescaling the independent variable z; see Biondini et al., 2004 for details.) Performing IS-MC simulations with multiple biasing strengths, this biasing choice generates region 3 in Fig. 7.

In many practical situations, however, a more complete coverage of the $|\vec{\tau}||\vec{\tau}_\omega|$ plane is needed. In this case, intermediate biasing choices must also be used in addition to pure first- and second-order biasing. Such choices can be obtained by using calculus of variations to maximize a linear combination of $|\vec{\tau}|$ and $|\vec{\tau}_\omega|$, as obtained from Eqs. (26a). The resulting form of $\vec{b}(z)$ is the same as above, except that the value of the final angle $\alpha_{\max}$ now varies between 0 and π, the particular value depending upon the specific linear combination of first- and second-order PMD being maximized. A selection of angles, together with the resulting regions in the $|\vec{\tau}||\vec{\tau}_\omega|$ plane, is shown in Fig. 7. (Region 1 is the result in the case of biasing for pure first-order PMD.) The advantage of using multiple biasing—as opposed to just pure first- or second-order biasing or no biasing at all—is evident. Each value of $\alpha_{\max}$ generates samples lying in a region that emanates in a roughly radial fashion from the location where the joint pdf is maximum. Together, a set of angles $\alpha_{\max}$ can be used to cover the entire $|\vec{\tau}||\boldsymbol{\tau}_\omega|$ plane. Indeed, Fig. 8 shows the joint pdf of the magnitude of first- and second-order PMD (which is a two-dimensional reduction of the

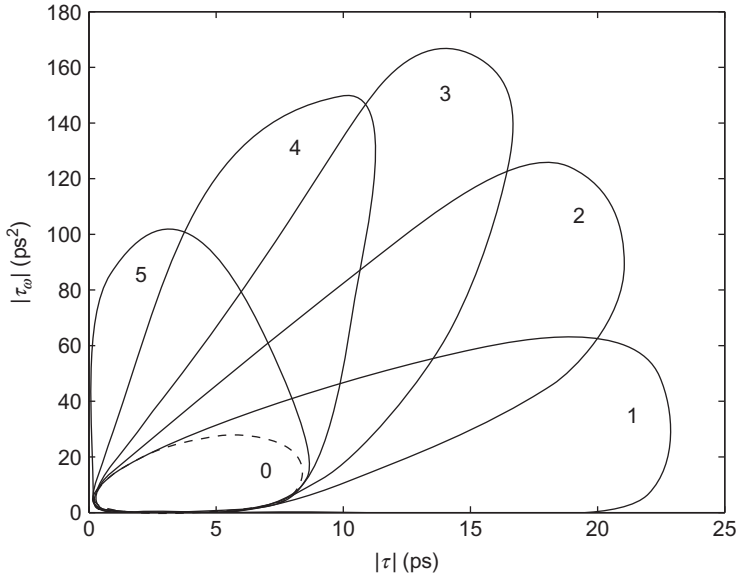


FIGURE 7 The regions of the $|\boldsymbol{\tau}||\boldsymbol{\tau}_\omega|$ plane targeted by the various biasing methods. Region 1 corresponds to pure first-order biasing ($\alpha_{\max} = 0$), region 2 to pure second-order biasing ($\alpha_{\max} = \pi/2$), and regions 3, 4, and 5 to $\alpha_{\max} = \pi/4$, $3\pi/4$, and π, respectively. The dashed line shows the much smaller region obtained with unbiased samples. Fifty birefringent sections with 0.5 ps DGD each were used. *Source: From Biondini et al. (2004).*

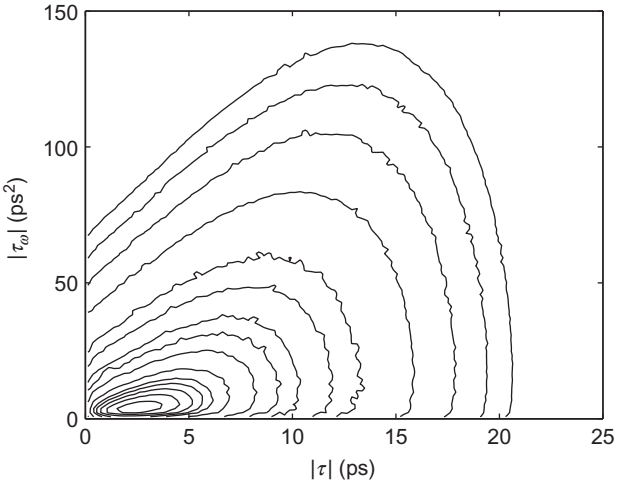


FIGURE 8 Contour plots of the joint pdf of first- and second-order PMD for a concatenation of 50 birefringent sections with 0.5 ps DGD each, as reconstructed from IS-MC simulations. The contours are at 10^{-n} with $n = 1.5, 1.75, 2, 2.25, 3, 4, 5, 6, 8, 10, 15, 20, 25$, and 30. A total of 10^6 Monte-Carlo samples were used. *Source: From Biondini et al. (2004).*

full 3D joint pdf of first- and second-order PMD; Foschini and Poole, 1991) for a system of 50 polarization scramblers, as calculated with the multiple biasing technique described above. In a similar fashion, one can use the same biasing strategies in numerical simulations of pulse transmissions to quantify PMD-induced transmission errors.

7.2 Noise-Induced Perturbations

Together with the invention of the laser in 1960, the birth of optical fiber transmission systems was made possible by the development of low-loss optical fibers, with typical loss coefficients of 0.2 dB/km. Nonetheless, for long-distance communication systems, which span thousands of kilometers, fiber loss remains a serious obstacle, which is compensated by inserting optical fiber amplifiers at various points along the transmission line. Modern optical amplifiers allow the signal to be boosted in the optical domain, avoiding the need for electronic conversion. The downside of this process, however, is the introduction of spontaneous emission photons, which get combined to the signal in the form of additive white Gaussian noise. In addition, since the fiber is weakly nonlinear, the noise interacts with the signal to generate random pulse fluctuations. While these perturbations are not too large on average, they are one of the main sources of errors.

The propagation of optical pulses in fibers is governed by a perturbed nonlinear Schrödinger (NLS) equation with varying coefficients (Agrawal, 2007):

$$\mathrm{i}\frac{\partial q}{\partial z} + \frac{1}{2}d(z)\frac{\partial^2 q}{\partial t^2} + g(z)|q|^2 q = \mathrm{i}S(t,z)\,. \tag{27}$$

Here z is the dimensionless propagation distance, t is the dimensionless retarded time, $q(t,z)$ is the dimensionless slowly varying electric field envelope (rescaled to account for loss and amplification in communication systems), $d(z)$ is the local value of the dispersion coefficient, and $g(z)$ describes the periodic power variations, which are due to loss and amplification. The source term $S(t,z)$ can represent various kinds of perturbations. Here, we focus on the physically interesting case of spontaneous emission noise originating from the optical amplifiers. That is, we consider

$$S(t,z) = \sum_{n=1}^{N_a} v_n(t)\delta(z - nz_a),$$

where N_a is the number of amplifiers, z_a is the dispersion map period, $\delta(z)$ is the Dirac delta distribution, and $v_n(t)$ is white Gaussian noise, satisfying $\mathbb{E}[v_n(t)] = 0$ and $\mathbb{E}[v_n(t)v_{n'}^*(t')] = \sigma^2\delta(t-t')\delta_{nn'}$. In other words, at each amplifier, $z = nz_a$, Eq. (27) is replaced by the jump condition $q(t, nz_a^+) = q(t, nz_a^-) + \sigma v_n(t)$. We note in passing that the numerical simulation of (27) involves a very large (several tens of thousands in practical situations) number of RVs comprised by

the real and imaginary parts of $S(z,t)$ at each of the collocation points in time for each amplifier over the whole transmission line.

In the simplest case of constant dispersion and no gain/loss power variations, without loss of generality one can take $d(z) = g(z) = 1$. In this case, when $S(z,t) = 0$, Eq. (27) is a completely integrable model that admits an infinite number of exact solutions describing elastic interactions among N particle-like objects called solitons (Ablowitz and Segur, 1981; Zabusky and Kruskal, 1965). The simplest case is that of a 1-soliton solution, which is simply the traveling wave solution

$$q(t,z) = A\,\mathrm{sech}[A(t-T)]\,\mathrm{e}^{\mathrm{i}\theta(t,z)}\,, \tag{28}$$

where $\theta(t,z) = V(t-T) + \Phi$ and with $T(z) = Vz + t_o$ and $\Phi(z) = \frac{1}{2}(A^2 + V^2)z + \phi_o$. Note that the 1-soliton solution (28) contains four constant parameters: the amplitude A (which is also its inverse width), the frequency V (which is also the group velocity offset), a temporal offset t_o, and a phase offset ϕ_o.

The case when $d(z)$ and $g(z)$ are not constant but periodic describes a periodic concatenation of fibers with different dispersion properties and is referred to as dispersion management (DM) in the literature. Equation (28) is replaced by a more complicated pulse shape, and the resulting pulses are called dispersion-managed solitons (DMS). Nonetheless, the invariances of the equation imply that DMS still contain the same four pulse parameters. In this case, one can use suitable perturbation methods to derive an equation, called dispersion-managed nonlinear Schrödinger (DMNLS) equation which captures all the essential features of the dynamics as well as the DMS pulse shape (Ablowitz and Biondini, 1998; Spiller and Biondini, 2010).

When noise is present [i.e., $S(t,z) \neq 0$], the nonlinear term in Eq. (27) causes part of the noise to couple to the soliton and induce random deviations of the soliton parameters. One can use perturbation theory on either the NLS or the DMNLS equation to capture the effects of noise on the soliton parameters, obtaining (Li et al., 2007)

$$\frac{\mathrm{d}A}{\mathrm{d}z} = S_A(z)\,, \quad \frac{\mathrm{d}V}{\mathrm{d}z} = S_V(z)\,, \quad \frac{\mathrm{d}T}{\mathrm{d}z} = V + S_T(z)\,,$$

$$\frac{\mathrm{d}\Phi}{\mathrm{d}z} = \frac{1}{2}(A^2 + V^2) + V\,S_T(z) + S_\Phi(z)\,, \tag{29a}$$

where the source terms,

$$S_j(z) = \langle \mathrm{e}^{\mathrm{i}\theta}\bar{y}_j, S\rangle \big/ \langle \bar{y}_j, y_j\rangle\,, \quad j = A, V, T, \Phi, \tag{29b}$$

which are defined in terms of the inner product $\langle f, g\rangle = \mathrm{Re}\int f^*(t)g(t)\,\mathrm{d}t$, are the projection of the noise along the neutral modes y_j of the linearized NLS operator around the soliton solution. Each neutral mode is associated with one of the invariances of the NLS equation as well as with infinitesimal changes in

one of the soliton parameters. Note that since the linearized NLS operator is not self-adjoint, the modes are not mutually orthogonal, and therefore, the projection must be done using the corresponding adjoint modes $\bar{y}_j$. On the other hand, the neutral modes and their adjoints form a biorthogonal basis for the null space of the linearized NLS operator: $\langle \bar{y}_j, y_k \rangle = \langle \bar{y}_j, y_j \rangle \, \delta_{jk}$, where δ_{jk} is the Kronecker delta.

Equations (29a) are a system of nonlinear stochastic differential equations, which cannot be solved in closed form. (The nonlinearity arises not only from the explicit appearance of A and V in the equations but also, and in a more essential way, on the fact that the source terms depend on the soliton amplitude A.) Useful information can still be extracted from them, however. For the present discussion, it is convenient to employ a continuum approximation of the noise. That is, we consider $S(t, z)$ to be a Gaussian white noise process with zero mean and autocorrelation function $\mathbb{E}[S(t, z)S^*(\tau, \zeta)] = \sigma^2 \, \delta(t - \tau)\delta(z - \zeta)$. As a result, the source terms in Eqs. (29a) become independent white noise processes, with autocorrelation function

$$\mathbb{E}[S_j(z)S_k^*(\zeta)] = \sigma_j^2 \delta_{jk} \delta(z - \zeta) \,, \tag{30}$$

where the source term variances are

$$\sigma_j^2 = \mathrm{var}[S_j(z)] = \mathbb{E}\big[\langle \mathrm{e}^{\mathrm{i}\theta} \bar{y}_j, S \rangle^2 \big/ \langle \bar{y}_j, y_j \rangle^2\big] = \sigma^2 \|\underline{y}_j\|^2 \big/ \langle \bar{y}_j, y_j \rangle^2 \,. \tag{31}$$

In the limit of moderate amplitude deviations, one can approximate Eqs. (29a) by considering the variances of the source terms to be constant. The resulting equations can then be integrated exactly, to obtain

$$A(z) = A_o + W_A(z) \,, \quad V(z) = V_o + W_V(z) \,,$$

$$T(z) = T_o + \int_0^z V(\zeta) \mathrm{d}\zeta + W_T(z) \,, \tag{32a}$$

where for brevity we omitted the expression for $\Phi(z)$, and where

$$W_j(z) = \int_0^z S_j(\zeta) \, \mathrm{d}\zeta \,, \quad j = A, V, T, \Phi, \tag{32b}$$

is a Wiener process with zero mean and autocorrelation function $\mathbb{E}[W_j(z)W_k(\zeta)]$ $= \sigma_j^2 \delta_{jk} \min(z, \zeta)$. The mean values of the soliton parameters at the output $z = L$ are then

$$\mathbb{E}[A(L)] = A_o \,, \quad \mathbb{E}[V(L)] = V_o \,, \quad \mathbb{E}[T(L)] = T_o + V_o L,$$

$$\mathbb{E}[\Phi(L)] = \frac{1}{2}(A_o^2 + V_o^2) \, L + \frac{1}{4}(\sigma_A^2 + \sigma_V^2) \, L^2 \,. \tag{33}$$

Tedious but straightforward stochastic calculus (Papoulis, 1991) also yields the variances of the noise-perturbed output soliton parameters (Spiller and Biondini, 2010):

$$\text{var}[A(L)] = \sigma_A^2 L\,, \quad \text{var}[V(L)] = \sigma_V^2 L\,,$$

$$\text{var}[T(L)] = \sigma_T^2 L + \frac{1}{3}\sigma_V^2 L^3\,, \tag{34}$$

where the expression for var$[\Phi(L)]$ was again omitted for brevity. (Note how the mean phase is directly affected by the noise, unlike the other soliton parameters.) The cubic dependence of timing and phase jitter on distance (which arise, respectively, as a result of the coupling between carrier frequency and group velocity and as a result of the Kerr effect and Galilean invariance) are well-known in the optics literature and are referred to as Gordon–Haus jitter (Gordon and Haus, 1986) and Gordon–Mollenauer jitter (Gordon and Mollenauer, 1990), respectively.

The above mean variances agree very well with direct numerical simulations of the full NLS equation perturbed by noise. However, their knowledge is not sufficient to accurately estimate noise-induced transmission penalties, for several reasons. First of all, the variances are only correct for small deviations of the pulse amplitude, whereas we are interested in quantifying the probability of large deviations. Second, even though the noise is Gaussian-distributed, the noise-induced changes of the soliton parameters are not necessarily Gaussian. In particular, the variance of each amplitude shift depends on the previous value of the amplitude, which causes the distribution of A to deviate significantly from Gaussian. A Gaussian approximation will therefore only be valid in the limit of small amplitude shifts, and even then only in the core region of the pdf and not in the tails. Finally, even if the noise-induced changes of the soliton parameters were approximately Gaussian-distributed, calculating the probability densities in the tails from the (analytically or numerically obtained) variances would require an exponential extrapolation, and any errors or uncertainties would be magnified correspondingly.

Nonetheless, the information obtained from the above perturbation theory is the key to devise a successful IS for the problem, as we show next. In our case, to successfully apply IS one must find the most likely noise realizations that lead to a desired change of the soliton parameters at the output. As demonstrated in Moore et al. (2003) and Li et al. (2007), one can approach this problem by decomposing it into two logically distinct steps: (i) finding the most likely noise realizations that produce a given parameter change at each amplifier and (ii) finding the most likely way in which individual parameter changes at each amplifier combine to produce a total change at the output. This two-step approach is justified by the fact that the noise at different amplifiers is statically independent. We next briefly describe each of these two steps.

(i) Biasing at a single amplifier. Consider a generic perturbation to the solution at the nth amplifier, $b_n(t)$. Recall from Eqs. (29) that the noise-induced change to a soliton parameter Q (with $Q = A, V, T, \Phi$) is found by taking the projection of the perturbation onto the adjoint mode of the linear DMNLS operator associated

with Q. That is, if $q(t, nz_a^+) = q(t, nz_a^-) + b_n(t)$, the change to parameter Q due to the perturbation $b_n(t)$ is given by

$$\Delta Q_n = \mathrm{Re} \int \bar{y}_Q^* b_n(t) \mathrm{d}t \Big/ \int |\bar{y}_Q|^2 \mathrm{d}t \,. \tag{35a}$$

The problem of finding the optimal biasing at each amplifier is to find the most likely noise realization subject to the constraint of achieving, on average, a desired parameter change at that amplifier. In other words: given a specific parameter change ΔQ_n at the nth amplifier (with $Q = A, V, T, \Phi$), what is the form of $b_n(t)$ that is most likely to produce this prescribed change? For white Gaussian noise, maximizing its probability amounts to minimizing the negative of the log-likelihood, i.e., the negative of the argument of the exponential in the noise pdf. That is, we need to minimize the L_2 norm of the noise,

$$\|b_n(t)\|^2 = \int |b_n(t)|^2 \mathrm{d}t \,, \tag{35b}$$

subject to achieving the desired parameter change ΔQ_n given by Eq. (35a). One can formulate this as a variational problem, whose solution yields the deterministic biasing direction (Moore et al., 2008)

$$b_n(t) = \Delta Q_n \big(\mathrm{Re} \int \bar{y}_Q^* y_Q \mathrm{d}t \Big/ \int |\bar{y}_Q|^2 \mathrm{d}t \big) \, \bar{y}_Q \,. \tag{36}$$

(ii) Biasing across all amplifiers. Next we address the question of how one should distribute the bias for the soliton parameters among all amplifiers in order to achieve a specified parameter change at the output. In other words: what is the most likely set of individual parameter changes $\{\Delta A_n, \Delta V_n, \Delta T_n, \Delta \Phi_n\}_{n=1,\ldots,N_a}$ that realizes a given value of ΔQ_{target} (with Q equal to either A, V, T, or Φ, as before) at the output? For simplicity, we limit our discussion to amplitude deviations, even though the same approach can be used to study variations of all four soliton parameters (Spiller and Biondini, 2010).

We begin by examining the amplitude evolution from one amplifier to the next, namely

$$A_{n+1} = A_n + \Delta A_{n+1} \,. \tag{37}$$

Recall that the most likely noise realization that achieves a given amplitude change at a single amplifier is given by (36), with $Q = A$ in this case. Also recall that the norms and inner products of the linear modes depend on the soliton amplitude and therefore also indirectly on distance. It should be clear that maximizing the probability of obtaining a given amplitude at the output is equivalent to minimizing the sum of the L_2 norm of the biasing functions $b_n(t)$ over all amplifiers. That is, we need to minimize the sum

$$\sum_{n=1}^{N_a} \|b_n\|^2 = \sum_{n=1}^{N_a} |\Delta A_n|^2 / \sigma_A^2 \,, \tag{38a}$$

subject to the constraint

$$\sum_{n=1}^{N_a} \Delta A_n = A_{\text{target}} - A_o \,, \tag{38b}$$

where σ_A^2 is given by Eq. (31). To solve this problem, we consider a continuum approximation. That is, we replace Eq. (37) by the first of Eqs. (29a), with $S(t,z) = b(t,z)$ and $b(t,z)$ given by the continuum analogue of Eq. (36) with $Q = A$, that is: $b(t,z) = (\langle y_A, \bar{y}_A \rangle / \|\bar{y}_A\|^2)\, \bar{y}_A(t)\, \dot{A}$. We then seek a function $A(z)$ that minimizes the continuum limit of Eq. (38a). That is, we seek to minimize the integral from $z = 0$ to $z = L$ of the L_2 norm of $b(t,z)$, namely, the functional

$$J[A] = \int_0^L \dot{A}^2 / \sigma_A^2 \, \mathrm{d}z \,, \tag{39}$$

subject to the fixed boundary conditions $A(0) = A_o$ and $A(L) = A_{\text{target}}$ [which are the continuum limit of (38b)]. Hereafter, the dot denotes total differentiation with respect to z, and L is the total transmission distance as before. After some straightforward algebra, the Euler–Lagrange equation associated with the functional $J[A]$ in (39) can be written as

$$2\ddot{A}\, \frac{1}{\sigma_A^2} + \dot{A}^2 \frac{\partial}{\partial A}\left(\frac{1}{\sigma_A^2}\right) = 0 \,,$$

which is readily integrated to give

$$\dot{A} = c\ \sigma_A \,, \tag{40}$$

where c is an integration constant which determines the total amount of biasing being applied and thereby the value of the amplitude at the output. One can now integrate Eq. (40) to find the optimal path $A(z)$ that realizes a desired amplitude change at the output. Once this path has been obtained, one can then calculate ΔA_n, which was the only unknown in the optimal biasing directions b_n in Eq. (36).

Equation (40) can be solved exactly in the case of constant $d(z)$ and $g(z)$ (that is, for the classical NLS equation). In this case, Eq. (40) reduces to $\dot{A} = c\sqrt{A}$, which is trivially integrated to $A_{\text{nls}}(z) = \left[\left(\sqrt{A_{\text{target}}} - \sqrt{A_o}\right) z/L + \sqrt{A_o}\right]^2$. When $d(z)$ or $g(z)$ are not constant, the functional dependence of σ_A on A is not known explicitly, and therefore, it is not possible to integrate Eq. (40) analytically. Numerical expressions are available for the norms and inner products, however, so one can proceed by numerically integrating $\dot{A}$, obtaining an expression for $z = z(A)$, and then inverting this expression to find the optimal biasing paths. As an example, Fig. 9 shows the results of numerical simulations in which the MC samples were biased along the optimal paths (shown by the thick curves) that produce three given amplitude changes (also indicated in the figure), demonstrating how the random trajectories are indeed closely clustered around these paths. Figure 10 shows the pdf of the output energy as reconstructed

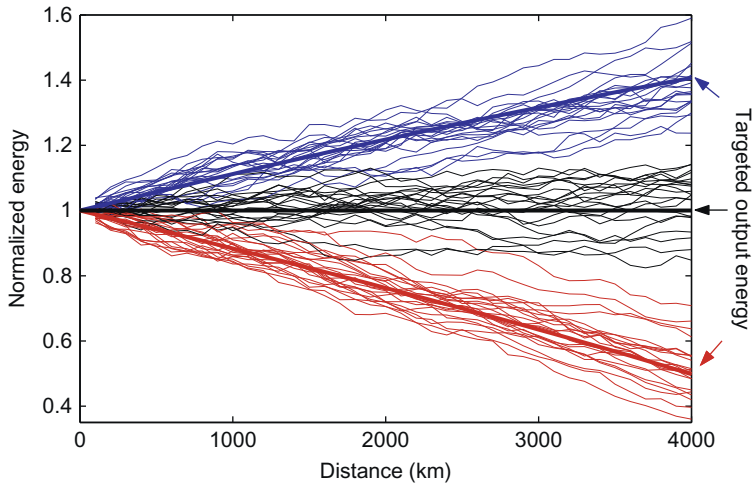


FIGURE 9 Samples from IS-MC simulations of the DMNLS equation. Here, the pulse energy (normalized to input energy) is plotted as a function of time (i.e., distance in physical units). The arrows represent the different targeted output energies: a larger than normal output energy (blue (dark gray in the print version)), a smaller than normal output energy (red (light gray in the print version)), and unbiased energy (black). Also plotted are deterministic paths (thick, smooth curves, with color corresponding to the target) predicted by our perturbation theory. These are the preferential paths around which we attempt to sample by biasing the noise with the adjoint linear modes. For each of three different targeted output energies, a few dozen IS-MC samples are also shown (also colored correspondingly), demonstrating that the actual trajectories indeed follow the predictions of the theory. *Source: From Li et al. (2007).*

from IS-MC simulations of the DMNLS equation using multiple IS and the biasing techniques described above. For comparison purposes, the results of unbiased MC simulation of the original NLS equation (27) with DM and a much larger number of MC samples are also shown, as well as a Gaussian fit to those results, demonstrating that pdf deviates significantly from a Gaussian, and at the same time that IS-MC simulation is an effective to quantify the probability of rare events in the system.

Similar techniques have been recently applied to quantify the effect of noise-induced perturbations in a variety of other system configurations, e.g., see Donovan and Kath, 2011; Li and Kath, 2015; Li et al., 2007; Moore et al., 2003, 2005, 2008; Spiller and Biondini, 2009, 2010; and references therein.

8 LARGE DEVIATIONS THEORY, ASYMPTOTIC EFFICIENCY, AND FINAL REMARKS

A key concept in assessing the effectiveness of a given biasing strategy and/or when using IS to reconstruct a sequence of quantities with decreasing probability (as in the case of the pdf in the example in Section 4.3) is that of asymptotic

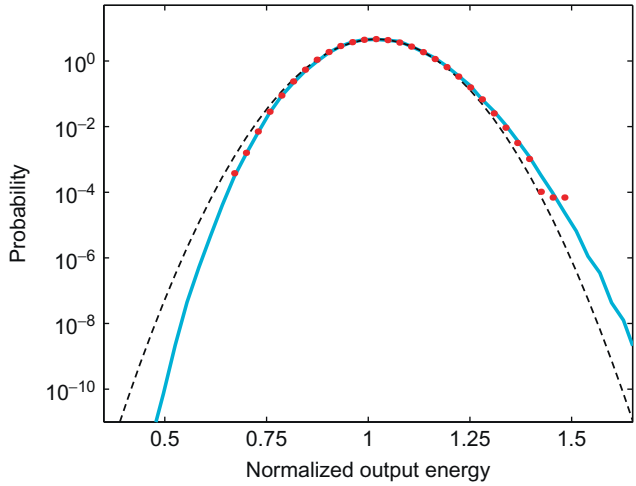


FIGURE 10 pdf of normalized output energy of a dispersion-managed soliton affected by amplifier noise. The solid (cyan (light gray in the print version)) curve shows results from IS-MC simulations of the DMNLS equation with 42,000 samples. The (red (dark gray in the print version)) dots are the results from standard MC simulations of the NLS equation with DM with 1,000,000 samples. The (black) dashed curve is a Gaussian fit to that simulation. Note how unbiased MC simulations of the NLS equation with DM deviate from Gaussian, but agree well with IS-MC simulations of the DMNLS equation as far as down in probability as the unbiased simulations can reach. *Source: From Li et al. (2007).*

efficiency (Glynn and Whitt, 1992; Sadowsky and Bucklew, 1990). The precise definition of asymptotic efficiency is formulated in the framework of large deviations theory (Bucklew, 1990; Dembo and Zeitouni, 1983). Here we will limit ourselves to giving an informal discussion of both of these topics.

Often, for simplicity, the choice of biasing distributions is restricted to a specific family of distributions, usually dependent on one or more parameters, e.g., in a specific situation these could be the mean translation parameters. Now consider a set of probabilities P_n dependent on a parameter n, e.g., P_n could be defined as the probability that the RV $y(\mathbf{X})$ takes values that are larger than n times its mean: $P_n = \mathbb{P}[y(\mathbf{X}) > n\mu]$, with $\mu = \mathbb{E}[y(\mathbf{X})]$. As another example, let $Y_n = (X_1 + \cdots + X_n)/n$ be the mean of n i.i.d. RVs $X_1, \ldots, X_n$. One could ask what is the probability that Y_n deviates more than ϵ from its mean, i.e., $P_n = \mathbb{P}[|Y_n - \mu| > \epsilon]$, where now $\mu = \mathbb{E}[X]$. Furthermore, suppose that the probabilities P_n tend to zero as $n \to \infty$, as is indeed the case in the two examples given. Large deviations theory is concerned with the *rate* at which these probabilities tend to zero. In this sense, it can be thought of as an extension of the law of large numbers.

It is often the case in practical situations that the probabilities P_n decay exponentially as n increases. Loosely speaking, when this happens we say that the sequence $\{P_n\}_{n\in\mathbb{N}}$ satisfies a *large deviations principle*. More explicitly, in

the above example we say that P_n satisfies a large deviations principle with rate function $I(\epsilon)$ if

$$\lim_{n\to\infty} \frac{1}{n} \log P_n = -I(\epsilon) .$$

More precise and comprehensive definitions can be given, which allow one to include a larger class of processes, for some of which the simple requirement above is not satisfied. A large body of work has been accumulated on large deviations theory. Two famous results, namely Cramér's theorem and the Gärtner–Ellis theorem, identify some properties of rate functions. In particular, for the sum of RVs considered above, one can show that the rate function is

$$I(\epsilon) = \sup_{s\in\mathbb{R}} [s\epsilon - \log(M(s))] ,$$

where $M(s) = \mathbb{E}[\exp(sX)]$ is the moment-generating function. For further details, we refer the reader to Bucklew (1990) and Dembo and Zeitouni (1983).

Now let us return to the problem of rare event simulation. It should be clear that the computational cost required for an accurate estimation of P_n with standard MC methods will obviously grow with n. Next, consider a sequence of biasing distributions $p_{*n}(\mathbf{x})$. Roughly speaking, the sequence is said to be *asymptotically efficient* if the computational burden grows less than exponentially fast.

The concept of asymptotic efficiency has important practical consequences. If a family of biasing distributions is asymptotically efficient, the increase in computational efficiency will be larger and larger the further we reach into smaller probabilities. The best-case scenario is that in which the computational cost to reach probability levels of 10^{-n} is independent of n. In that case, the increase in computational efficiency can be arbitrarily large in principle, and in practice is just dictated by how far down in probability we need to reach. We refer the reader to Bucklew (2004) for a discussion of precise conditions that guarantee that a sequence of simulation distributions is asymptotically efficient.

As a final remark, we should comment on the relation between large deviations theory and the study of random dynamical systems. In many cases, one can think of the input RVs as perturbations affecting the behavior of a dynamical system. For example, in the case of optical fiber communication systems, three kinds of randomness are present: (i) the fiber's random birefringence, which depends on distance, time, and wavelength; (ii) the optical amplifiers' quantum noise, which is added to the signal and propagates nonlinearly through the fiber; and (iii) the pseudo-random sequence of information bits. The problem of studying small random perturbations of dynamical systems was first posed in Pontryagin et al. (1933) and has received considerable attention in recent years. In many cases, the most likely configuration of RVs for which the system reaches a given output state can be thought of as a specific path in sample space. In turn, this path can be uniquely identified as the minimizer of the Wentzell–Freidlin action functional (Freidlin and Wentzell, 1984). IS can then

be thought of simply as a numerical (MC) technique to perform an integration in sample space around this "optimal" path. (Note the similarity between this point of view and the path integral formulation of quantum mechanics, e.g., see Weinberg, 1995.) The best-case scenario is of course that in which this optimal path can be identified analytically (e.g., as in Biondini et al., 2004; Moore et al., 2008). In other situations, however, one may be able to solve the minimization problem numerically (as in Spiller and Biondini, 2010). Finally, if this is also not practical, one can avoid the Wentzell–Freidlin formulation altogether and search for it adaptively using the cross-entropy method (as in Donovan and Kath, 2011; Marzec et al., 2013; Schuster et al., 2014).

REFERENCES

Ablowitz, M.J., Biondini, G., 1998. Multiple scale dynamics in communication systems with strong dispersion management. Opt. Lett. 23, 1668–1670.

Ablowitz, M.J., Segur, H., 1981. Solitons and the Inverse Scattering Transform. Society for Industrial and Applied Mathematics, Philadelphia.

Ablowitz, M.J., Biondini, G., Chakravarty, S., Horne, R.L., 1998. On timing jitter in wavelength-division multiplexed soliton systems. Opt. Commun. 150, 305.

Agrawal, G.P., 2002. Fiber optics communication systems. Wiley, New York.

Agrawal, G.P., 2007. Nonlinear Fiber Optics. Academic Press, New York.

Biondini, G., Kath, W.L., 2004. PMD emulation with Maxwellian length sections and importance sampling. IEEE Photon. Technol. Lett. 16, 789–791.

Biondini, G., Kath, W.L., 2005. Polarization-dependent chromatic dispersion and its impact on return-to-zero transmission formats. IEEE Photon. Technol. Lett. 17, 1866–1868.

Biondini, G., Kath, W.L., Menyuk, C.R., 2004. Importance sampling for polarization mode dispersion: techniques and applications. IEEE J. Lightwave Technol. 22, 1201–1215.

Bucklew, J.A., 1990. Large Deviation Techniques in Decision, Simulation and Estimation. Wiley, New York.

Bucklew, J.A., 2004. Introduction to Rare Event Simulation. Springer, New York.

Chan, C.C., Kroese, D.P., 2012. Improved cross-entropy method for estimation. Stat. Comput. 22, 1031–1040.

de Boer, P.-T., Kroese, D.P., Rubinstein, R.Y., 2004. A fast cross-entropy method for estimating buffer overflows in queueing networks. Manag. Sci. 50, 883–895.

de Boer, P.-T., Kroese, D.P., Mannor, S., Rubinstein, R.Y., 2005. A tutorial on the cross-entropy method. Ann. Oper. Res. 134, 19–67.

Dembo, A., Zeitouni, O., 1983. Large Deviation Techniques and Applications. Jones & Bartlett, Boston.

Donovan, G.M., Kath, W.L., 2011. An iterative stochastic method for simulating large deviations and rare events. SIAM J. Appl. Math. 71, 903–924.

Fishman, G.S., 1996. Concepts, Algorithms and Applications. Springer-Verlag, Monte Carlo.

Fishman, G.S., 2006. A First Course in Monte Carlo. Thomson, Belmont.

Foschini, G.J., Poole, C.D., 1991. Statistical theory of polarization dispersion in single mode fibers. IEEE J. Lightwave Technol. 9, 1439.

Freidlin, M.I., Wentzell, A.D., 1984. Random Perturbations of Dynamical Systems. Springer-Verlag, New York.

Glynn, P.W., Whitt, W., 1992. The asymptotic efficiency of simulation estimators. Oper. Res. 40, 505.

Gordon, J.P., Haus, H.A., 1986. Random walk of coherently amplified solitons in optical fiber transmission. Opt. Lett. 11, 665–667.

Gordon, J.P., Kogelnik, H., 2000. PMD fundamentals: polarization-mode dispersion in optical fibers. Proc. Natl. Acad. Sci. U.S.A. 97, 4541–4550.

Gordon, J.P., Mollenauer, L.F., 1990. Phase noise in photonic communications systems using linear amplifiers. Opt. Lett. 15, 1351–1353.

Homem-de-Mello, T., Rubinstein, R.Y., 2002. Rare event probability estimation using cross-entropy. In: Yucesan, E., Chen, C.-H., Snowdon, J.L., Charnes, J.M. (Eds.), Proceedings of the 2002 Winter Simulation Conference. pp. 310–319.

Kaminov, I.P., Koch, T.L. (Eds.), 1997. Optical Fiber Telecommunications IIIA. Academic Press, New York.

Knuth, D.E., 2011. The Art of Computer Programming, vols. I–IV. Addison-Wesley, Boston.

Kogelnik, H., Nelson, L.E., Jopson, R.M., 2002. Polarization mode dispersion. In: Kaminow, I.P., Li, T. (Eds.), Optical Fiber Telecommunications IVB. Academic Press, pp. 725–861.

Kroese, D.P., Rubinstein, R.Y., 2004. The transform likelihood ratio method for rare event simulation with heavy tails. Queueing Syst. 46, 317–351.

Kroese, D.P., Taimre, T., Botev, Z.I., 2011. Handbook of Monte Carlo Methods. Wiley Series in Probability and Statistics. Wiley, New York.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. Ann. Math. Stat. 22, 79–86.

Landau, D.P., Binder, K., 2000. A Guide to Monte Carlo Simulations in Statistical Physics. Cambridge University Press, Cambridge.

Li, J., Kath, W.L., 2015. Predicting and simulating rare, large deviations in nonlinear lightwave systems. preprint.

Li, J., Spiller, E.T., Biondini, G., 2007. Noise-induced perturbations of dispersion-managed solitons. Phys. Rev. A 75 (053818), 1–13.

Li, J., Biondini, G., Kath, W.L., Kogelnik, H., 2008. Anisotropic hinge model for polarization-mode dispersion in installed fibers. Opt. Lett. 33, 1924–1926.

Li, J., Biondini, G., Kath, W.L., Kogelnik, H., 2010. Outage statistics in a waveplate hinge model of polarization-mode dispersion. IEEE J. Lightwave Technol. 28, 1958.

Lima, A.O., Lima, I.T., Menyuk, C.R., 2005. Error estimation in multicanonical Monte Carlo simulations with applications to polarization-mode-dispersion emulators. IEEE J. Lightwave Technol. 23, 3781–3789.

Lu, T., Yevick, D., 2005. Efficient multicanonical algorithms. IEEE Photon. Technol. Lett. 17, 861–863.

MacKay, D.J.C., 2003. Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge.

Marzec, Z., Schuster, J., Biondini, G., 2013. On the efficiency of importance sampling techniques for polarization-mode dispersion in optical fiber transmission systems. SIAM J. Appl. Math. 73, 155–174.

Mecozzi, A., 1998. Timing jitter in wavelength-division-multiplexed filtered soliton transmission. J. Opt. Soc. Am. B 15, 152.

Metropolis, N., 1987. The beginning of the Monte Carlo method. Los Alamos Sci. 15, 125–130 (special issue).

Metropolis, N., Ulam, S., 1949. The Monte Carlo method. J. Am. Stat. Assoc. 44, 335–341.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. J. Chem. Phys. 21, 1087–1092.

Moore, R.O., Biondini, G., Kath, W.L., 2003. Importance sampling for noise-induced amplitude and timing jitter in soliton transmission systems. Opt. Lett. 28, 105–107.

Moore, R.O., Schafer, T., Jones, C.K.R.T., 2005. Soliton broadening under random dispersion fluctuations: importance sampling based on low-dimensional reductions. Opt. Commun. 256, 439–450.

Moore, R., Biondini, G., Kath, W.L., 2008. A method for the study of large noise-induced perturbations of nonlinear Schrödinger solitons using importance sampling. SIAM Rev. 50, 523–549.

Owen, A., Zhou, Y., 2000. Safe and effective importance sampling. J. Am. Stat. Assoc. 95, 135.

Papoulis, A., 1991. Probability, Random Variables and Stochastic Processes. McGraw Hill, New York.

Pontryagin, L.S., Andronov, A.A., Vitt, A.A., 1933. O statisticheskom rassmotrenii dinamicheskikh sistem. Zh. Eksp. Teor. Fiz. 3, 165–180.

Rubinstein, R.Y., Kroese, D.P., 2004. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning. Springer, New York.

Sadowsky, J.S., Bucklew, J.A., 1990. On large deviations theory and asymptotically efficient Monte Carlo estimation. IEEE Trans. Inf. Theory 36, 579.

Schuster, J., Marzec, Z., Kath, W.L., Biondini, G., 2014. A hybrid hinge model for polarization-mode dispersion in installed fiber links. J. Lightwave Technol. 32, 1412–1419.

Secondini, M., Forestieri, E., 2005. All-order PMD outage probability evaluation by Markov chain Monte Carlo simulations. IEEE Photon. Technol. Lett. 17, 1417–1419.

Sinkin, O.V., Grigoryan, V.S., Menyuk, C.R., 2007. Accurate probabilistic treatment of bit-pattern-dependent nonlinear distortions in BER calculations for WDM RZ systems. IEEE J. Lightwave Technol. 25, 2959.

Smith, P.J., Shafi, M., Gao, H., 1997. Quick simulation: a review of importance sampling techniques in communications systems. IEEE J. Select. Areas Commun. 15, 597.

Spiller, E.T., Biondini, G., 2009. Phase noise of dispersion-managed solitons. Phys. Rev. A 80 (011805), 1–4.

Spiller, E.T., Biondini, G., 2010. Importance sampling for the dispersion-managed nonlinear Schrödinger equation. SIAM J. Appl. Dyn. Syst. 9, 432–461.

Srinivasan, R., 2002. Importance Sampling: Applications in Communications and Detection. Springer, New York.

Thomas, A., Spiegelhalter, D.J., Gilks, W.R., 1992. Bugs: a program to perform Bayesian inference using Gibbs sampling. In: Bernardo, J., Berger, J., Dawid, A., Smith, A. (Eds.), Bayesian Statistics 4. Clarendon Press, Oxford, pp. 837–842.

Veach, E., 1997. Robust Monte Carlo methods for light transport simulation. Ph.D. thesis, Stanford University, California.

Weinberg, S., 1995. The Quantum Theory of Fields, vol. I. Cambridge University Press, Cambridge.

Yevick, D., 2002. Multicanonical communication system modeling—application to PMD statistics. IEEE Photon. Technol. Lett. 14, 1512–1514.

Zabusky, N.J., Kruskal, M.D., 1965. Interaction of solitons in a collisionless plasma and the recurrence of initial states. Phys. Rev. Lett. 15, 240–243.

Chapter 3

A Large-Scale Study of Language Usage as a Cognitive Biometric Trait

Neeti Pokhriyal[*,1], Ifeoma Nwogu[*,1], Venu Govindaraju[*,1]
[*]*Department of Computer Science and Engineering, University at Buffalo, State University of New York, NY-14221*
[1]*Corresponding authors: e-mail: neetipok@buffalo.edu; inwogu@buffalo.edu; venu.govindaraju@gmail.com*

ABSTRACT
In this chapter, we discuss a novel biometric trait, called **cognitive biometrics**. It is defined as the process of identifying an individual through extracting and matching unique signatures based on the cognitive, affective, and conative state of that individual. Currently, there is an increasing need for novel biometric systems that engage multiple modalities because of the changing notion of privacy in today's world. Also, cognitive biometrics will become essential as pervasive computing becomes more prevalent, where computing can happen anywhere and anytime. The use of cognitive traits for biometrics is relatively underexplored in the research community. We study whether the cognitive state of a person can be learned and used as a soft biometric trait and discuss our large-scale experimental setup, which yielded encouraging results.

Keywords: Soft biometrics, Linguistic analysis, Classification, Authentication, Authorship attribution

1 INTRODUCTION

Biometrics is the science of verifying the identity of an individual through the physical or behavioral traits. Physical biometrics extracts signatures from fingerprints, hand geometry, iris patterns, etc. Behavioral biometrics involve finding patterns of users based on gait, speech, signature, and handwriting. Physical and behavioral are two prominent and often studied modalities in the study of biometrics.

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00003-4

The notion of privacy is changing in today's world. Thus, we need novel biometric systems that engage multiple modalities for authentication of an individual. As we increase the quantities of personal and identifiable data on cloud networks, such as Google drives, Dropbox, iCloud, etc., passwords alone will no longer suffice as the only authentication methods. Also with the limits on physical biometric modalities (fingerprint, hand, iris, etc.), and the ever-changing ways of human–machine interactions, there is an urgent need to develop newer methods of authentication that go beyond existing password driven authentication. Passwords suffer from a variety of vulnerabilities including brute-force and dictionary-based attacks, and as a result, these highly personalized data on the cloud become vulnerable to being "stolen." Thus, there is an imminent need to shift focus from user-proxy-driven-authentication to individual-biometric-driven-authentication. In an effort to develop individuality-driven-authentication, new biometric modalities are needed, as standalone or fused with existing modalities to make an informed decision about the identity of the user at the console.

Also, user expectations have immensely changed with respect to authentication systems with the advent of newer modes of communication and computing. The user wants to be continuously authenticated even as more private and secure communication takes place over the console, than previously known. Since the user is simultaneously logged into a number of devices and computing environments, he/she wants minimal interruption for any authentication information. An authentication system that identifies person based on his/her cognition, has better chances of continuously verifying an individual's identity, than a system based on possession of passwords or keys.

Cognitive biometrics is defined as the process of identifying an individual through extracting and matching unique signature based on the cognitive, affective, and conative state of that individual. Cognitive biometrics like many behavioral patterns, falls under the category of *soft biometrics*. Soft biometrics, described by handwriting, speech, gait, etc., are, at times, relatively easy to capture and process. These can, then, be used in combination with physical biometrics, which are often more difficult and obtrusive to obtain and work with. Cognitive biometrics can be used alone or in conjunction with other biometric modalities depending on the level of security required.

As studies by psycholinguist, Carroll (2007) has shown people often develop a jargon that is more meaningful to other people in similar age groups, social backgrounds, academic settings, etc. Thus, language becomes a badge of sorts, identifying information about the user. Given the psychology-based evidence, it has been shown that language of a user does provide distinctive characteristics that can be used as a cognitive biometric trait for authentication.

While traditional biometric models use physical characteristics of individuals such as face, fingerprints, retina scans, voice, etc., using language expressed by an individual as an identifying characteristic has not been explored in this context before. The empirical results are encouraging and show that users do have distinctive writing styles, which is evident even when analyzing a corpora

as large and diverse as the Internet. Typically, language written by an individual encompasses various facets, like handwritten material, or structured text like books and other formal publications. Of particular interest are the *weblogs* (or *blogs*), which are unstructured digital text authored on the Internet, because of their easy availability, and reflection on how people author text informally in their day-to-day settings.

Blogs are extremely popular ways by which people express their thoughts and concerns and is an important way of communication over the Web. For instance, there are around 172 million Tumblr and 75.8 million WordPress blogs. The data generated from the "blogosphere" is one of key types of big data available on the Internet. While *text analytics* from big text corpora, such as blogs, has been a widely researched problem. Most focus has been to infer the latent information in the text. Here, we discuss the role of such big and unstructured data in understanding if language usage of an individual can be learnt as his/her cognitive signature.

Some of the questions that we discuss in this chapter include:

- Can language usage of an individual has enough discriminatory power to be used as a cognitive biometric trait?
- Does it qualify as a successful biometric trait based on its properties, namely: identifiability, universality, uniqueness, permanence, collectability, performance, and acceptability?
- Are unstructured blogs on the Internet a good metric to capture the cognitive signatures of authors?
- What are the quantifiable metrics for such a system: How does the number of blogs/texts per author, the total number of authors and the quality of blogs (richer, cleaner text) affect the results?
- What comprises of a cognitive biometric system: Is it the linguistic style of the authors (how they write), or is it the context in which the language is used (what they write), or is it a combination of the two?

We have performed an extensive set of experiments under various settings to empirically obtain some answers to these questions above. We varied the number of blogs per author, the total number of authors, and the size of the entire dataset, in order to study how well language can be used as cognitive biometric trait. We have, also, extracted linguistic style of the authors well as learnt the context on which the blogs are written from the data, and combined the two for our analysis. We describe our methodology and report the findings in the ensuing sections of the chapter.

The rest of the chapter is organized as follows. We introduce the problem of cognitive biometrics in Section 2. Section 3 describes the data used to extract cognitive fingerprints of the users. The details of the methodology is given in Section 4. The results of our experiments are described in Section 5. For further discussions on the study, see Section 6. Existing research in relevant areas is discussed in Section 7. Lastly, Section 8 concludes the chapter and provides future directions for researchers interested in this area.

2 COGNITIVE FINGERPRINTS: PROBLEM DESCRIPTION

We define a cognitive fingerprint of a user as a unique signature that captures the user's mental thoughts or experiences (or cognition). Individuals express their thoughts in a multitude of ways, either using spoken words or written language. As language is the medium of communication and has varied connotations (spoken word, gestures, handwritings, etc.,) attached to it; in this work, we only refer to the written words of a person. We exclude hand-written texts, which have been extensively studied in literature for biometric identification (Zhu et al., 2000).

To get a realistic dataset of authors and their natural writings, we looked for the blogosphere data, where people typically identify themselves with a username, and pen down their thoughts as entities called *blogs*. The dataset, used in this work is the ICWSM 2009 Spinn3r dataset (Burton et al., 2009) where the corpus consists of 44 million entries, a snapshot of weblog activities from August 1, 2008 and October 1, 2008.

Hence, our goal is to develop a cognitive-biometric authentication system that can determine whether any given individual, posing to be an enrollee, is genuine or an impostor. This involves obtaining biometric scores for each (user, enrollee) pair, where the pair is genuine if the enrollee is very similar to the user, or else is an impostor. The problem is posed as a binary classification problem, where the two classes are genuine and impostor. Each data point in this space is a feature vector representation of the blogs written by the user, such that the feature values are small if the blogs are written by the same author, and large if written by another user. They are dissimilarity scores, with low scores meaning low dissimilarity (or high similarity), and high scores meaning high dissimilarity. Our proposed method learns a classifier that can distinguish between genuine and impostor authors. As with a typical biometric authentication system, users are initially enrolled into the system and we treat this as our training/validation dataset. We then perform different quantitative tests including an ROC analysis to show the biometric strength of using language as a cognitive biometric trait.

3 DATA DESCRIPTION

The ICWSM 2009 Spinn3r data was collected in 2009, as a part of 3rd International AAAI Conference on weblogs and Social Media sponsored Data Challenge. The raw data is obtained by crawling various blog publishing sites to get the syndicated text of the blogs and the associated contents. Thus, the corpus consists of a snapshot of weblog activities from August 1, 2008 and October 1, 2008. It consists of 44 million entries in Spinn3r.com Website's XML format.

The raw format includes the RSS and the ATOM descriptors and also several meta-data tags. The blogs are also arranged in 13 tier groups based on the blog influence. Since the data is obtained as a snapshot of the blogosphere at a certain time, a portion of the items contain only the first few hundred

characters rather than the entire text of the weblog entry. Also as evident in cyber space, much of the data is not *real* blog entries, like many are posts in threaded online conversations, advertisements, blogs just posting images, forwarding mails.texts, etc. Additionally the dataset consisted of blogs written in many languages, including English.

To get blogs that contained real text written by authors, we chose a subset of the ICWSM 2009 Spinn3rDataset, which is called the Personal Stories Dataset (Gordon and Swanson, 2009). This dataset, consisted of the only the blogs which can be best characterized as a personal story. The logic, behind such decision, was our intuition that personal stories are expected to contain more distinguishing writing style markers. Thus we get blogs which are personal stories written by the blog author. Depending on the nature of Internet, many of the blogs which were marked as personal stories, however, had no author name associated with the ⟨authorname⟩ tag in their XML mark-up. We chose the unique identifier as the text contained in the ⟨authorname⟩ within each XML ⟨item⟩ and discarded the blogs which had no author.

The data is organized into 13 tiers, with stories spread across all tiers, but the most number of stories are found in Tier 1. The characteristics of the data for all tiers and for Tier 1 are summarized in Table 1. We report statistics for authors who have written at least 15 blogs. Since Tier 1 has the most number of stories (Gordon and Swanson, 2009), we chose Tier 1 for majority of our experiments.

To find the distribution of blogs per author, we found that some authors write lots of blogs, while a lot of authors write few blogs. Thus, the number of blogs per author follows power law as shown in Fig. 1.

Table 2 summarizes the statistics regarding authors and their corresponding blogs, who have written a minimum of 5, 15, or 30 blogs. As expected, as the minimum number of blogs per author increased, we got fewer authors, on average, they have written more number blogs.

TABLE 1 Data Characteristics for All Tiers and Only for Tier 1 of the Spinn3r Dataset

	All Tiers	Tier 1
# Blogs	153,796	65,945
# Words	61,703,661	24,724,073
# Authors	3510	1714
Avg. # blogs	44	38
Median # blogs	22	21
Max. # blogs	2477	1400

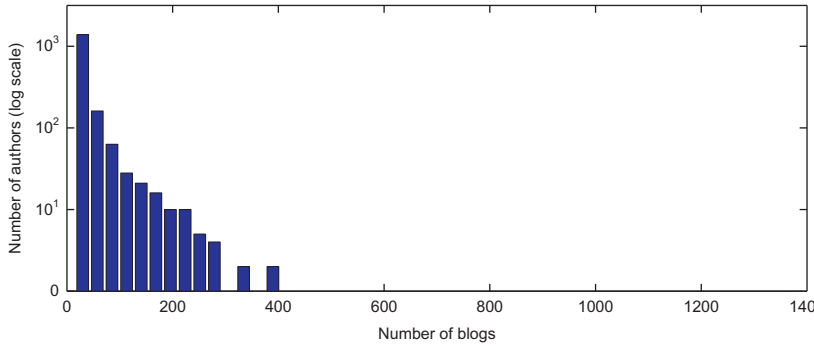


FIGURE 1 Distribution of number of blogs per author for all (1417) authors.

TABLE 2 Data Characteristics for Blogs in Tier 1

	1	5	15	30
# Blogs	248246	153119	65945	41655
# Words	611M	58M	25M	15M
# Authors	56338	11926	1714	498
Avg. # blogs	4	13	38	83
Median # blogs	2	8	21	53

4 METHODOLOGY

Our methodology has following four steps:

1. Data Preparation: The Spinn3r dataset consists of blogs and their associated content in an XML format. We extracted personal stories from the above-mentioned XML dump, using the information and code provided by the ICWSM website (Gordon and Swanson, 2009). We extracted only those blogs whose authors chose to identify themselves with ⟨authorname⟩ tag within the XML mark-up. We removed a lot of software debris and other XML markup in order to have only the content written by the author, which we refer to as the *description* of the blogs, i.e., everything that is written within the ⟨desc⟩ tags of the XML mark-up.

2. Feature Extraction: We extracted two sets of features from the descriptions:

(i) Stylistic Features that characterize the text and capture the writing style of the authors. These features have been used in the past in different contexts as effective discriminators of authorship. They fall in numerous categories including lexical (like word count), syntactic (like the frequency of stop words),

TABLE 3 Stylistic Features

Feature #	Description	Number
Length	Number of unique words/characters in blog	2
Vocabulary richness	Yule's K	1
Word shape	Frequency of words with different combinations of upper case and lower case letters	5
Word length	Frequency of words that have 1–20 characters	20
Letters	Frequency of a to z, ignoring case	26
Digits	Frequency of 0 to 9	10
Punctuation	Frequency of punctuation characters	11
Special characters	Frequency of other non-alphabet non-digit characters	21
Function words	Frequency of special words like "the," "of," etc.	117

structural (like the paragraph length), and personal (like lexical diversity). To extract these features, we computed different statistics from the description of the blogs so that in total we extracted 213 stylistic features detailed in Table 3. The stylistic features were calculated quantitatively from the data. Some of the features like the number of unique words, number of digits, letters, punctuation used were calculated by writing regular expressions to search and count the number of their occurrences in the texts. Vocabulary richness was calculated using a variant of the *Yule's K* function ($= \frac{M_1}{M_2}$), where M_1 is the number of all unique *stemmed words* in the text. The stemming was done using the standard *Porter's stemming algorithm* (Manning et al., 2008). M_2 is calculated as $\sum_{i=1}^{K} f(s_i)^2$, where $f(s_i)$ is the number of times the i^{th} stemmed word occurs in the text. If a text has rich vocabulary, the above Yule's K measure for that text will be high.

Except for vocabulary richness, number of unique words, and number of characters in the blog, all other stylistic features are ratios or frequencies, i.e., each of the feature value is divided by the number of characters in the blog. This is done so that the feature values are not biased towards the length of the blogs written by a particular author.

(ii) Semantic Features (or thematic features) capture the themes running through the blogs, i.e., contexts or topics of that the author talks about in the blogs. To get these features, we applied the *Latent Dirichlet Distribution* (LDA) algorithm (Blei et al., 2003) (also known as *topic modeling*) on the description of

the blogs to extract the dominant topics, or themes that pervade our collection of large, unstructured blogs data. We used the Mallet toolbox (McCallum, 2002) for to achieve this. The tool takes as input a collection of texts, which in our cases are the blogs written by authors, and outputs the main topics, or themes that exist in the collection. The topics are thus a distribution over the words that exist in the collection. As an additional input, the tool also requires the number of topics. We tested with 20, 50, and 100 topics and on visually inspecting them, we found that 50 topics gave a good enough representation of the themes in the blog descriptions. Table 4 presents a listing of some of the topics and the top 19 words associated with the topics. A full list of the topics and top words are included in the supplementary material (http://dx.doi.org/10.1016/B978-0-444-63492-4.00003-4). The semantic features, corresponding to each blog, were the topic proportions of each of the 50 topics, i.e., probability values associated with each of the 50 topics.

In all, we used 213 stylistic features and 50 semantic features. For our analysis, we worked with the stylistic, and the semantic features both separately and combined (the two feature vectors were concatenated to create a new vector of 263 (213 + 50) features), to examine how performance was affected by the different classes of features. To reduce the computational complexity of the dataset, we used some heuristics to reduce the size of the dataset; in one set of experiments, we extracted a fixed number of authors from the entire corpus,

TABLE 4 A Sample of Topics Inferred from the Blogs

Topic #	Top Words
4	night bar beer drink people party drunk drinking club drinks place wine good dance dancing friends guy pretty fun
7	mom dad family house sister home brother mother parents years year aunt time father daughter remember uncle day kids
13	store bought shopping buy shoes mall found clothes pair shirt wear stuff mom shop find wearing home wanted cute
16	school class year teacher classes room college day high today students back test homework grade student campus week semester
21	food dinner chicken eat good cheese lunch restaurant ate made eating salad bread sauce delicious meal soup rice ordered
30	water beach back lake day trip park river trail mountain road miles beautiful fish island found area boat hike
44	birthday party wedding year happy day christmas cake friends dinner fun made family love time gift great pictures night
47	doctor pain hospital blood sick baby feeling day appointment back surgery nurse told started days gave dr bad feel

while in another, we extracted a range or a fixed number of blogs written by a particular author.

3. Training a Classifier: We performed experiments for both identification and authentication. For biometric identification, the problem of authorship attribution can be formulated as a multiclass classification problem, where we have the classes as the authors. For biometric authentication, the problem of authorship attribution can be formulated as a binary classification problem, with the two classes being genuine and impostor classes. For this study, we investigated language as a biometric trait primarily in the context of biometric authentication, where we posed the problem of authorship attribution as a binary classification problem. We can visualize a square matrix, whose rows/columns are the number of authors, where each cell is a score of matching N users with N enrollees. Enrollee i and user i is the same person, and enrollee i and user j ($i \neq j$) are different persons. Each user is matched against each enrollee once. Thus the diagonal elements correspond to genuine matches and nondiagonal elements correspond to impostor matches.

To get the biometric score between a user and an enrollee, we used pairwise difference for all the features, for all blogs. This difference is used a biometric score, such that a low values between blog A and blog B signifies that both are written by the same author (a genuine match), while a large value between blog A and blog B signifies that they are written by different authors (an impostor). Since we have multiple blogs for each author, the genuine scores consist of blocks around the diagonal (unlike the traditional case where only the diagonal entries are the genuine scores).

Thus, for each data point we take the difference of individual feature values of this point from every other point. Since we already know the labels of the data-points, we know if the difference feature vector corresponds to a genuine or an impostor score. The difference feature vector is the same size of the feature vector length, i.e., 263. Thus, we create a dataset consisting of genuine and impostor data points, with 263 length feature vectors, labeled as genuine or impostors.

4. Evaluating Performance of Classifier: We chose the classifier that gave the maximized area under the ROC curve, which is our chosen performance metric. We worked with Support Vector Machine, K-Nearest Neighbor based classifier, and Logistic Regression on a very small subset of the data (40 authors) and found that *logistic regression* (Bishop, 2006) gave the best results. Hence all experiments were performed with this classifier using the version implemented in Weka (Hall et al., 2009), as multinomial logistic regression model with a ridge estimator (set at 1.0×10^{-8}). Before training the classifier, the data was standardized, i.e., all feature values were set to have zero mean and unit variance.

For the multiclass classification approach, during testing, each blog had to be attributed to one of the classes, however, on an Internet-style author attribution dataset like ours, the number of authors is very large (order of 50 K in this case), and the text written by a most of the authors was small (average is 4). Thus

we got a large number of classes, with very limited number of data points for each class. To alleviate this issue, we simplified the problem by using a single unique signature for each author. We computed the median of the feature values for all the blogs of a particular author as the signature. Each new blog was then compared against this signature, and we took the nearest three neighbors that were closest to the median of each of the authors. We also tested with the mean of the blogs, instead of the median. Mean gave better results than the median, and together the thematic and stylistic features give better results than individual features alone.

Rare-class Mining: Note that in the binary classification data that we create, there are only N genuine scores, and $N^2 - N$ impostor scores, where N is the number of authors. Thus, the result is a highly imbalanced dataset, with a lot more impostor data points than genuine ones. Typically, classification algorithms are designed under the assumption of a relatively uniform/balanced distribution of classes for training. But in a problem such as our genuine/impostor identification, there is a need to better balance the number of data-points in each class. This problem is referred to as *rare-class mining*, or *masking problem* which often leads to misleading results; the accuracy of the classification algorithms can be high even though it potentially misclassifies all or many of the points of the minority class. This problem can be handled by: synthetically oversampling the minority class; undersampling the majority class; and generating samples so that the resulting distribution of the two classes are balanced.

The over- and undersampling methods artificially add or remove data to achieve balance, hence we adopted the more realistic approach of generating balanced sample sets. We used a Bernoulli distribution to randomly select data points that were added to the sample. We used two different Bernoulli distributions to select the genuine and impostor samples and the parameters of the distributions were set to ensure a balance between the samples of the two classes.

5 RESULTS

5.1 Evaluating Performance on Different Types of Data

Table 5 compares the results of combining the data from all tiers. The results degrade when all the tiers of the data are used. We think the plausible reasons could be the following: The blogs are organized into various tiers by their influence score. Thus more influential blogs are in Tier 1 and least in Tier 13. We hypothesize that the quality of the text written by authors degrade as we go to less influential blogs, and it becomes increasingly difficult to discern authors. Additionally, we only consider the blogs written by an author who is identified with an author name in the XML dump of the dataset. It may happen that among various tiers, we may have different authors using the same author names, but

TABLE 5 AUC Scores When Blogs from All Tiers are Used Versus Blogs from Tier 1

	AUC		
Data	Stylistic	Semantic	both
550-All tiers	0.60	0.59	0.61
550-Tier1	0.67	0.70	0.71

TABLE 6 Classification Results Using Logistic Regression for Various Types of Features with Different Number of Authors (*N*)

# of Authors (*N*)	AUC		
	Stylistic	Semantic	Both
200	0.61 (0.075)	0.65 (0.066)	0.65 (0.077)
550	0.67 (0.064)	0.70 (0.059)	0.71 (0.068)
All	0.66 (0.053)	0.68 (0.050)	0.70 (0.056)

Standard deviation across multiples runs is reported in parenthesis.

in our analysis thus far, we assume that all blogs having the same author name were written by the same author.

5.2 Evaluating Performance of the Biometric Trait

For this experiment, we chose a random sample of N authors out of a total of 1714 authors (see Table 1) who had written more than 15 blogs. We created a dataset of genuine and impostor samples as discussed above. We trained a logistic regression classifier on part of this data and tested the performance of the classifier on a held-out dataset (34% of the original dataset). This experiment was repeated 10× to capture the variance in performance. We use *Area under the ROC Curve* (AUC) as the evaluation metric. Additionally, we experimented with three types of features for each setting: stylistic (see Table 3), semantic or topic-based (using 50 topics learned by LDA), and both combined. Table 6 shows the results obtained using different values of N (200, 550, and 1714). As evident from the table, the AUC does not vary significantly with the number of authors. With a smaller number of authors, the execution runs very fast, as the complexity of generating genuine samples is of the order of N, and the

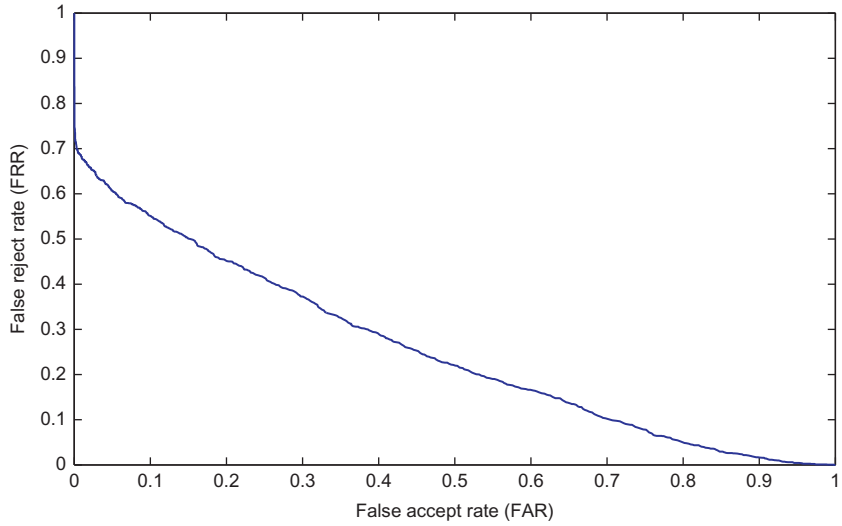


FIGURE 2 ROC curve (x-axis is the false accept rate and y-axis is the true accept rate) using logistic regression on Tier 1 data.

complexity of generating impostor samples is quadratic in N, but otherwise, the accuracy does not change significantly. In Fig. 2, we report the ROC curve for one instance where the number of authors is 550.

5.3 Impact of Features

To gain additional insight into the impact of the types of features, we used **information gain** as a metric to measure the relative importance of the features. Figure 3 indicates that some of the most discerning features are one of the topics, blog size and frequency of some of the function words. The effectiveness of the function words is also reported in literature (Mosteller and Wallace, 1964). It is interesting to note that the topic with the most discriminating power in the feature space consisting of a topic which is mostly *adult* words, which co-occurred together a lot of times in the text. This signifies that there are authors who chose to write blogs on adult themes and make use of such words quite a bit. This buttresses the fact that stylistic features along with a semantic ones offer good metrics of discriminating authors. Notice from the AUC results in Table 6 that the accuracies are higher when both the feature sets are combined.

5.4 Using Authors with Different Minimum Number of Blogs

In this section, we investigate the question: *What is the minimum number of words (text) required to efficiently learn to distinguish between genuine and*

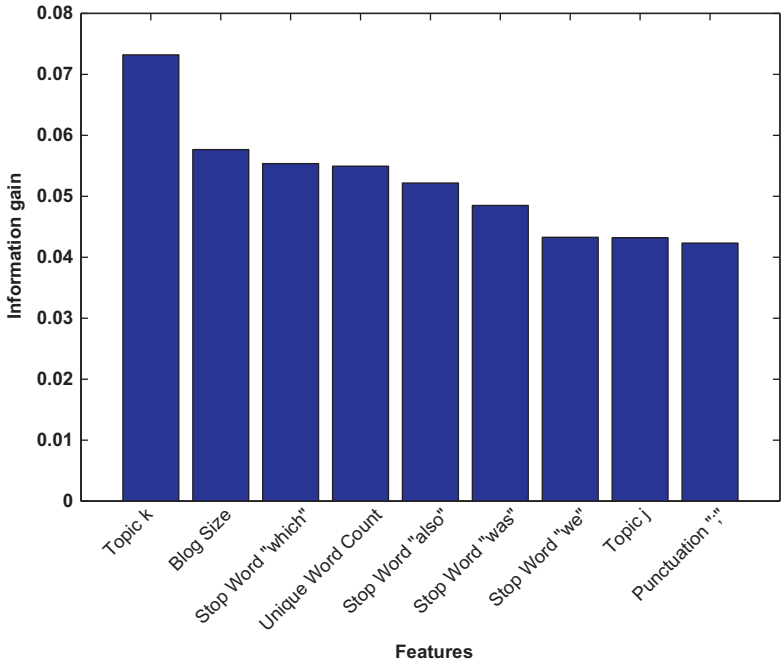


FIGURE 3 Top features using information gain.

impostor authors? To study this, we took authors who wrote at least 5, 10, or 15 blogs in Tier 1. In Table 7, we report the results using other values of k, where k is the minimum number of blogs written. The experiments were repeated $3 - -5\times$ to capture the variance in performance. The results indicated that 15 was the best number of blogs with which we could efficiently learn to distinguish between genuine and impostor authors. However, this heuristic is data dependent. The results may be poor at 30, because more number of blogs add more variance, and thus more confusion for the classifier. We require additional experiments to obtain a conclusive answer.

5.5 Varying the Number of Blogs per Author

In this section, we investigate the question: *Given a fixed number of users (or authors), what is the minimum number of text required to train a biometric system such that it efficiently learns to distinguish between genuine users and impostors*. For this, we fixed the minimum number of blogs as 5, 15, and 30 for a comparative study. We calculated the number of authors who had written at least 30 blogs in Tier 1, which is 498 authors (out of 1714). We then randomly sampled p blogs ($p \leq 30$) for each author and generated the biometric data, where p was 5, 15, or 30.

TABLE 7 AUC with Different Thresholds on Minimum Number of Blogs Written per Author (k)

Min # of Blogs (k)	AUC		
	Stylistic	Semantic	Both
5	0.62 (0.003)	0.67 (0.006)	0.64 (0.018)
15	0.67 (0.069)	0.68 (0.063)	0.71 (0.072)
30	0.62 (0.076)	0.64 (0.092)	0.65 (0.091)

Standard deviation across multiple runs is reported in parenthesis.

TABLE 8 Classification Results Using Logistic Regression for Different Types of Features with Different Number of Blogs per Author (p) Used for Training

# of Blogs (p)	AUC		
	Stylistic	Semantic	Both
5	0.73 (0.002)	0.75 (0.008)	0.75 (0.005)
15	0.65 (0.007)	0.67 (0.015)	0.69 (0.006)
30	0.60 (0.049)	0.62 (0.000)	0.64 (0.044)

Standard deviation across multiple runs is reported in parenthesis.

The results in Table 8 suggest that if we have a fixed number of authors, then we may achieve a good AUC value, even with a smaller number of blogs per author. One possible reason is that increasing the number of blogs per author increases the variability per author, and thus tends to give poorer results. This is also data dependent. As we use more blogs per author for the same author, we add more variance in the blog writings, which may extend to both the style of writing and the thematic contents of the blogs.

5.6 Odd Man Out Analysis

We performed two types of analyses to evaluate how well the developed system generalizes to data from users who were not included in the training data.

We assume that we have trained our system to distinguish between genuine and impostor users. If we have new users that our system has never seen before, how effectively can the system detect whether the user is a genuine author (these authors were not included in the training data, but have been successfully enrolled in the system)? To evaluate this, we set aside a set of authors who were not used for training the classifier (*test authors*). For each blog written by a test

author, we compared it with all the other blogs written by the same author, let the count of other blogs be denoted as n. We then counted the number of times our classifier successfully predicted these pairs as a genuine match, denoted as m. Ideally, the fraction $\frac{m}{n}$ should be greater than 0.5. Our classifier correctly classified **78%** of such matches as genuine.

Similarly, we compared each blog written by a test author with blogs written by another randomly selected test author where both sets of authors and their data have never been seen by the classifier. We then counted the number of times our classifier predicted these matches as impostors (m). Again, the fraction $\frac{m}{n}$ should be ideally greater than 0.5. Our classifier correctly classified **76%** of such matches as impostors.

This strongly indicates that our methodology has not just overfit to the styles of the authors used for training, but in general, has learned to a large extent, how to distinguish between an pair of writing styles, whether or not the writings originate from the authors that have been seen before by the classifier during training.

6 DISCUSSIONS

This is also a study of biometrics in big data, i.e., when we have unprecedented amount of data available for biometric authentication and verification tasks. Big data implies that we not only need to address the challenges in volume, but also in the variety and uncertainty of the data. Our study deals with millions of blogs written by tens of thousands of authors. Here are some of our learnings while working with such data:

1. We need to develop methods that are computationally inexpensive and scales well with the increasing data. We use a random sample of genuine and impostor data points, as the actual number of data points become prohibitively large to store and analyze,
2. Since the inception of the biometric system, developers need to carefully choose programming environments and software tools appropriate for handling big data. Given that this analysis requires computing distance between every pair of blogs, the complexity of this authorship attribution is $O(N^2)$ in the number of blogs. To address this challenge, we used efficient data structures (such as *associative maps* in Java and Python) and scalable procedures involving large matrices.
3. To reduce the uncertainty in the data, we constrained the dataset, by taking only the blogs which are personal stories written by an identifiable author.

7 RELATED WORK

Since we are extracting cognitive signatures for the authors of written texts, we discuss a similar problem, called *authorship-attribution*. This problem has been heavily studied in literature, where given a certain

author, the goal is to determine whether a not-previously-seen piece of writing can be attributed to that author. The problem has been studied in different guises with datasets of varying sizes and types. Stylometric analysis techniques have been used for attributing authorship in the past (see Federalist papers; Mosteller and Wallace, 1964) and for more recent surveys see Koppel et al. (2009) and Stamatatos (2009). The studies on authorship attribution are performed in varied databases: theater plays (Mendenhall, 1887), essays (Mosteller and Wallace, 1964; Yule, 1939), biblical books (Mealand, 1995), book chapters (Koppel et al., 2007), emails (Abbasi and Chen, 2008; de Vel et al., 2001; Koppel and Schler, 2003), Web forum messages (Abbasi and Chen, 2005; Thamar et al., 2011), blogs (Koppel et al., 2006, 2011), chat messages (Abbasi and Chen, 2008), and SMS messages (Ishihara, 2011).

Authorship attribution studies deal with author identification and similarity detection of texts. Identification involves comparing anonymous texts against those belonging to identified ones, where each anonymous text is written by one of those entities. Similarity detection involves comparing anonymous text with the known ones and assessing the degree of similarity, and thus attributing the unknown text to a potential author. Drawing parallels between biometrics, the former can be seen as biometric verification and the latter biometric authentication.

A related problem formulation is in the area of online privacy and anonymity (Narayanan et al., 2012) where the goal is to unmask an anonymous blogger or whistleblower. In this area of work, one piece of text is compared against every piece of writing within a large corpus of text whose authors are known. Authorship of the anonymous text is thus attributed to that of the most similar text in the corpus.

Plagiarism detection is yet another variant of authorship attribution where portions of writings are compared against large bodies of published writings, although this is more related to the use and arrangement of words than to other cognitive writing features (developed through mental experiences). Another direction of related study is authorship deception identification (Pearl and Steyvers, 2012), which deals in identifying an author if there is a suspected author who may be trying to anonymize his/her message or is actively imitating another author's writing in order to conceal his/her true identity (in biometrics this will be related to "spoofing"). This type of work is very relevant for cybercrime forensic investigation.

The different types of features used in authorship attribution (Abbasi and Chen, 2008) works include lexical, syntactic, structural, content features. Stylometric features have also been used for forensics and privacy assessment, and other features used include relative frequency words, character *n*-gram, word *n*-grams, part-of-speech *n*-grams, vocabulary richness, etc. The various classification algorithms used include naive Bayes, neural networks, K nearest neighbor, etc.

8 CONCLUSIONS AND FUTURE WORK

Very recently researchers have started looking at written language usage as a biometric trait (Fridman et al., 2013; Pokhriyal et al., 2014; Stolerman et al., 2014). Some of the cognitive modalities reported in the literature involve the use of biological signals captured through electrocardiograms, electroencephalograms, and electrodermal responses, to provide possible individual-authentication modalities (Faria et al., 2011). However, these are invasive and require users to have the electrodes placed on their specific body parts. It is an exciting prospect to investigate the use of language by people as a cognitive biometric trait, based on the previously reported psycholinguistic study (Pennebaker et al., 2003). Our biometrics analysis are performed on a very large corpus of real user data, having several thousands of authors and writings. In general, such large-scale studies are not typical in biometrics although are essential in order to transition biometric systems from the lab to real-life. Because we are evaluating language as a biometric modality, it is important to have a large-scale study such as this for complete results, since deductive results can only be obtained when large data is studied. This study also incorporates *big data into biometrics* where our dataset is characterized by high volume and high noise (veracity).

We conclude that language can indeed be used as a biometric modality, as it does hold some biometric fingerprint of the author. We report reasonable performance (72% AUC), even when the data consisted of unstructured blogs collected from across the Internet. Our study indicates that blogs provide a diverse and convenient way to study about authorship on the Internet. We found that better results are obtained with cleaner, high-quality texts. We found that if number of authors are known, than even few texts per author would suffice to build a good classifier. However, the accuracy of the classifier is independent of the number of authors for the study. We also performed stricter testing, where our classifier was to correctly classify an unseen author. When classified genuine authors 78% of the time, and impostors 76% of the time. Obviously these conclusions are data dependent, but provide an encouraging lead.

Regarding the issue of permanence, as long as the author maintains a specific writing style, this methodology will work. As our features are canonical in nature, they should be resistant to moderate changes in writing style and are expected to capture the variability in the nature of blogs. More work needs to be done to better understand permanence and spoof-ability. For the dataset used in this study, we verified that multiple persons have not authored with same author name. It is difficult to ensure in the blogs dataset, when a single person has written as multiple author names (he/she created profiles with different names). In that sense, our results are for the worst-case scenario.

The problem of author attribution can also be formulated as a multiclass classification approach, such that during testing, each blog has to be attributed to one of the known classes (authors). However, on an Internet-style author

attribution like ours, the number of authors is very large (order of 50 K in this case), and the text written by most of the authors is usually small (average is 4). Thus we get a large number of classes, with very limited number of data points for each class. A simplistic solution can be devised in which each author is characterized by a signature, which is obtained by combining the blogs written by that author. A new blog is then compared to all the available signatures and assigned to the author with most similar signature. Given that fusion of data instances of an enrollee into a signature is an open-area in biometrics, this is definitely an area of future work for our language biometrics paradigm.

An interesting extension to this research would be to work more closely with psycholinguistic community to investigate additional language-based features to more effectively capture the cognitive fingerprint of a person. With a large set of features to work with, we can employ feature selection algorithms to reduce the feature spaces and increase the area under the ROC curve. So far, we have only performed our evaluative study on Tier 1 and compared this with all the other tiers combined. However, there needs to be a more detailed study on the other tiers individually to see how the statistics regarding authorship attribution vary with the tiers. Also, is the blog of an author (how influential that author is) a measure of the personality of the author?

Since we take only the author name associated with a blog as the identity of the author—may be another statistic, such as email id could be taken so that we don't run into the problem when authors give themselves same pen-names. According to our algorithm all blogs written by such authors would be treated as written by one author, when in essence they are different.

To get all the blogs written by a particular author, we have to scan through the entire dataset once, which provides a considerable bottleneck as the entire dataset needs to be in memory, and also limits the amount of data that we can process. An alternative to this could be to assume that each blog is written by a different author, so that there is no need to read the entire dataset once, and it can be processed sequentially.

ACKNOWLEDGMENT

This work was supported by NSF Grant 1314803.

REFERENCES

Abbasi, A., Chen, H., 2005. Applying authorship analysis to arabic web content. In: Proceedings of the 2005 IEEE International Conference on Intelligence and Security Informatics, ISI'05, pp. 183–197.

Abbasi, A., Chen, H., 2008. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. ACM Trans. Inf. Syst. 26 (2), 7:1–7:29.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA

Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Burton, K., Java, A., Soboroff, I., 2009. The ICWSM 2009 Spinn3r dataset. In: Proceedings of the Third Annual Conference on Weblogs and Social Media (ICWSM 2009), San Jose, CA.

Carroll, D., 2007. Psychology of Language. Cengage Learning. ISBN 0495099694, 9780495099697.

de Vel, O., Anderson, A., Corney, M., Mohay, G., 2001. Mining e-mail content for author identification forensics. SIGMOD Rec. 30 (4), 55–64.

Faria, L., Sa, V., de Magalhaes, S., 2011. Multimodal cognitive biometrics. In: 6th Iberian Conference on Information Systems and Technologies (CISTI), 2011, pp. 1–6.

Fridman, A., Stolerman, A., Acharya, S., Brennan, P., Juola, P., Greenstadt, R., Kam, M., 2013. Decision fusion for multi-modal active authentication. In: IEEE IT Professional.

Gordon, A.S., Swanson, R., 2009. Identifying personal stories in millions of weblog entries. In: Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: an update. SIGKDD Explor. Newsl. 11 (1), 10–18.

Ishihara, S., 2011. A forensic authorship classification in SMS messages: a likelihood ratio based approach using N-gram. Int. J. Speech Lang. Law 21 (1), 47–56.

Koppel, M., Schler, J., 2003. Exploiting stylistic idiosyncrasies for authorship attribution. In: IJCAI03 Workshop on Computational Approaches to Style Analysis and Synthesis, pp. 69–72.

Koppel, M., Schler, J., Argamon, S., Messeri, E., 2006. Authorship attribution with thousands of candidate authors. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06, pp. 659–660.

Koppel, M., Schler, J., Bonchek-Dokow, E., 2007. Measuring differentiability: unmasking pseudonymous authors. J. Mach. Learn. Res. 8, 1261–1276. ISSN 1532-4435.

Koppel, M., Schler, J., Argamon, S., 2009. Computational methods in authorship attribution. J. Am. Soc. Inf. Sci. Technol. 60 (1), 9–26.

Koppel, M., Schler, J., Argamon, S., 2011. Authorship attribution in the wild. Lang. Resour. Eval. 45 (1), 83–94.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, New York, NY. ISBN 0521865719, 9780521865715.

McCallum, A.K., 2002. MALLET: a machine learning for language toolkit. URL http://mallet.cs.umass.edu.

Mealand, D.L., 1995. Correspondence analysis of luke. Lit. Ling. Comput. 10 (3), 171–182.

Mendenhall, T.C., 1887. The characteristic curves of composition. Science ns-9 (214S), 237–246.

Mosteller, F., Wallace, D., 1964. The Federalist: Inference and Disputed Authorship, Addison-Wesley Series in Behavioral Science Quantitative Methods. Addison-Wesley, University of Michigan, Ann Arbor, MI.

Narayanan, A., Paskov, H., Gong, N., Bethencourt, J., Stefanov, E., Shin, E., Song, D., 2012. On the feasibility of internet-scale author identification. In: IEEE Symposium on Security and Privacy (SP), 2012, pp. 300–314.

Pearl, L., Steyvers, M., 2012. Detecting authorship deception: a supervised machine learning approach using author writeprints. Lit. Ling. Comput. 27 (2), 183–196.

Pennebaker, J., Mehl, M., Niederhoffer, K., 2003. Psychological aspects of natural language use: our words, our selves. Ann. Rev. psychol. 54 (1), 547–577.

Pokhriyal, N., Nwogu, I., Govindaraju, V., 2014. Use of language as a cognitive biometric trait. In: IEEE International Joint Conference on Biometrics, Clearwater, IJCB, FL, USA, September 29-October 2, 2014, pp. 1–8.

Stamatatos, E., 2009. A survey of modern authorship attribution methods. J. Am. Soc. Inf. Sci. Technol. 60 (3), 538–556.

Stolerman, A., Fridman, A., Greenstadt, R., Brennan, P., Juola, P., 2014. Active linguistic authentication revisited: real-time stylometric evaluation towards multi-modal decision fusion. In: International Conference on Digital Forensics.

Thamar, S., Sangita, P., Sindhu, R., Manuel, M.G., 2011. Modality specific meta features for authorship attribution in web forum posts. In: Proceedings of 5th International Joint Conference on Natural Language Processing. Asian Federation of Natural Language Processing, pp. 156–164.

Yule, G., 1939. On Sentence-Length as a Statistical Characteristic of Style in Prose: With an Application to Two Cases of Disputed Authorship. Biometrika Office, University College, London.

Zhu, Y., Tan, T., Wang, Y., 2000. Biometric personal identification based on handwriting. In: Proceedings of the 15th International Conference on Pattern Recognition, 2000, vol. 2, pp. 797–800.

Chapter 4

Customer Selection Utilizing Big Data Analytics

Jungsuk Kwac[*,1], **Ram Rajagopal**[*,1]
[*] *Stanford Sustainable Systems Lab, Stanford University, Stanford, California, USA*
[1] *Corresponding authors: e-mail: kwjusu1@stanford.edu; ramr@stanford.edu*

ABSTRACT
Customer selection (or targeting) problem has been studied as a very important problem for a long time in the domain of sales and marketing. In the 2000s, as much more resources became available (e.g., various data sources, large databases, and improved computing power), new approaches based on data mining techniques have been tried. In this chapter, we try to expand the customer selection problem to a specific domain of demand response (DR) program targeting based on big data analytics using proper data mining techniques. To guarantee the scalability, we propose an efficient customer selection method via stochastic knapsack problem solving and a simple response modeling in one example DR program.

Keywords: Customer targeting, Demand response program, Algorithms, Big data, Smart meter data

1 INTRODUCTION

Traditionally, the customer selection (or targeting) problem has been studied as a very important problem for a long time in the domain of sales and marketing. In 2000s, much more resources became available (e.g., various data sources, large databases, improved computing power), and new approaches based on data mining techniques have been tried. For example, Chou et al. (2000) describe data mining techniques designed to address the problem of selecting prospective customers from a large pool of candidates, Shaw et al. (2001) have a good review of how data mining can be integrated into a knowledge-based marketing, Kim and Street (2004) propose a data mining approach to select an optimal target point where expected profit from direct mailing is maximized, Woo et al. (2005)

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00004-6

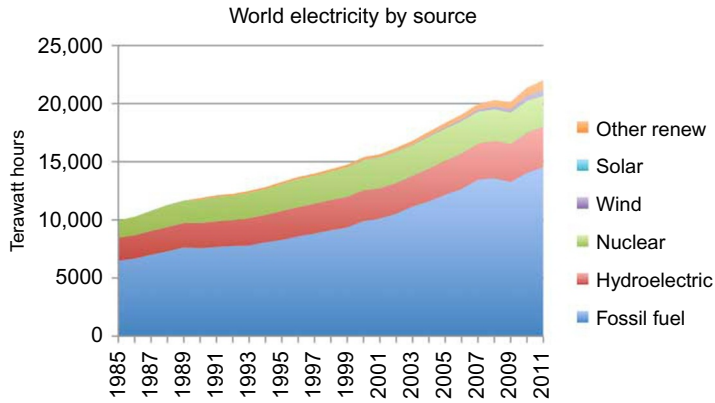


FIGURE 1 Electricity consumption trend (http://www.financialsense.com/).

suggest a "customer map," which is the visualization method for customer targeting based on data mining techniques, and Lawrence et al. (2007) describe the analytics-based solutions to identify new sales opportunities and to estimate future revenue opportunities.

In this chapter, we try to expand the customer selection problem to a specific domain of demand response (DR) program targeting based on big data analytics using proper data mining techniques. Now, we provide the reason why we want to address customer selection problems in DR program targeting with explaining what is DR program.

It is well known that the worldwide electricity consumption has been continuously increased as shown in Fig. 1. To meet this increasing electricity demand, there have been several approaches. From the perspective of supply, more power generation facilities and more transmission equipments have been built. Also, with increasing concerns on the sustainability of energy supply, renewable energy resources have been actively studied, and their proportion has been increased. From the perspective of demand management, various energy efficiency (EE) programs and DR programs have been designed and executed. To compare EE and DR briefly, EE focuses on using less power to perform the same tasks on a continuous basis (e.g., promoting appliance replacements for better EE), while DR focuses on reducing the temporary power consumption when system reliability is expected to be jeopardized (e.g., incentivizing to reduce the consumption for system peak time).

For both EE and DR programs, communications with customers are inevitable, and certain incentives, equipment installations, or maintenances are required, which implies that utilities cannot try every customer with limited resources (budgets, equipments, etc). Thus, customer targeting is very important to achieve better performance of energy programs within allowed conditions.

1.1 Prior Work

Similar to the history in the marketing area, much of the previous research on customer segmentation in the energy domain was based on the surveys of individuals regarding their self-reported values, attitudes, knowledge, and behaviors (Sanquist et al., 2012; Sütterlin et al., 2011). The recent widespread deployment of advanced metering infrastructure has made available concrete information about user consumption from smart meters, and data mining or machine learning techniques started to be applied on customer characterization and segmentation, e.g., Figueiredo et al. (2005) and Kwac et al. (2014).

Regarding the previous studies on DR programs, there is significant literature addressing DR (e.g., Albadi and El-Saadany, 2007; Borenstein et al., 2002; Han and Piette, 2008; Rahimi and Ipakchi, 2010; Sanquist et al., 2012), including definitions, design of DR programs, and estimation of energy savings. However, to the best of our knowledge, there is no research focused on targeting customers for DR programs using their temporal consumption data from smart meters except Kwac and Rajagopal (2015). Currently, most DR program targeting relies on segmentation of customers based on their monthly billing data or surveys (Lutzenhiser, 2009; Moss et al., 2008). The availability of smart meter data has shown that such approaches are very inaccurate since segments obtained from actual consumption differ from those obtained by alternative methods (Kwac et al., 2014; Wong and Rajagopal, 2012).

1.2 Goal

Figure 2 shows the smart meter penetration status of United States by 2014. According to U.S. EIA (Energy Information Administration), 43,165,185 smart

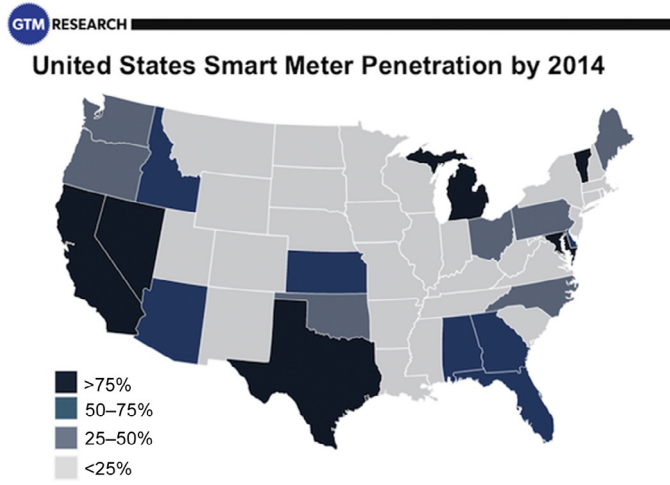


FIGURE 2 United States Smart Meter Penetration by 2014. *Source: From http://www.greentechmedia.com/.*

meters have been installed by 2012. Considering that every smart meter records hourly or 15-min interval consumption, the size of the entire smart meter data is huge. Thus, to select a relatively small number of proper customers for DR program targeting, the methodology must guarantee the scalability to be applied to the large dataset. Here, using relevant data mining techniques, we show how to solve a customer selection problem for a specific DR program targeting with guaranteeing the scalability of the methodology proposed.

2 METHODOLOGY

In this chapter, we investigate how to select customers for a certain DR program relying on big data analytics. The used data is hourly consumption data generated from smart meters for residential customers. We provide a customer response (energy saving) model for an air-conditioning (AC) controlling DR program, which estimates the energy saving potential of each customer during DR events. Also, we propose how to select appropriate customers from a large population using the estimated response information with appropriately chosen constraints. Figure 3 shows the overall customer selection flow proposed in this chapter with brief computational complexity information.

2.1 Response Modeling

We illustrate here a simple linear model to estimate the response distribution for each customer in a specific DR program which controls AC system. According to the DR strategy list in Han and Piette (2008), it belongs to "Global Temperature Adjustment," which is a very effective strategy and can be used in all AC systems. Basically, the working mechanism is to increase the temperature setpoint for each individual's AC system by some amount and achieve energy saving from reduced AC power consumption.

To estimate each customer's energy consumption decrease occurring from the increased temperature setpoint, we need an energy consumption model which can capture the energy consumption of AC systems. If AC power consumption is independently metered and indoor temperature is recorded, a simple model can be built based on those observed data. In general, however, only total house-level consumption and external temperature are available. From

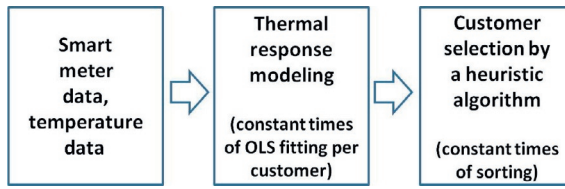


FIGURE 3 Overall flow of the proposed customer selection methodology.

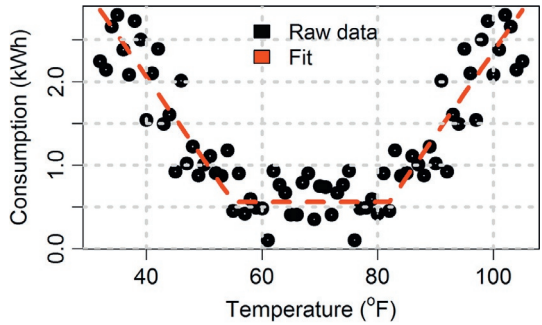


FIGURE 4 The example of a piecewise linear model (two breakpoints).

previous study (Han and Piette, 2008), it is known that the main proportion of temperature-sensitive energy consumption is from HVAC (heating, ventilation, and air conditioning) systems. From Mathieu et al. (2011), Birt et al. (2012), and Cancelo et al. (2008), the relationship between energy consumption and outside temperature is piecewise linear with two or more breakpoints as shown in Fig. 4. The reasoning behind the piecewise linear models is that, if the temperature is lower than some point, people start to use their heating system, and if the temperature is higher than some point, people use their cooling system.

Based on these facts, we provide a simple piecewise linear model with one breakpoint for 4–6 PM (usual system peak hours) during the summer season. The hourly power consumption of a customer k at time t on day d, $l(k,t,d)$, is modeled as

$$l(k,t,d) = a(k,t)(T\_ext(k,t,d) - T\_ref(k)) + b(k,t)(T\_ref(k) - T\_ext(k,t,d)) + c(k,t) + e(k,t), \quad (1)$$

where $T\_ext(k,t,d)$ is the outside temperature, $T\_ref(k)$ is the breakpoint which is the proxy of the temperature setpoint for the customer k's HVAC system, $a(k,t)$ is the cooling sensitivity, $b(k,t)$ is the heating sensitivity (or the temperature sensitivity before turning on AC system), $c(k,t)$ is base load, and $e(k,t)$ is a random variability. The reason why T_ref is a function of k, not both k and t, is that we assume each customer has his or her own temperature setpoint which does not change at least over the concerned hours.

In the model (1), we have to estimate $a(k,t)$, $b(k,t)$, $c(k,t)$, and $T\_ref(k)$ by least squares estimation. If $T\_ref(k)$ is provided, the other parameters can be obtained by ordinary least squares (OLS) estimates. However, $T\_ref(k)$ is also a variable to estimate. If we try to find the best $T\_ref(k)$ among all possible temperatures, it is too expensive from the perspective of computational cost and not scalable to big datasets. Thus, we put some constraints on $T\_ref(k)$ that it should be an integer and in the range 68–86 °F, which is typical. Additionally, to prevent the cases that one or two dominant data points determine the breakpoint and provide invalid temperature sensitivity (which is too high) as shown in Fig. 5,

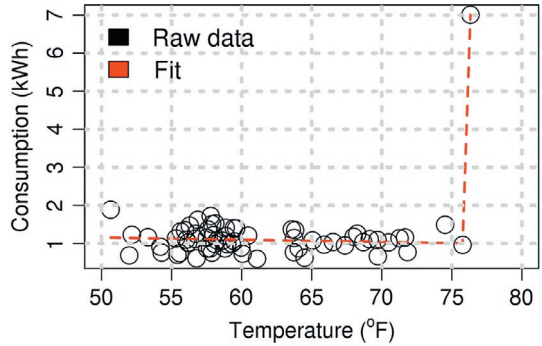


FIGURE 5 Invalid fitting example by one dominant point.

we put another constraint on the breakpoint that both sides of the breakpoint should have at least certain fraction of data (e.g., we set 15%). Then, fitting the model (1) for each customer consists of constant times (at most 19 times) of OLS estimation and finding the best parameters with minimum sum of square errors.

Additionally, we need one more step to avoid overfitting of the proposed model, which means there may not exist a breakpoint. In that case, the model changes to below,

$$l(k,t,d) = a(k,t)T\_ext(k,t,d) + c(k,t) + e(k,t), \tag{2}$$

which is nested by the model (1). Thus, we run one more linear regression and can check the overfitting by F-test. Overall, to find the cooling sensitivity per hour for each customer, it requires at most 20 times of OLS estimation.

Figure 6 shows one example of the power consumption model (1) fitting results. The breakpoint is 82 °F, and the red (gray in the print version) dotted slope after 82 °F corresponds to the estimate of $a(k,t)$, $\hat{a}(k,t)$. If a DR event increases the temperature setpoint by 3 °F, it is the same as moving the breakpoint to the right side by the same amount, 3 °F, as shown in Fig. 7.
Then, the expected energy consumption in the DR event is following the blue (dark gray in the print version) dotted line in Fig. 7. From here, the customer k's response (energy saving) at time t, $r(k,t)$, is the difference between the red (gray in the print version) dotted line and the blue (dark gray in the print version) dotted line, which can be calculated by

$$r(k,t) = a(k,t)^* \Delta T\_ref(k), \tag{3}$$

where $\Delta T\_ref(k)$ is the temperature setpoint increase.

Given that we know the estimate of $a(k,t)$, $\hat{a}(k,t)$, and its standard deviation $\sigma(a(k,t))$, $r(k,t)$ has the mean $\mu(k,t) = \hat{a}(k,t)^* \Delta T\_ref(k)$ and the standard deviation $\sigma(k,t) = \sigma(a(k,t))^* \Delta T\_ref(k)$. Once we calculate these estimates for

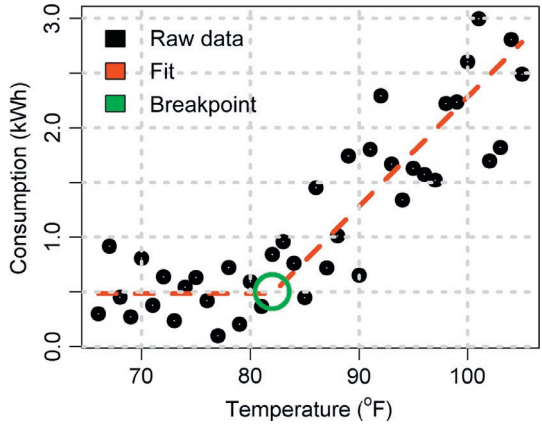


FIGURE 6 Before increasing the temperature setpoint.

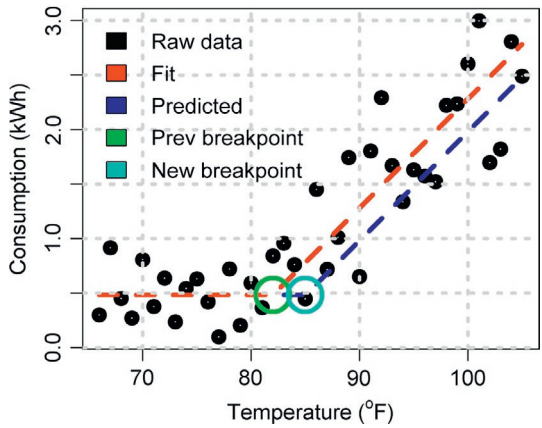


FIGURE 7 After increasing the temperature setpoint.

every customer, we can proceed to customer selection process based on this response distribution information.

To summarize, to estimate the energy saving occurring from temperature setpoint increase, we fit a piecewise linear model (hourly model) with one breakpoint for concerned hours during the summer period. By putting certain constraints on fitting parameters, we guarantee the scalability of this response model fitting (constant times of OLS per each customer) and provide the response distribution information to the next step: customer selection. Note that we do not argue that the response modeling explained here is the best response modeling for a given DR program. What we are trying to show is how to build scalable response modeling with proper reasoning.

2.2 Customer Selection

From the response modeling process, we know the means and the variances of customer responses. Given this information, we develop a scalable customer selection algorithm that provides utility managers the trade-off between DR reliability (the probability of achieving a certain energy saving) and the DR program cost (proportional to the number of customers engaged) given aggregate target response (energy saving).

2.2.1 Customer Selection Problem Setting

Let's assume there are K candidate customers who can be recruited for the explained DR program. Given the mean and the variance of each customer's response for a DR event, we want to select a small number of appropriate of customers from the entire candidates, who can provide large energy saving (high DR availability) with low risk (high DR reliability). On the assumption that the DR event time t is set, we represent the response distribution parameters as below (Table 1).

Then, we propose the optimization problem to select customers below.

$$\max_x P\left(\sum_{k=1}^{K} r_k x_k \geq T\right), \tag{4}$$

$$\text{s.t.} \sum_{k=1}^{K} x_k \leq N,$$

$$x_k \in \{0, 1\}\, \forall\, k,$$

where T is the aggregate target response which is a design parameter, and N is the limit of the number of participating customers which may be set by the budget limit. In this problem, x_k stands for the indicator showing whether the customer k is selected or not, and x is the vector of x_k.

The interpretation of this optimization problem is that we are trying to find the best combination of customers to maximize the probability of achieving

TABLE 1 Response Distribution Parameter Notations

$r(k,t) : r_k, \mu(k,t) : \mu_k, \sigma(k,t) : \sigma_k$
$\boldsymbol{\mu} = \{\mu_1, \ldots, \mu_k\}, \quad \boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_k\}$
$\Sigma =$ the covariance matrix of responses
$\Sigma_{i,i} = \sigma_i^2, \Sigma_{j,k} = \text{COV}(r_j, r_k)$

the aggregate target response under the limit of the number of participating customers. The limit of the number of participating customers should come from the budget constraint as each participating customer will incur a certain cost for the DR program operator. In other words, this optimization problem maximizes the DR reliability within the DR program budget. In this problem, the DR availability is represented by T. By varying T with fixed N, the selected customers and the maximum probability will change together.

Basically, finding the best combination of entities from a large population is a combinatorial optimization problem. Moreover, in this case, the response r_k of each customer is a random variable with known mean and standard deviation so that this problem (4) is a stochastic knapsack problem (SKP). Here, we provide a short explanation about a knapsack problem (KP) and a SKP. The KP is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. In the definition of the KP, if the value of each item is a random variable with known distribution, then the problem becomes a SKP. From the computational complexity perspective, both KP and SKP are generally NP-hard (nondeterministic polynomial-time hard).

2.2.2 The Optimization Problem Transformation

Here, we show how the original optimization problem in (4) can be changed to a different form with a proper assumption. We assume that K is very large as the total number of customers is huge (e.g., PG&E has more than 5 million customers) and N is sufficiently large (at least larger than a hundred). Then, by the central limit theorem, the sum of the responses ($\Sigma r_k x_k$) in (4) for the selected customers can be approximated to follow a Gaussian distribution below

$$\sum_{k=1}^{K} r_k x_k \sim N(\boldsymbol{\mu}^T \boldsymbol{x}, \boldsymbol{x}^T \Sigma \boldsymbol{x}). \tag{5}$$

Then, the problem (4) is equivalent to the problem below

$$\max_x P\left(\sum_{k=1}^{K} r_k x_k \geq T\right) \Leftrightarrow \min_x \frac{T - \boldsymbol{\mu}^T \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T \Sigma \boldsymbol{x}}}. \tag{6}$$

And, we define $\rho(\boldsymbol{x})$ and ρ^* like below

$$\rho(\boldsymbol{x}) = \frac{T - \boldsymbol{\mu}^T \boldsymbol{x}}{\sqrt{\boldsymbol{x}^T \Sigma \boldsymbol{x}}}, \quad \rho^* = \min_x \rho(\boldsymbol{x}). \tag{7}$$

In this transformed SKP (6), the optimization direction depends on the optimal value of the problem. Note that ρ^* is the optimal value of the transformed

problem. The sign of ρ^* is determined by the numerator, $T - \boldsymbol{\mu}^T \boldsymbol{x}$, whether it can be larger than 0 or not. (Knowing the sign of ρ^* is very easy. If the sum of N largest μ_k is larger than T, the sign is negative, which means we can set the sign of ρ^* by setting the design parameter T properly.) If ρ^* is larger than 0, we need to select the customers whose response mean is large and the variance is also large. This corresponds to wishing a luck to get high energy saving from a small number of customers by selecting those with high variation. However, if ρ^* is less than 0, we have to select the customers whose response mean is large and the variance is small. Practically, this is the right approach as it corresponds to maximizing the response with minimizing the risk. From these facts, the optimization problem should be solved differently depending on the sign of ρ^*, though it can be determined from the problem setting. Different solving approaches depending on the sign of ρ^* will be addressed later in the proposed heuristic algorithm explanation.

2.2.3 Previous Approaches to Solve the SKP

As explained, the proposed customer selection problem (4) is a SKP of which the optimal solution cannot be obtained within a polynomial-time from the perspective of the computational complexity. Here, we provide the previous approaches to solve the transformed SKP problem approximately.

When ρ^* in (7) is less than 0, the optimal solution for the SKP can be obtained by solving the equation below multiple times for various λ (Geoffrion, 1967; Henig, 1990)

$$\lambda \boldsymbol{\mu}^T \boldsymbol{x} - (1 - \lambda) \boldsymbol{x}^T \Sigma \boldsymbol{x}, \quad (0 \leq \lambda \leq 1).$$

Here is a brief sketch why this approach can obtain the optimal solution.

How to set various λ is related to nonlinear optimization techniques referred in Henig (1990). According to Henig (1986), the expected complexity of finding the extreme points by setting various λ is $O((KCN)^2)$, which is corresponding to square times of finding every combination of N customers among K customers. This is very heavy computation cost when the number of customers, K, is large. Another approach in Morton and Wood (1998) is based on a dynamic programming (DP) method which requires solving $\mathrm{O}(N \max({\sigma_s}^2))$ times of integer linear programs with dimension K, obtained by discretizing the range of ${\sigma_s}^2$ and assuming the covariance matrix is diagonal.

2.2.4 A New Heuristic Algorithm to Solve a SKP

To find the extreme points in $\boldsymbol{H} = \{(\mu_s, \sigma_s^2) : \mu_s > \mathrm{T}, \sigma_s^2 > 0\}$, we propose an efficient heuristic algorithm below.

By the monotonicity property shown in Table 2, the maximum should be obtained from one of the extreme points in the right bottom area of $\boldsymbol{H}$, e.g., the

TABLE 2 A Brief Sketch of How to Obtain the Optimal Solution

Set $\mu_s = \boldsymbol{\mu}^T\boldsymbol{x}, \sigma_s^2 = \boldsymbol{x}^T\Sigma\boldsymbol{x}$, $\rho(\boldsymbol{x}) = \rho(\mu_s, \sigma_s^2) = \dfrac{T-\mu_s}{\sigma_s}$
Then, $\rho(\mu_s, \sigma_s^2)$ is pseudoconcave over $\{(\mu_s, \sigma_s^2) : \mu_s > T, \sigma_s^2 0\}$ by Lemma 1 of Henig (1990).
As shown below, the gradient of $\rho(\mu_s, \sigma_s^2)$ is monotonic decreasing in μ_s and increasing in σ_s^2.
$\overline{\nabla}\rho(\mu_s, \sigma_s^2) = \left(\dfrac{-1}{\sigma_s}, \dfrac{\mu_s - T}{2\sigma_s^3}\right), \quad \left(\dfrac{-1}{\sigma_s} < 0, \dfrac{\mu_s - T}{2\sigma_s^3} > 0\right)$
If $\boldsymbol{H}$ is the convex hull of $\{(\mu_s, \sigma_s^2)\}$, by the monotonicity property shown above, the optimal solution should be obtained from one of the extreme points in the convex hull (Bazaraa et al., 1979). From the gradient information, μ_s must be as large as possible, and σ_s^2 as small as possible.
Thus, to find all the eligible extreme points, the problem below should be solved multiple time with various λ, which corresponds to finding the point of contact with a slope λ'.
$\arg\max_{\mu_s, \sigma_s^2} \lambda\mu_s - (1-\lambda)\sigma_s^2, (0 \le \lambda \le 1) \Leftrightarrow \arg\max_{\mu_s, \sigma_s^2} \lambda'\mu_s - \sigma_s^2, \left(\lambda' \ge 0, \lambda' = \frac{\lambda}{1-\lambda}\right)$
After finding all extreme points, we select the best (μ_s, σ_s^2) minimizing $\rho(\mu_s, \sigma_s^2)$.

red (gray in the print version) area shown in Fig. 8. Depending on the positive slope, λ', different extreme points will be obtained. Thus, the heuristic we propose here is trying to find the extreme points by finding a contact point for a given positive slope which increases by the same angle, $\pi/2M$. Once $M+1$ extreme points (some can be duplicate) in the red (gray in the print version) area of Fig. 8 and corresponding $\boldsymbol{x}$ vectors are found, we select the best $\boldsymbol{x}$ vector which minimizes $\rho(\boldsymbol{x})$. Regarding the approximation bound of this heuristic algorithm, it is depending on the data, and the proof is provided in Kwac and Rajagopal (2015).

When ρ^* in (7) is larger than 0, the optimal solution for the SKP may not be one of the extreme points (Henig, 1990). However, we approximate that the optimal solution stays at one of extreme points and find the extreme points via solving (9) instead of (8) in Algorithm 1 because the optimization direction is increasing μ_s and σ_s^2 as much as possible when ρ^* is larger than zero (Table 3). This is trying to find all the extreme points (contact points with various negative slopes) in the blue (gray in the print version) area in Fig. 8.

Now, we explain how the computational complexity of this heuristic algorithm is constant times of sorting. When we solve the subproblem in (8) (the similar way in (9)), if we assume that the customer response is independent (the covariance matrix is diagonal), then the subproblem changes like below.

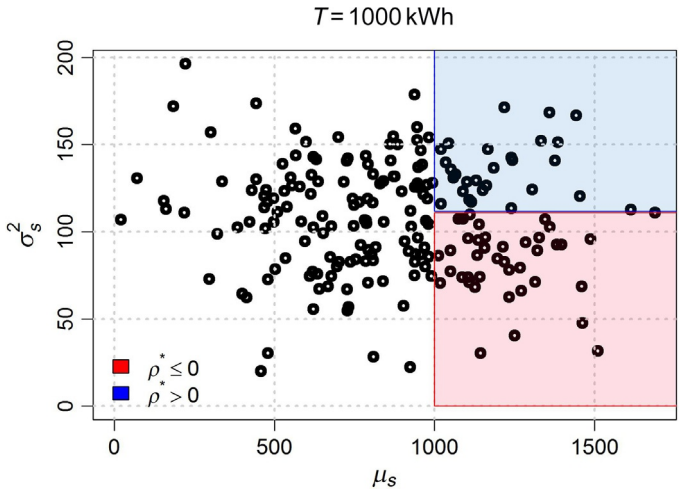


FIGURE 8 The area to find extreme points depending on the sign of ρ^* when $T = 1000$ kWh.

TABLE 3 Algorithm 1 to Solve the SKP

Require: $\boldsymbol{\mu}$ and Σ from response modeling.

Set a integer constant, M (= the number of iterations).

for i from 0 to M **do**

$\lambda_i' = \tan\left(\frac{i\pi}{2M}\right)$.

Solve the (sub) problem below and save $\mathbf{x_i}$:

$$(\text{If} \rho^* \leq 0)\ \mathbf{x_i} = \arg\max_x \lambda_i' \boldsymbol{\mu}^T \boldsymbol{x} - \boldsymbol{x}^T \Sigma \boldsymbol{x}, \tag{8}$$

$$(\text{If} \rho^* > 0)\ \mathbf{x_i} = \arg\max_x \lambda_i' \boldsymbol{\mu}^T \mathbf{x} + \mathbf{x}^T \Sigma \mathbf{x}, \tag{9}$$

$$\text{s.t.} \sum_{k=1}^{K} x_k \leq N,$$

$$x_k \in \{0, 1\}\ \forall k.$$

end for

Return $\bar{\text{x}} = \arg \min_{\mathbf{x} \in \{\mathbf{x}_0, \ldots, \mathbf{x}_M\}} \frac{T - \boldsymbol{\mu}^T \mathbf{x}}{\sqrt{\mathbf{x}^T \Sigma \mathbf{x}}}$.

$$\lambda_i' \boldsymbol{\mu}^T \boldsymbol{x} - \boldsymbol{x}^T \Sigma \boldsymbol{x} = (\lambda_i' \boldsymbol{\mu} - \boldsymbol{\sigma}^2)^T \boldsymbol{x}, \tag{10}$$

where $\boldsymbol{\sigma}^2$ is the vector of customers' response variances (diagonal elements of covariance matrix, Σ). Then, maximizing this equation with the constraint that the sum of $\mathbf{x}$ is equal or less than N is the same with selecting the highest (at most) N values from $(\lambda_i' \boldsymbol{\mu} - \boldsymbol{\sigma}^2)$ vector, which are larger than 0. This can be easily solved by sorting the K values of $(\lambda_i' \boldsymbol{\mu} - \boldsymbol{\sigma}^2)$ vector, which is $O(K \log K)$ in computational complexity.

2.2.5 Greedy Algorithm to Solve a SKP

To evaluate the performance of the proposed heuristic algorithm, we propose a simple greedy algorithm to solve the SKP. If we assume Σ is diagonal, and $T = 0$, then the transformed problem (6) is solved by sorting the customers by $\mu k/\sigma k$ in descending order. This greedy algorithm is similar with the greedy approximation algorithm to solve the unbounded KP in Dantzig (1957) and Dean et al. (2004). However, the algorithm is arbitrarily poor when $T > 0$ and the aggregate response of the first N customers does not exceeds T. Gradual greedy algorithm in Table 4 accounts for this issue and achieves at least more than 50% probability when $\rho^* < 0$.

Note that there are several reasons in that we compare the performance of the proposed heuristic with only that of the greedy algorithm here.

1. Fair comparison: For the algorithms which guarantee the optimal with heavy computation cost, there is no need to compare. For other fast algorithms based on DP, they require specific assumptions (e.g., diagonal and integer covariance matrix). Depending on how to discretize the variances, their performances will be different, which is not clear to find a fair comparison point.
2. Ease of comparison: Greedy algorithms have been used long time and are easy to understand. They do not need any specific assumptions to apply.

TABLE 4 Gradual Greedy Algorithm

Require: $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ vector from (10).
$\mathbf{x} = 0$, $T_0 = T$.
for i from 1 to N **do**
Find j from below and set $x_j = 1$ in $\boldsymbol{x}$:
$j = \arg \max_{\{k \mid x_k = 0\}} \frac{\mu_k}{\sigma_k}$ (when $\rho^* \leq 0$, s.t. $\mu_k \geq \frac{T_{i-1}}{N+1-i}$).
$T_i = T_{i-1} - \mu_j$
End for
Return $\boldsymbol{x}$

3 EXPERIMENTS

3.1 Data Description

The anonymized smart meter data used in this chapter is provided by Pacific Gas and Electric Company (PG&E). The data contains the electricity consumption of residential PG&E customers at 1-h intervals. There are 218,090 smart meters, and the total number of 24-h load profiles is 66,434,179. The data corresponds to 520 different zip codes and covers climate zones 1, 2, 3, 4, 11, 12, 13, and 16 according to the California Energy Commission climate zone definition (shown in Fig. 9). The targeting methodology is tested by focusing on a cool climate zone (Zone 3: 32,339 customers) and a hot climate zone (Zone 13: 25,954 customers) during the summer season (May to July 2011). Zone 3 is a coastal area and Zone 13 an inland area. Weather data corresponding to the same period is obtained from Weather Underground for each zip code in these two zones.

3.2 Response Modeling Result

We first check the quality of the consumption model fits by verifying the distribution of the R^2 values for each climate zone as shown in Fig. 10. In hot

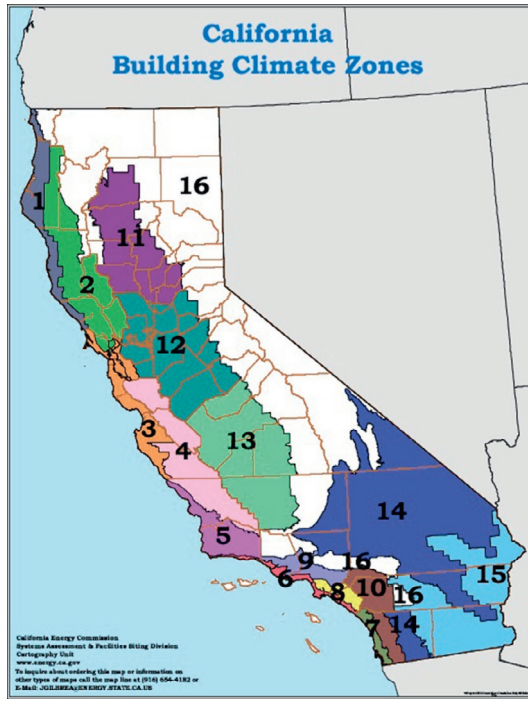


FIGURE 9 California Energy Commission climate zone definition (http://www.energy.ca.gov/).

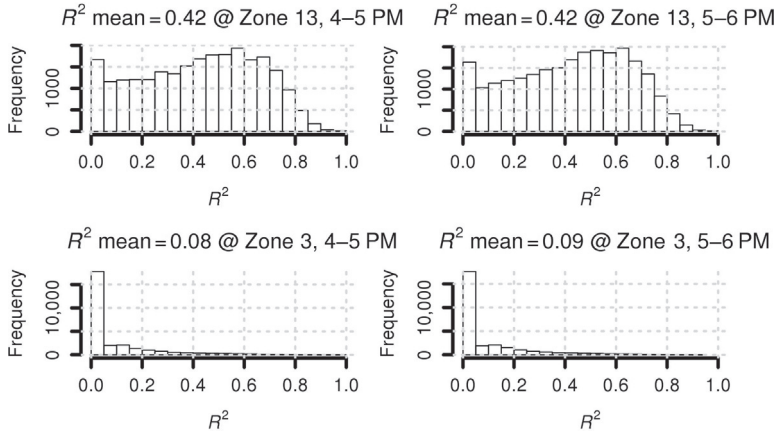


FIGURE 10 Consumption model fitting evaluation: R^2 distribution.

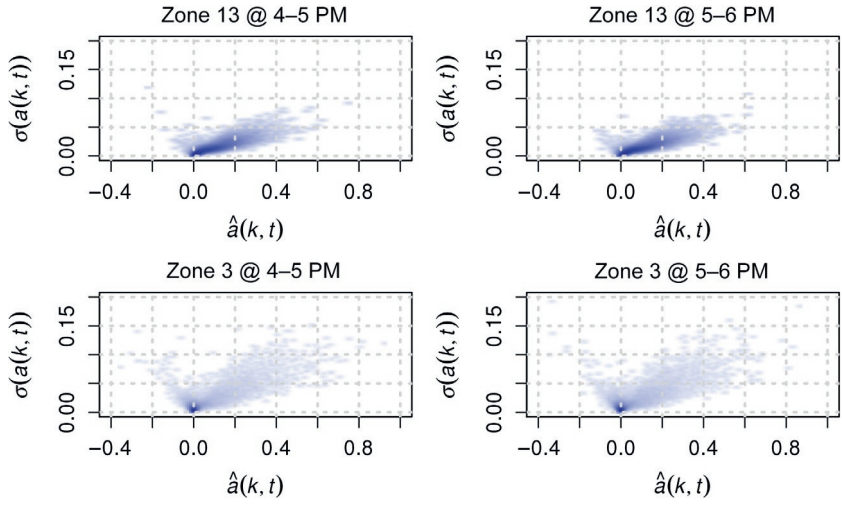


FIGURE 11 Scatter plot of cooling sensitivity estimate and standard deviation.

area, the model explains about 42% of the variance on average, while it only explains 9% in cool area. This means that the temperature is not very important factor of energy consumption in Zone 3 (cool area).

Figure 11 shows the smoothed scatter plots of the estimate and the standard deviation of each customer's cooling sensitivity in both climate zones. As shown in the figure, most customers in Zone 3 have very small cooling sensitivities. Only a small fraction of customers are scattered widely over the plot. Meanwhile, most of customers in Zone 13 have positive cooling sensitivities and their standard deviations are relatively small.

3.3 Customer Selection Result

Figure 12 shows the DR reliability curves with increasing budget limit (corresponding to N). Each black point in the curves is obtained from the proposed optimization problem solving by the proposed heuristic algorithm. The red (gray in the print version) points are obtained by the gradual greedy algorithm explained in Table 4. We assume that DR events happen during 4–6 PM, and they require the temperature setpoint increased by 3 °F. The aggregate response target T is 1500 kWh. As shown in the figure above, the heuristic algorithm always achieves equal or higher probability than the greedy algorithm. As designed, the greedy algorithm guarantees more than 50% of probability when it is possible.

As shown in Fig. 12, a smaller number of customers are needed in Zone 13 to guarantee aggregate response 1000 kWh. Also, note the probability (DR reliability) changes from 0% to 100% within about 40 customers in the hot climate Zone 13, though it needs about 3300 customers to achieve 1500 kWh of energy saving. This indicates that an inappropriate number of enrollments can cause a program to fail, reinforcing the need for the analytic targeting mechanism. However, in the cool climate Zone 3, it takes more than 250 customers to move from probability 0% to 100%. Figure 11 supports all these statements. From Fig. 11, we saw that most customers in Zone 3 have small cooling sensitivities, which resulted more customers required for the same energy saving. Also, the customers with large responses (large cooling sensitivities) in Zone 13 showed relatively small standard deviation compared to the customers in Zone 3, which

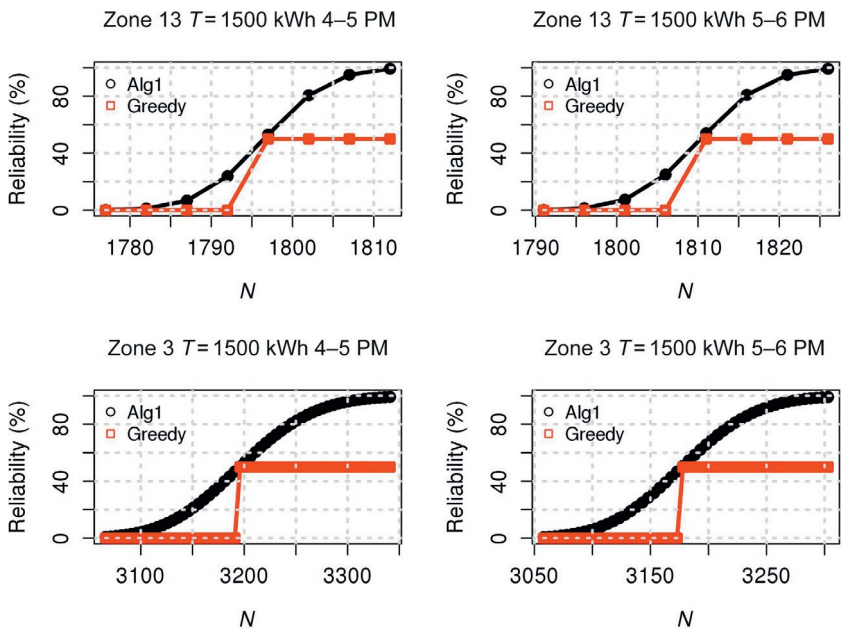


FIGURE 12 Trade-off between DR reliability and DR program cost.

resulted in a smaller number of customers required to increase the reliability from 0% to 100%.

4 CONCLUSION

In this chapter, we showed how to select the proper customers for a specific DR program via big data analytics. For the first step, we developed a simple response modeling by putting certain constraints on the model fitting to guarantee the scalability to a large data set (the computation cost per each customer: 20 times of OLS estimation). Then, we proposed an optimization problem setting to select the appropriate customers from a large population, which is a special case of SKP. To solve this SKP, we developed a fast heuristic algorithm which requires sorting K values constant times. Note that the proposed heuristic algorithm does not have to be confined to the DR program targeting explained in this chapter. For any type of DR program or EE program, if we can estimate the mean and variance of each customer response, this heuristic algorithm can be applied with guaranteeing low computational cost.

As an example DR program to show the result of our customer selection methodology, we suggested a DR program controlling the temperature set-point during the summer season. From the experiment results, it is found that the outside temperature does not explain much about customers' electricity consumption in the cool climate Zone 3. Also, we found that most customers in Zone 3 have a very small temperature sensitivity. Therefore, with a limited budget, targeting some customers in the hot climate zones may be more efficient on the assumption that all the selected customers comply well to the DR program. However, the high consumption customers with high mean response may be relatively rich and may not feel the need much to sacrifice their comforts for some incentives. In other words, our customer selection methodology shows the optimistic energy saving expectation result from a pure technical perspective on the assumption of 100% customer compliance. Thus, to enhance the customer targeting performance in practice, our customer selection method should be integrated with the approaches from psychological and behavior science perspectives.

REFERENCES

Albadi, M.H., El-Saadany, E.F., 2007. Demand response in electricity markets: an overview. In: IEEE Power Engineering Society General Meeting, vol. 2007. pp. 1–5.

Bazaraa, M.S., Sherali, H.D., Shetty, C.M., 1979. Nonlinear Programming: Theory and Algorithms. John Wiley & Sons.

Birt, B.J., Newsham, G.R., Beausoleil-Morrison, I., Armstrong, M.M., Saldanha, N., Rowlands, I.H., 2012. Disaggregating categories of electrical energy end-use from whole-house hourly data. Energy Build. 50, 93–102.

Borenstein, S., Jaske, M., Rosenfeld, A., 2002. Dynamic pricing, advanced metering, and demand response in electricity markets. Center for the Study of Energy Markets.

Cancelo, J.R., Espasa, A., Grafe, R., 2008. Forecasting the electricity load from one day to one week ahead for the Spanish system operator. Int. J. Forecast. 24 (4), 588–602.

Chou, P.B., Grossman, E., Gunopulos, D., Kamesam, P.V., 2000. Prospective customer selection using customer and market reference data. U.S. Patent No 6,061,658, Washington, DC.

Dantzig, G.B., 1957. Discrete-variable extremum problems. Oper. Res. 5 (2), 266–288.

Dean, B.C., Goemans, M.X., Vondrdk, J., 2004. Approximating the stochastic knapsack problem: the benefit of adaptivity. In: Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science.

Figueiredo, V., Rodrigues, F., Vale, Z., Gouveia, J.B., 2005. An electric energy consumer characterization framework based on data mining techniques. IEEE Trans. Power Systems 20 (2), 596–602.

Geoffrion, A.M., 1967. Solving bicriterion mathematical programs. Oper. Res. 15 (1), 39–54.

Han, J., Piette, M.A., 2008. Solutions for summer electric power shortages: demand response and its applications in air conditioning and refrigerating systems. Lawrence Berkeley National Laboratory.

Henig, M.I., 1986. The shortest path problem with two objective functions. Eur. J. Oper. Res. 25 (2), 281–291.

Henig, M.I., 1990. Risk criteria in a stochastic knapsack problem. Oper. Res. 38 (5), 820–825.

Kim, Y., Street, W.N., 2004. An intelligent system for customer targeting: a data mining approach. Decis. Support Syst. 37 (2), 215–228.

Kwac, J., Flora, J., Rajagopal, R., 2014. Household energy consumption segmentation using hourly data. IEEE Trans. Smart Grid 5 (1), 420–430.

Kwac, J., Rajagopal, R., 2015. Data-driven targeting of customers for demand response. IEEE Trans. Smart Grid (under review).

Lawrence, R., Perlich, C., Rosset, S., Arroyo, J., Callahan, M., Collins, J.M., Weiss, S.M., 2007. Analytics-driven solutions for customer targeting and sales-force allocation. IBM Syst. J. 46 (4), 797–816.

Lutzenhiser, L., 2009. Behavioral assumptions underlying California residential energy efficiency programs. In: CIEE Energy and Behavior Program.

Mathieu, J.L., Price, P.N., Kiliccote, S., Piette, M.A., 2011. Quantifying changes in building electricity use, with application to demand response. IEEE Trans. Smart Grid 2 (3), 507–518.

Morton, D.P., Wood, R.K., 1998. On a Stochastic Knapsack Problem and Generalizations. Springer, USA, pp. 149–168.

Moss, S.J., Cubed, M., Fleisher, K., 2008. Market Segmentation and Energy Efficiency Program Design. California Institute for Energy and Environment, Berkeley.

Rahimi, F., Ipakchi, A., 2010. Demand response as a market resource under the smart grid paradigm. IEEE Trans. Smart Grid 1 (1), 82–88.

Sanquist, T.F., Orr, H., Shui, B., Bittner, A.C., 2012. Lifestyle factors in US residential electricity consumption. Energy Policy 42, 354–364.

Shaw, M.J., Subramaniam, C., Tan, G.W., Welge, M.E., 2001. Knowledge management and data mining for marketing. Decis. Support Syst. 31 (1), 127–137.

Sütterlin, B., Brunner, T.A., Siegrist, M., 2011. Who puts the most energy into energy conservation? A segmentation of energy consumers based on energy-related behavioral characteristics. Energy Policy 39 (12), 8137–8152.

Wong, J., Rajagopal, R., 2012. A simple way to use interval data to segment residential customers for energy efficiency and demand response program targeting. In: ACEEE Proceedings.

Woo, J.Y., Bae, S.M., Park, S.C., 2005. Visualization method for customer targeting using customer map. Expert Syst. Appl. 28 (4), 763–772.

Chapter 5

Continuous Model Selection for Large-Scale Recommender Systems

Simon Chan[*,1], **Philip Treleaven**[*,1]

[*] *Department of Computer Science, University College London, London, United Kingdom*

[1] *Corresponding authors: e-mail: m.chan@cs.ucl.ac.uk; p.treleaveng@cs.ucl.ac.uk*

ABSTRACT

While the prediction accuracy of a large-scale recommender system can generally be improved by learning from more and more training data over time, it is unclear how well a fixed predictive model can handle the changing business dynamics in a real-world scenario. The adjustment of a predictive model is controlled by the hyperparameter settings of a selected algorithm. Although the problem of model selection has been studied for decades in various disciplines, the adaptiveness of the initially selected model is not as well understood. This chapter presents an approach to continuously reselect hyperparameter settings of the algorithm in a large-scale retail recommender system. In particular, an automatic model selection technique is applied on collaborative filtering (CF) algorithms in order to improve prediction accuracy. Experiments have been conducted on a large-scale real retail dataset to challenge traditional assumption that a one-off initial model selection is sufficient. The proposed approach has been compared with a baseline approach and a widely used approach with two scalable CF algorithms. The evaluations of our experiments are based on a 2-year real purchase transaction dataset of a large retail chain business, both its online e-commerce site and its offline retail stores. It is demonstrated that continuous model selection can effectively improve the prediction accuracy of a recommender system. This chapter presents a new direction in improving the prediction performance of a large-scale recommender system.

Keywords: Hyperparameter optimization, Model selection, Collaborative filtering

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00005-8

1 INTRODUCTION

Large-scale recommender systems are increasingly important for retail businesses. Retailers provide personalized product recommendations to consumers through various mediums, for instance, web advertisement, email newsletters, and in-store discount vouchers. Recommender systems attempt to accurately predict consumers' preferences, or their behaviors, so that they can recommend products according to the best expected scenarios. Collaborative filtering (CF) is one of the most popular techniques to build recommender systems, as it relies on previous consumer behavioral data only. For instance, previous data of purchase transactions or website browsing history of consumers is sufficient for CF algorithms to predict consumers' future preferences. Consumer profiles, such as their age, gender, and other attributes, are not required. Product profiles are not required either. Commercial companies such as Amazon (Linden et al., 2003) and YouTube (Davidson et al., 2010) have successfully applied CF to build recommender systems for production use. The two main classes of algorithms under CF are neighborhood models (Sarwar et al., 2001) and latent factor models (Bell et al., 2007). Recently, latent factor models have gained popularity in both the research community and in the industry due to their superior accuracy and scalability. The discussion and our proposed approach in this chapter are therefore mainly based on latent factor models. The same approach, however, can be applied to neighborhood models directly.

Large retail businesses collect huge amounts of data every day. Scalable algorithms that can process data and construct predictive models using multiple machines in parallel are used in large-scale recommender systems. In this chapter, we focus on two scalable latent factor models, namely alternating least squares with weighted regularization (ALS-WR) (Zhou et al., 2008) and stochastic gradient descent (SGD) (Gemulla et al., 2011), that construct consumer and product feature matrices in Singular Value Decomposition (SVD) form. Both algorithms can be run on distributed Hadoop system (Shvachko et al., 2010). Like almost all other algorithms, these algorithms require the setting of one or more hyperparameters, such as regularization and learning rate, to build the latent factor models. These hyperparameters affect the prediction accuracy of the resulting models directly. Model selection problems and techniques have been studied in several disciplines such as Machine Learning and Statistics. Grid search and random search (Bergstra and Bengio, 2012) are two popular methods for global optimization. All these studies, however, take for granted that model selection (or setting) is a one-off initial process that needs to be done only once. No study, to the best of our knowledge, has been conducted on the effect of reselecting new hyperparameters for learning algorithms after new data has been collected over time.

Retail business environment changes rapidly. Product prices, for instance, may be changing frequently and new products appear regularly. Simply retraining the same predictive model with the addition of new data may not be

sufficient to accommodate these changes. Our hypothesis is that a recommender system model may render less accuracy, or even become invalid, as the business dynamic changes over time. We propose a continuous model selection approach, in which hyperparameters are re-evaluated, and are reselected if needed, before the model is retrained with new data. The prediction accuracy of this approach is compared to traditional noncontinuous approaches empirically based on a large and real retail datasets that contain both online e-commerce transactions and offline in-store transactions. Experiments show that the proposed approach outperforms a baseline approach and a state-of-the-art approach significantly in our retail scenarios.

The rest of the chapter is organized as follows. We review the related work in Section 2. We introduce the definitions of preference prediction and model selection in Section 3. We present and analyze the proposed approach in Section 4. We provide experimental evaluations in Section 5 and conclude in Section 6.

2 RELATED WORK

The related work is categorized into two parts: CF and automatic model selection.

Collaborative Filtering: The term CF was started by the first CF recommender system—Tapestry (Goldberg et al., 1992). Since CF approach requires no domain knowledge and it is generally more accurate due to the power of uncovering hidden patterns, it attracts a lot of attention from the research community (Militaru and Zaharia, 2010; Su and Khoshgoftaar, 2009) and is widely used in commercial systems (Linden et al., 2003). There are two primary techniques for CF to build predictive models using previous behavioral data, they are: the neighborhood approach and latent factor models. Neighborhood approach, such as item-based algorithm (Sarwar et al., 2001), estimates the relationships between product or consumer pairs, which are used for making a prediction. Latent factor models, such as SVD, estimate feature vectors that represent product and consumer profiles, which make them directly comparable. Latent factor models tend to be more accurate than neighborhood models (Koren, 2008). The two widely used scalable algorithms of latent factor models are SGD (Gemulla et al., 2011; Koren, 2008) and ALS-WR (Zhou et al., 2008), which are implemented in the experiments of this chapter.

Automatic Model Selection: Almost all CF algorithms come with one or more adjustable hyperparameters, such as regularization and learning rate. The setting of these values plays a key role on the generalizability and accuracy of the predictive model (Rasmussen et al., 2012). The problem of finding better hyperparameter values for an algorithm to improve prediction accuracy or to optimize certain goals is called hyperparameter optimization or model selection. This problem has been studied in multiple disciplines in the past,

for instance, response surface methodology (Weihs et al., 2006) is a popular method for tuning parameters in statistics while in Machine Learning, advanced methods to optimize hyperparameters have been proposed, such as the use of Gaussian Process (Snoek et al., 2012), Random Forests (Hutter et al., 2011), Tree-structured Parzen Estimator (Thornton et al., 2012), and Reinforcement learning (Nareyek, 2003). On the other hand, two simple techniques are widely used in practice, they are: grid search and random search (Bergstra and Bengio, 2012). Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. Random search, on the other hand, selects a value for each hyperparameter independently using a probability distribution. Existing literature in CF, or in recommender systems in general, handle hyperparameter settings by trial and error manually. While this approach may be acceptable if we need to conduct model selection only once at the initial stage, it is impractical when we want to conduct continuous model selection when new data comes it repeatedly. We, therefore, apply random search as a baseline model selection technique on CF algorithms in this chapter.

3 PREFERENCE PREDICTION

In this chapter, we narrowly define the objective of a recommender system is to suggest the top N products to each customer that he or she will most likely purchase. This kind of system, broadly speaking, involves a two-step process: preference prediction and ranking. During the prediction step, the system predicts a preference score on each available product for a targeted consumer. Higher score means that the consumer likes the product more and therefore has a higher chance to make a purchase transaction. Then, during the ranking step, the system ranks all available products for this consumer according to the predicted score. The top-ranked N products will be returned as recommendation results. Here are the formal definitions:

Definition 1 (Prediction). In our retail scenario, a prediction of consumer preference can be regarded as an estimation of the probability a consumer will purchase a product, i.e., $p(purchase = 1|u, i)$ where u is a targeted consumer and i is a targeted product. Preference prediction of all customers on all products can be represented by a $M \times N$ matrix $\boldsymbol{r}$ where M is the total number of consumers, N is the total number of products and $r_{mn} = p(purchase = 1|m, n)$.

Definition 2 (Model Selection). The term hyperparameter is used to distinguish it from regular parameters of a predictive model. Regular parameters characterize the model and they are estimated by model training using existing data. Hyperparameters, on the other hand, control how these regular parameters are estimated. Therefore, model selection is also called hyperparameter optimization. Like most machine learning algorithms, most CF algorithms come with some tunable hyperparameters such as learning rate and regularization. Selecting appropriate settings for these hyperparameters is crucial for the

accuracy of prediction. Hyperparameter optimization, or model selection, can be defined formally as an optimization problem itself, which can be written as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, C(\boldsymbol{p}_{\mathrm{v}}, \boldsymbol{q}_{\mathrm{v}}, \boldsymbol{r}_{\mathrm{v}}, M(\boldsymbol{p}_{\mathrm{tr}}, \boldsymbol{q}_{\mathrm{tr}}, \boldsymbol{r}_{\mathrm{tr}}, \theta))$$

and the performance of a set of hyperparameter settings can be evaluated by a cost function:

$$C(\boldsymbol{p}_{\mathrm{test}}, \boldsymbol{q}_{\mathrm{test}}, \boldsymbol{r}_{\mathrm{test}}, M(\boldsymbol{p}_{\mathrm{tr+v}}, \boldsymbol{q}_{\mathrm{tr+v}}, \boldsymbol{r}_{\mathrm{tr+v}}, \theta^*))$$

where C is the cost function, M is the modeling function, θ is a hyperparameter set, $\boldsymbol{p}_{\mathrm{tr}}$ and $\boldsymbol{q}_{\mathrm{tr}}$ are the consumer and product feature vectors of the training dataset, $\boldsymbol{p}_{\mathrm{v}}$ and $\boldsymbol{q}_{\mathrm{v}}$ are the consumer and product feature vectors of the validation dataset, $\boldsymbol{p}_{\mathrm{test}}$ and $\boldsymbol{q}_{\mathrm{test}}$ are the consumer and product feature vectors of the test dataset and $\boldsymbol{r}_{\mathrm{tr}}$, $\boldsymbol{r}_{\mathrm{v}}$, and $\boldsymbol{r}_{\mathrm{test}}$ are the vectors of preference scores of training, validation and test datasets, respectively.

4 PROPOSED CONTINUOUS MODELING

We propose a novel approach to search for better hyperparameters for a targeted algorithm before each iteration of model retraining with the addition of new data. Algorithm 1 is an overview of the process. Root mean squared error (RMSE) and mean average precision (MAP@k) are used to evaluate the prediction results. Major components of the proposed approach is described as follows:

Algorithm 1 Continuous model selection.

Data: D (dataset of month 1-24), A (algorithm)
Result: RMSE and MAP@k on each month of 13-24
trainD = D[month 1-12];
for $m \in \{13..24\}$ **do**
    θ^* = select hyperparameters of A based on *trainD*;
    P = model of A trained by *trainD* and θ^*;
    evaluate RMSE of P on $D[m]$;
    evaluate MAP@k of P on $D[m]$;
    trainD = *trainD* + $D[m]$;
end

4.1 Collaborative Filtering

Retail businesses, especially those operate as a chain, collect huge amount of data every day. A parallel distributed CF algorithm is necessary to handle the data in a scalable way. In our experiments, two popular distributed CF algorithms, SGD and ALS-WR, are implemented. We also apply a technique

to handle data of implicit consumer feedback as consumers do not express their preferences explicitly, such as rating products, in many retail scenarios.

4.1.1 Stochastic Gradient Descent

SGD initializes feature vectors that represent the profiles of consumers and products with random values. It then computes the gradient of the cost function and updates the values with steps in the direction of the gradient based on training data (Koren et al., 2009; Takács et al., 2009). This chapter follows the SGD implementation of Koren (2008), namely Bias SGD, which is a slightly improved version of pure SGD as it computes a bias for each consumer and product. The prediction accuracy is improved at the cost of extra computation. The tuning of three hyperparameters, which are listed on Table 1, is needed in order to apply a SGD algorithm to construct a predictive model.

4.1.2 Alternating Least Squares with Weighted Regularization

ALS-WR also initializes feature vectors that represent the profiles of consumers and products with random values. It then updates the feature vectors by applying a least squares solution to solve a quadratic cost function (Zhou et al., 2008). Although the computational cost of a least squares is more expensive than the one of SGD, usually fewer ALS iterations are required to obtain similar prediction accuracy as SGD. Also, the distributed implementation of ALS is more efficient than the one of SGD. Furthermore, ALS requires less hyperparameter settings. The only hyperparameter required by ALS-WR is shown in Table 2.

TABLE 1 Hyperparameters of Bias SGD Algorithm

Parameter Name	Description
λ	Regularization to prevent overfitting
lrate	Learning rate
decay	Learning rate decay

TABLE 2 Hyperparameters of ALS Algorithm

Parameter Name	Description
λ	Regularization to prevent overfitting

4.1.3 Implicit Feedback

In real world retail environment, consumer behavioral data is usually collected implicitly. Consumers are not required to express their preferences on products explicitly as it may intrude their shopping experiences. Therefore, the most common form of data being collected is nonnegative feedback, such as purchase transactions and website browsing history. Explicit feedback such as rating is rarely collected. Under this situation, many CF algorithms cannot be applied directly as we lack feedback on which products consumers dislike. Some techniques can be applied to solve this problem. For ALS-WR, consumers' purchase history is transformed into confidence for preference. The cost function takes not only consumer and product pairs that have interactions in the past into consideration, but all other unobserved consumer and product pairs as well. The details of this technique has been discussed in Hu et al. (2008). For SGD, we follow the technique described in Pan et al. (2008), which is a straightforward procedure to add negative examples randomly for unobserved consumer and product pairs.

4.2 Update with New Data

Existing literature assumes that model selection for CF algorithms needs to be done once only at the initialization stage. Under this assumption, recommender systems retrain models with the same hyperparameter settings when new data comes in. This chapter challenges this status quo. Our hypothesis is that the dynamics of real world environments, especially in retail businesses, change rapidly, a model that is appropriate in the past may become less effective as time goes by. Prediction accuracy of a recommender system should be improved if the model is updated, not only retrained, continuously. To update the model continuously, the system should be able to automatically select new hyperparameter settings for the underlying algorithm when more data is collected. Tuning these hyperparameters manually is not practical as it would require human involvement every time. Automatic model selection technique addresses this issue and is discussed next.

4.3 Automatic Model Selection

Automatic hyperparameter tuning has been studied in various disciplines, such as Machine Learning and Statistics. While some techniques are algorithm-specific such as Dongyuan and Xiaoyun (2010) and Acevedo et al. (2007), some global optimization approaches can be applied to any algorithms, including CF algorithms. Two widely used global optimization approaches are *grid search* and *random search* (Bergstra and Bengio, 2012). Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. Random search, on the other

hand, selects a value for each hyperparameter independently using a probability distribution. Both approaches evaluate the cost function based on the generated hyperparameter sets. Despite its simplicity, random search has proven to be more efficient than grid search in some cases empirically (Bergstra and Bengio, 2012). Since we mainly concern about the comparison between the traditional one-off modeling approach and our proposed continuous modeling approach in this chapter, the detailed study of the efficiency of model selection techniques is not within the scope. It is therefore reasonable to choose either one of these techniques to perform automatic model selection. The random search technique is chosen due to its simplicity.

5 EXPERIMENTAL EVALUATIONS

5.1 Data Set

Our dataset, which is provided by a large UK retail chain business, is a 2-year anonymized product purchase records of loyalty card holders on the retailer's e-commerce site and in all physical stores of the retailer in the UK. It contains complete transaction records of 10,217,972 unique loyalty card holders and 2939 unique products under 10 selected brands. There are 21,668,137 in-store purchase transaction records and 2,583,531 online purchase transaction records. All data is collected in a real non-experimental setting.

We take all records of purchase transactions and give each a preference score of 1. For instance, consumer u purchases a product i at time t is represented by a comma-separated line: $u, i, t, \mathbf{1}$. The resulting dataset contains only consumer–product pairs with preference score of 1 only. This transaction dataset is further divided into two: One contains transaction records of the online e-commerce site and the other one contains transaction records of offline retail stores.

5.2 Evaluation Metrics

One way to evaluate the prediction accuracy of a recommender system is to measure its prediction error of preference scores directly. Recommender systems predict consumer preferences based on predictive models that are built for predicting preference scores of unseen consumer–product pairs. A preference score ranges between 0 and 1, which can also be seen as an estimated probability of a consumer purchasing a product, i.e., $p(purchase = 1|u, i)$ where u is the targeted consumer and i is the targeted product. A higher score means that the consumer is more likely to purchase the product. The predicted scores will be evaluated against test sets that contain the actual purchase transactions, i.e., consumer–product pairs with score of 1. RMSE can be used as the prediction error metric: $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}$, where n is the total number of pairs

in a test set, r_i is the true score which is always 1 in this case, and $\hat{r}_i$ is the predicted score. The lower the RMSE score, the more accurate the predictive model is.

Normally, the end result of a recommender system is to return a top-K recommendation list. Another way to evaluate the prediction accuracy of a recommender system is, therefore, to treat the prediction as a ranking problem. Specifically, we want to evaluate the average precision of the predicted recommendation list for each consumer (Herlocker et al., 2004). Suppose a consumer has purchased n products in the test dataset and the system can recommend up to K products to this consumer. The average precision score at K, i.e., ap@K, is: ap@K= $\sum_{k=1}^{K} P(k)/\min(n, K)$ where $P(k) = 0$ if the consumer has not purchased the k-th product of the recommended list in the test dataset and $P(k) = k$ if otherwise. The mean average precision for M consumers at K, i.e., MAP@K, is the average of the average precision of each consumer, which is defined as: MAP@K= $\sum_{m=1}^{M} ap@K/M$. The higher the MAP@K score, the better the recommender system performs. We are going to evaluate our proposed continuous model selection approach with both RMSE and MAP@K metrics.

5.3 Experimental Setup

In this experiment, we evaluate the effect of continuous model selection under two separate environments, namely online e-commerce and offline stores. Under each environment, the prediction accuracy over time of three approaches is compared. In particular, this experiment evaluates and compares the consumer preference prediction accuracy of predictive models built, and updated, by three approaches on each month of the second year of our dataset, as illustrated in Fig. 1. The detailed procedure is described as follows.

We first evaluate the online e-commerce environment along with ALS-WR algorithm. We use $D1$ to represent all online e-commerce transaction data of the first year and $D2_m$ to represent all online e-commerce transaction data of the m month of the second year. For instance, $D2_1$ means all transaction data of the first month of the second year.

In this experiment, we define the width of the feature vectors for consumers and products as 20 and the maximum number of iterations as 30. By using random search technique, a pool of 1000 available hyperparameter sets, known

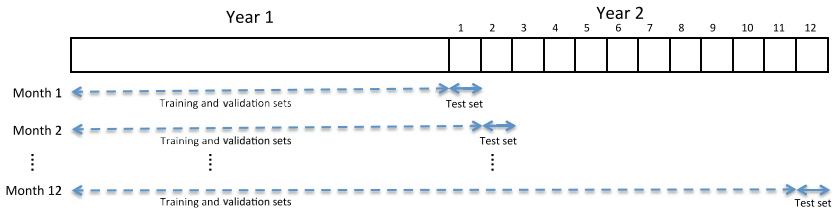


FIGURE 1 Training, validation, and test sets split for 12 months.

as $\boldsymbol{\theta}$, is generated for the only hyperparameter of ALS-WR: the search range of λ is $[0.01, 0.1]$. The RMSE and MAP@20 of the following three approaches on each $D2_m$ are evaluated:

Fixed: The first approach starts by selecting one set of hyperparameters from $\boldsymbol{\theta}$ that performs the best based on the first year data. This set of hyperparameters is written as $S_{\boldsymbol{\theta}}(D1)$. A predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built using $D1$ and $S_{\boldsymbol{\theta}}(D1)$. We evaluate the RMSE and MAP@20 of this model on each month of the second year, i.e., $D2_m$. This model is not updated with any new data. It serves as the baseline to compare with the other two approaches.

NewDataOnly: The second approach also starts by selecting the same set of hyperparameters, i.e., $S_{\boldsymbol{\theta}}(D1)$, from $\boldsymbol{\theta}$. A predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built as well. We evaluate the RMSE and MAP@20 of this model on the *first month* of the second year, i.e., $D2_1$, only. Then, we retrain the predictive model using $S_{\boldsymbol{\theta}}(D1)$ and the first year data *plus* the data of the first month of the second year, which becomes $M(D1 + D2_1, S_{\boldsymbol{\theta}}(D1))$. We evaluate the RMSE and MAP@20 of this updated model on $D2_2$. Similarly, we build $M(D1 + D2_1 + D2_2, S_{\boldsymbol{\theta}}(D1))$ and evaluate its RMSE and MAP@20 on $D2_3$ and so on, until we have evaluated all twelves $D2_m$. The hyperparameter set remains unchanged throughout the whole process.

NewDataNewParams: The third approach, as outlined in Algorithm 1, also starts by selecting the same set of hyperparameters, i.e., $S_{\boldsymbol{\theta}}(D1)$, from $\boldsymbol{\theta}$. Again, a predictive model, $M(D1, S_{\boldsymbol{\theta}}(D1))$, is built. Similar to the second approach, we evaluate the RMSE and MAP@20 of this model on the *first month* of the second year, i.e., $D2_1$, only. Uniquely for this proposed approach, we do the model selection again at this stage. We try to find another set of hyperparameters from $\boldsymbol{\theta}$ that performs the best based on the first year data *plus* the data of the first month of the second year. This new hyperparameter set, $S_{\boldsymbol{\theta}}(D1 + D2_1)$, may be the same as $S_{\boldsymbol{\theta}}(D1)$ if it is still the best performing set. Then, we retrain the predictive model using $S_{\boldsymbol{\theta}}(D1 + D2)$ and the first year data *plus* the data of the first month of the second year, which becomes $M(D1 + D2_1, S_{\boldsymbol{\theta}}(D1 + D2_1))$. We evaluate the RMSE and MAP@20 of this updated model on $D2_2$. Similarly, we reselect the best hyperparameter set from $\boldsymbol{\theta}$ and evaluate the RMSE and MAP@20 of $M(D1 + D2_1 + D2_2, S_{\boldsymbol{\theta}}(D1 + D2_1 + D2_2))$ on $D2_3$ and so on, until we have evaluated all twelves $D2_m$.

For all these approaches, the selection of the best set of hyperparameters is done by evaluating RMSE or MAP@20, according on the final evaluation metrics, with k-fold cross-validation where $k = 10$. Also, techniques to handle implicit feedback data for ALS-WR and SGD discussed previously have been applied.

We want to observe two issues in particular:

- whether a hyperparameter set that doesn't perform the best at earlier month would become the best choice in later months
- compare the prediction accuracy of **Fixed**, **NewDataOnly**, and **NewDataNewParams**

The results of the experiment are shown in Fig. 2a and c. This experiment is repeated with another CF algorithm—SGD. For SGD, we define the width of the feature vectors for consumers and products as 20 and the maximum number of iterations as 30. By using random search technique, a pool of 1000 available hyperparameter sets is also generated for the three hyperparameters of SGD: the search range of λ is [0.001, 0.01], the range of *lrate* is [0.001, 0.01] and the range of *decay* is [0.1, 1]. The results are shown in Fig. 2b and d.

At the end, the whole experimental setup is rerun again for the purchase transaction dataset of offline stores. The results are shown in Fig. 3a–d.

5.4 Effectiveness on the Online Site

Figure 2a shows the RMSE evaluation of the consumer preference prediction using ALS-WR for the second year of the online e-commerce dataset. The *NewDataOnly* approach clearly outperforms the *Fixed* baseline approach in every month. It means that retraining the predictive model with new data every month helps to improve preference score prediction in this case. For the proposed *NewDataNewParams* approach, new hyperparameter sets are selected at month 5 and month 10, as shown in Table 3. The predictive model is therefore retrained based on a new λ value from month 5 to month 9 and another new value from month 10 to month 12. During the period between month 5 to month 12, inclusively, *NewDataNewParams* outperforms *NewDataOnly* in every month except month 11. Figure 2c shows the MAP@20 evaluation of Top-20 recommendation, i.e., MAP@20, for the same ALS-WR. As shown, the curve representing the Fixed approach fluctuates in a relatively large range while the curves of the other two approaches both appear to be relatively smoother. For the Fixed approach, the results of the top 20 recommendation perform badly on month 3, 8, and 12. The *NewDataOnly* approach improves the recommendation results in these and some other months significantly. It outperforms the *Fixed* baseline approach in 8 out of the 11 months after an exactly same starting in the first month. For the *NewDataNewParams* approach, it outperforms *NewDataOnly* every month from month 5 to month 9—the period during which the predictive model is retrained by the first new hyperparameter set. From month 10 to 12, when another new hyperparameter set is used, *NewDataNewParams* outperforms *NewDataOnly* in 2 out of 3 months.

Figure 2b and d show the RMSE and MAP@20 evaluation of prediction using SGD, respectively. For the proposed *NewDataNewParams* approach, a new hyperparameter set is selected at month 8, as shown in Table 4. Similar to the results for ALS-WR, the *NewDataOnly* approach outperforms the *Fixed* baseline approach in most months. Unlike ALS-WR, a new hyperparameter set is selected once only for the *NewDataNewParams* at month 8. Since then, the *NewDataNewParams* approach slightly outperforms the *NewDataOnly* approach in every month except at month 9 based on RMSE evaluation. In addition, the *NewDataNewParams* approach slightly outperforms the *NewDataOnly*

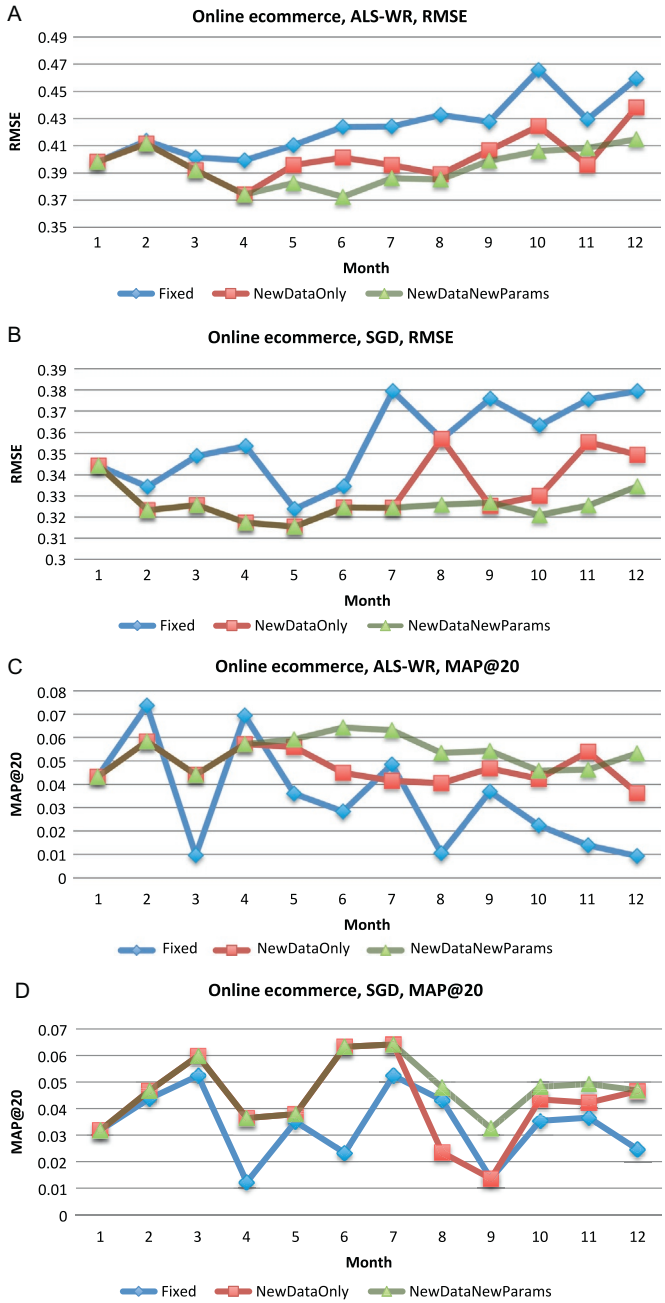


FIGURE 2 Results on online e-commerce dataset. (a) RMSE on ALS-WR—second year e-commerce; (b) RMSE on SGD—second year e-commerce; (c) MAP@20 on ALS-WR—second year e-commerce; and (d) MAP@20 on SGD—second year e-commerce.

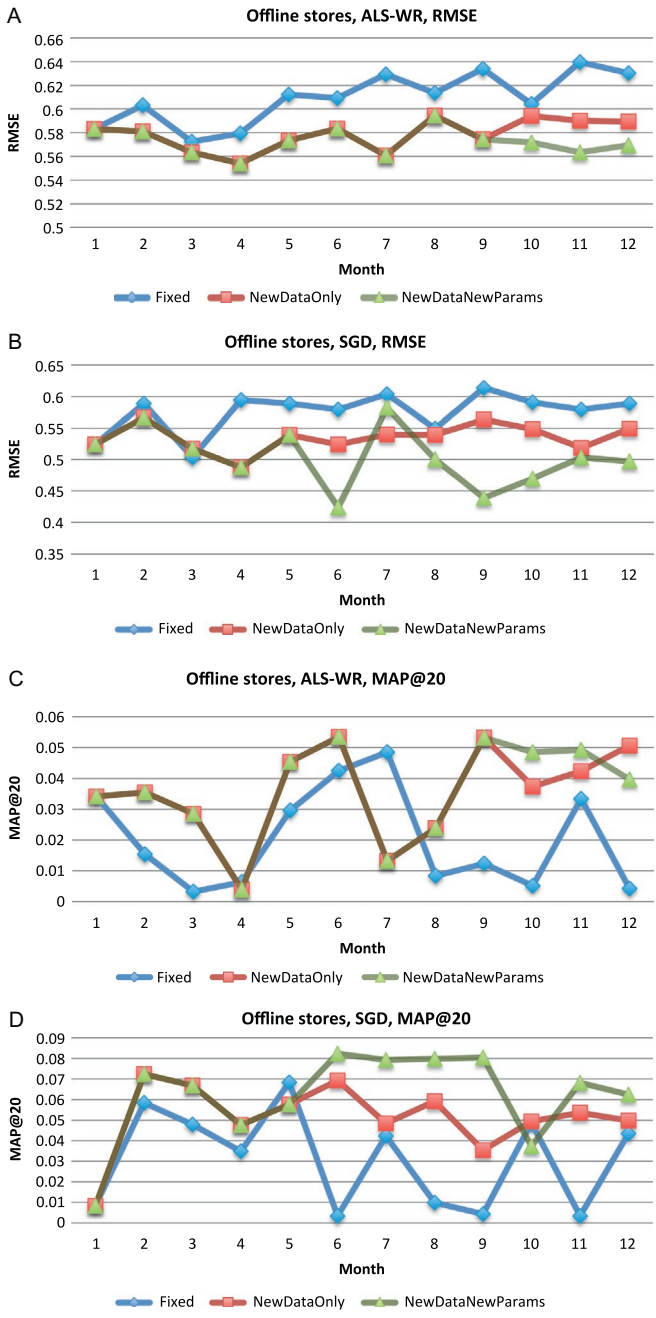


FIGURE 3 Results on offline stores dataset. (a) RMSE on ALS-WR—second year offline stores; (b) RMSE on SGD—second year offline stores; (c) MAP@20 on ALS-WR—second year offline stores; (d) MAP@20 on SGD—second year offline stores.

TABLE 3 Selected ALS-WR Hyperparameters

	Month	λ
Online	1–4	0.054
	5–9	0.038
	10–12	0.033
Offline	1–9	0.062
	10–12	0.048

TABLE 4 Selected SGD Hyperparameters

	Month	λ	*lrate*	*decay*
Online	1–7	0.009	0.004	0.98
	8–12	0.006	0.004	0.95
Offline	1–5	0.006	0.005	0.91
	6–12	0.005	0.001	0.78

approach in every month based on MAP@20 evaluation. The observations are similar to those under ALS-WR.

In this experiment for the online e-commerce site, there are two main observations: Firstly, a hyperparameter set that is not the best choice at the beginning may become the best choice as more data is being collected over time; Secondly, conducting model selection continuously as more data is being collected improves prediction accuracy in general.

5.5 Effectiveness on Offline Stores

Figure 3a shows the RMSE evaluation of the consumer preference prediction using ALS-WR for the second year of the offline stores dataset. Same as the result for the online e-commerce dataset, the *NewDataOnly* approach clearly outperforms the *Fixed* baseline approach in every month. For the proposed *NewDataNewParams* approach, a new hyperparameter set is not selected until month 10 and it is the only time a new hyperparameter set is selected, as shown in Table 3. The predictive model is therefore retrained based on a new λ value from month 10 to month 12. In these 3 months, *NewDataNewParams* always outperforms *NewDataOnly* . Figure 3c shows the MAP@20 evaluation of Top-20 recommendation for ALS-WR. As shown, the curve representing the Fixed approach fluctuates in a relatively large range while the curves of the other two approaches appear to be relatively smoother. Unlike the online

e-commerce environment, the *NewDataOnly* approach and the *NewDataNewParams* approach do not seem to have smoothen the curve when comparing to the *Fixed* approach under the offline stores environment. Both approaches outperform the *Fixed* approach in all months except month 4 and month 7. For the *NewDataNewParams* approach, a new hyperparameter set is selected at month 10. It outperforms *NewDataOnly* on month 10 and 12 but underperforms on month 11. The advantage of *NewDataNewParams* over *NewDataOnly* is relatively uncertain in this case.

Figure 3b and d show the RMSE and MAP@20 evaluation of prediction using SGD, respectively. For RMSE evaluation, the *NewDataOnly* approach outperforms the *Fixed* baseline approach in all months except month 3. A new hyperparameter set is selected once only for the *NewDataNewParams* at month 6, as shown in Table 4, which is much earlier than the case of ALS-WR. Since then, the *NewDataNewParams* approach outperforms the *NewDataOnly* approach in every month except at month 7 based on RMSE. For MAP@20 evaluation, the *NewDataOnly* approach outperforms the *Fixed* baseline approach significantly in all months except month 5. After the predictive model is retrained with the newly selected hyperparameter set at month 6, the *NewDataNewParams* approach outperforms the *NewDataOnly* approach significantly, except in month 10 where *NewDataNewParams* performs even worse than the baseline. In this experiment for the offline stores of the retail business, the general outcomes are similar to those of the online e-commerce site. The major difference is that a new hyperparameter set for ALS-WR is not selected until a much later stage in the offline stores environment. Because of this, the advantage of the *NewDataNewParams* approach over the *NewDataNewParams* approach is not conclusive in this single scenario. One possible reason is that the business dynamics or consumer preference does not change much in a year in the offline stores environment, therefore a new model selection is not necessary until a later time. Another possible reason, however, is that the model selection technique we apply is not efficient enough to find an optimal hyperparameter set that can improve the performance earlier. Identifying the real reason may become part of our future work.

6 CONCLUSION AND FUTURE WORK

In this chapter, we investigate whether reselecting hyperparameters for model selection repeatedly after the introduction of new data would have any impact on the prediction accuracy of recommender systems. Previous literatures assume that the process of model selection, or hyperparameter optimization, is needed only once at the initial stage. Therefore, the widely accepted approach nowadays is to retrain the predictive model periodically using the *same* hyperparameter settings after new data is collected.

The main contribution of the chapter is to present a novel study on the effect of optimizing hyperparameters continuously. Three approaches have been evaluated: Firstly, it is *Fixed*, which is a baseline approach that does not retrain the predictive model at all. Secondly, it is *NewDataOnly*, which is the existing state-of-the-art approach used by researchers and practitioners in the industry. For this approach, the predictive model is retrained periodically with new data. However, the hyperparameter settings are fixed once they are defined at the initial stage. Thirdly, it is our proposed *NewDataNewParams*, which tries to optimize the hyperparameter settings every time before the predictive model is retrained with new data. We have shown that the proposed continuous model selection approach, i.e., *NewDataNewParams*, improves prediction accuracy of a recommendation system most of the time empirically. In particular, experiments show that the improvement has been achieved in both scalable collaborative algorithms (ALS-WR and SGD) that we have implemented. Experiments also show that the proposed approach improves prediction accuracy of recommender systems for both online e-commerce site and offline retail stores of the retail chain business that we have investigated.

We conclude that, in our retail scenarios, not only does our proposed approach outperforms the baseline approach *Fixed* significantly, it also outperforms the existing state-of-the-art approach *NewDataOnly*: For experiments on the online e-commerce dataset, *NewDataNewParams* has an average of 15.8% and 22.6% improvement over *NewDataOnly* in terms of MAP@20 for ALS-WR and SGD, respectively, with a maximum improvement of 51.9% and 139.5%, respectively, on the best months. *NewDataNewParams* also has an average of 1.9% and 2% improvement over *NewDataOnly* in terms of RMSE for ALS-WR and SGD, respectively, with a maximum improvement of 7.2% and 8.7%, respectively, on the best months. For experiments on the offline stores dataset, *NewDataNewParams* has an average of 2.1% and 22.65% improvement over *NewDataOnly* in terms of MAP@20 for ALS-WR and SGD, respectively, with a maximum improvement of 30% and 126.7%, respectively, on the best months. *NewDataNewParams* also has an average of 1% and 5.6% improvement over *NewDataOnly* in terms of RMSE for ALS-WR and SGD, respectively, with a maximum improvement of 4.5% and 22.1%, respectively, on the best months. We have also discovered that a hyperparameter set that is not selected at the initial stage may become the best choice at a later time.

This chapter presents a new direction to improve the prediction performance of large-scale recommender systems in real-world retail scenarios. It is worth mentioning that a new hyperparameter set can only be selected in much later iteration in some situations in our experiments. Further work needs to be done to determine whether it is caused by the inefficiency of the selected automatic hyperparameter tuning technique, or that it is due to some limitations of the continuous model selection approach in certain cases. Future work should also consider the use of other automatic model selection techniques that are more efficient and less expensive computationally.

REFERENCES

Acevedo, J., Maldonado, S., Siegmann, P., Lafuente, S., Gil, P., 2007. Tuning L1-SVM hyperparameters with modified radius margin bounds and simulated annealing. In: Proceedings of the 9th International Work Conference on Artificial Neural Networks, IWANN'07. Springer-Verlag, Heidelberg, Germany, pp. 284–291. URL http://dl.acm.org/citation.cfm?id=1768409.1768448.

Bell, R., Koren, Y., Volinsky, C., 2007. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '07. ACM. San Jose, CA, USA, pp. 95–104. URL http://doi.acm.org/10.1145/1281192.1281206.

Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, 281–305. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2188385.2188395.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., et al., 2010. The YouTube video recommendation system. In: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10. ACM. Barcelona, Spain, pp. 293–296. URL http://doi.acm.org/10.1145/1864708.1864770.

Dongyuan, H., Xiaoyun, C., 2010. Tuning SVM hyperparameters in the primal. In: Second International Conference on Computational Intelligence and Natural Computing Proceedings (CINC 2010), vol. 1, pp. 201–204.

Gemulla, R., Nijkamp, E., Haas, P.J., Sismanis, Y., 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11. ACM. San Diego, CA, USA, pp. 69–77. URL http://doi.acm.org/10.1145/2020408.2020426.

Goldberg, D., Nichols, D., Oki, B.M., Terry, D., 1992. Using collaborative filtering to weave an information tapestry. Commun. ACM 35 (12), 61–70. ISSN 0001-0782. http://doi.acm.org/10.1145/138859.138867.

Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T., 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. 22, 5–53. ISSN 1046-8188. http://doi.acm.org/10.1145/963770.963772. URL http://doi.acm.org/10.1145/963770.963772.

Hu, Y., Koren, Y., Volinsky, C., 2008. Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining (ICDM '08), pp. 263–272.

Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration. In: Learning and Intelligent Optimization. Springer, Heidelberg, Germany, pp. 507–523.

Koren, Y., 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434.

Koren, Y., Bell, R., Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer 42 (8), 30–37. ISSN 0018-9162. http://doi.ieeecomputersociety.org/10.1109/MC.2009.263.

Linden, G., Smith, B., York, J., 2003. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. 7 (1), 76–80. ISSN 1089-7801. doi:10.1109/MIC.2003.1167344.

Militaru, D., Zaharia, C., 2010. A survey of collaborative filtering-based systems for online recommendation. In: Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business, ICEC '10. ACM, Seoul, Korea, pp. 43–47. URL http://doi.acm.org/10.1145/2389376.2389383.

Nareyek, A., 2003. Choosing search heuristics by non-stationary reinforcement learning. Appl. Optim. 86, 523–544.

Pan, R., Zhou, Y., Cao, B., Liu, N., Lukose, R., Scholz, M., et al., 2008. One-class collaborative filtering. In: Eighth IEEE International Conference on Data Mining (ICDM '08), pp. 502–511.

Rasmussen, P.M., Hansen, L.K., Madsen, K.H., Churchill, N.W., Strother, S.C., 2012. Model sparsity and brain pattern interpretation of classification models in neuroimaging. Pattern Recogn. 45 (6), 2085–2100. ISSN 0031-3203. doi:10.1016/j.patcog.2011.09.011. Brain Decoding. URL http://www.sciencedirect.com/science/article/pii/S0031320311003906.

Sarwar, B., Karypis, G., Konstan, J., Reidl, J., 2001. Item-based collaborative filtering recommendation algorithms. In: WWW '01 Proceedings of the 10th International Conference on World Wide Web. ACM, New York, NY, USA, pp. 285–295.

Shvachko, K., Kuang, H., Radia, S., Chansler, R., 2010. The Hadoop distributed file system. In: IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST, 2010). IEEE, Incline Village, Nevada, USA, pp. 1–10.

Snoek, J., Larochelle, H., Adams, R.P., 2012. Practical Bayesian optimization of machine learning algorithms. ArXiv preprint arXiv:1206.2944.

Su, X., Khoshgoftaar, T.M., 2009. A survey of collaborative filtering techniques. Adv. Artif. Intell. 2009, 4:2–4:2. ISSN 1687-7470. doi:10.1155/2009/421425. URL http://dx.doi.org/10.1155/2009/421425.

Takács, G., Pilászy, I., Németh, B., Tikk, D., 2009. Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res. 10, 623–656. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1577069.1577091.

Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2012. Auto-weka: automated selection and hyper-parameter optimization of classification algorithms. CoRR abs/1208.3719.

Weihs, C., Luebke, K., Czogiel, I., Technical report/Universität Dortmund, SFB 475 Komplexitätsreduktion in Multivariaten Datenstrukturen, 2006. Response surface methodology for optimizing hyper parameters.

Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R., 2008. Large-scale parallel collaborative filtering for the netflix prize. In: Proc. 4th Intíl Conf. Algorithmic Aspects in Information and Management, LNCS 5034. Springer, Shanghai, China, pp. 337–348.

Chapter 6

Zero-Knowledge Mechanisms for Private Release of Social Graph Summarization

Maryam Shoaran*,[1], Alex Thomo†, Jens H. Weber†

*Department of Mechatronics, School of Engineering Emerging Technologies, University of Tabriz, Tabriz, Iran

†Department of Computer Science, University of Victoria, Victoria, Canada

[1]Corresponding author: e-mail: mshoaran@tabrizu.ac.ir

ABSTRACT

Graphs have become increasingly popular for modeling data in a wide variety of applications, and graph summarization is a useful technique to analyze information from large graphs. Privacy-preserving mechanisms are vital to protect the privacy of individuals or institutions when releasing aggregate numbers, such as those in graph summarization. We propose privacy-aware release of graph summarization using zero-knowledge privacy (ZKP), a recently proposed privacy framework that is more effective than differential privacy for graph and social network databases. We first define group-based graph summaries. Next, we present techniques to compute the parameters required to design ZKP methods for each type of aggregate data. Then, we present an approach to achieve ZKP for probabilistic graphs.

Keywords: Zero-knowledge privacy, Differential Privacy, Social networks, Graph summarization, Randomization techniques, Sample complexity

1 INTRODUCTION

Nowadays, the graphs of many real world datasets are very large. For example, Facebook, the most well-known social network, contains data for over 900 million users and their relationships. Therefore, effective summarization methods need to be employed in order to make the analysis of such large graphs possible. We focus on graph summarization based on attribute groups. For instance, the nodes of a social graph can be grouped by attributes age and

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00006-X

profession, and statistics about the number of cross-group edges can be recorded. Statistics can reveal interesting facts about a graph. For instance, they could show surprising strong connections between groups of people in different age and profession groups. As such, group-based graph summarization (GGS) is a ubiquitous operation in virtually all the graph/social network software products (cf. Gep, 2009; Net, 2011; Nod, 2006; Paj, 1997; etc.). The result of GGS is a smaller summary graph, where each node summarizes a group of nodes in the original graph.

The problem is that, as with other aggregations, the summary graphs are often released to other parties for further research purposes, and this brings up the matter of privacy. Aggregate data included in a summary graph can reveal sensitive and private information about the nodes (participants) of the underlying graph (network).

Privacy-preserving data release has become one of the most important problems today. ϵ-Differential privacy (Dwork, 2008, 2010; Dwork et al., 2006) (DP for short) has been one of the leading privacy mechanisms in recent years. DP provides privacy for an individual of interest (IOI) by adding random noise to numerical outputs.

Some recent studies (cf. Gehrke et al., 2011; Kifer and Machanavajjhala, 2011), however, have highlighted situations in which DP might not provide sufficient privacy protection. This is especially pronounced in social networks where different types of auxiliary information, including the structure of network or the groups the individuals belong in, are often readily available to the public (Gehrke et al., 2011).

More specifically, whereas the goal of DP is to protect the participation of an individual (or relationship) in a dataset, in social networks we also need to protect the *evidence of participation* (Kifer and Machanavajjhala, 2011). To see this we present the following example. Suppose there are two groups g_1 and g_2 and we want to publish the number of edges between them. Bob, a member of g_1, has an edge to Alice, a member of g_2. As a consequence of this connection, some friends of Bob introduce edges to Alice. What we want to protect is Bob's edge to Alice. DP works in this case by ensuring that for any true answer, c or $c - 1$, the sanitized answer would be pretty much the same. However, this is not strong enough; the existence of Bob's edge changes the true answer not just by 1, but by a bigger number as it causes more edges to be created between the two groups.

Going beyond DP, Gehrke, Lui, and Pass proposed "zero-knowledge privacy" (ZKP) in Gehrke et al. (2011), which provides stronger privacy, especially for social graphs. The definition of ZKP is based on classes of aggregate functions. ZKP guarantees that an attacker cannot discover any personal information more than what can be inferred from some aggregate on a sample of a database with IOI removed. The sample complexity defines the level of privacy tolerance in ZKP. For instance, suppose in the Bob's example above the network size is 10,000 and the sample size is $\sqrt{10,000} = 100$. With such a sampling rate of

0.01, the evidence provided by say 10 more edges caused by Bob's edge will essentially be protected; with a high probability, none of these 10 edges will be in the sample.

In this chapter, we use ZKP to provide individual privacy in graph summarization. We address connection measures for groups in social graphs and present ZKP mechanisms for private release of such aggregate outputs. As ZKP inherently depends on the precise characterization of sample complexity, we propose methods to compute the sample complexity of our aggregate functions. In order to achieve this, we present techniques to express the aggregate functions as averages of specially designed, synthetic attributes on the nodes of the graph. Then we derive precise prescriptions on how to construct ZKP mechanisms for the aggregate functions we consider. Next, we present detailed examples and numeric evaluations of our ZKP mechanisms in terms of the parameters involved. These evaluations are valid for any case and illustrate the trade-offs involved when building ZKP mechanisms for graph summarization. Finally, we also present summarization measures for probabilistic graphs. This is especially important in social networks having edges of different influence captured by probabilities assigned on the edges.

2 RELATED WORK

Graph summarization is a ubiquitous method for analyzing large graphs. Virtually all the graph/social network products (cf. Gep, 2009; Net, 2011; Nod, 2006; Paj, 1997; etc.) create summaries in the form of smaller graphs by grouping the nodes based on attributes.

The common goal of privacy-preserving methods is to learn from data while protecting sensitive information of the individuals. k-Anonymity for social graphs (cf. Chester et al., 2011, 2013; Liu and Terzi, 2008) provides privacy by ensuring that combinations of identifying attributes appear at least k times in the dataset. The problem with k-anonymity and other related approaches, e.g., l-diversity (Machanavajjhala et al., 2007), is that they assume the adversary has limited auxiliary knowledge. Narayanan and Shmatikov (2009) present a de-anonymization algorithm and claim that k-anonymity can be defeated by their method using auxiliary data accessible by the adversary.

Among a multitude of different techniques, DP (Blum et al., 2005; Dwork, 2006, 2008; Dwork et al., 2006) has become one of the leading methods to provide individual privacy. Various differentially private algorithms have since been developed for different domains, including social networks (Hay et al., 2009; Rastogi et al., 2009). However as already shown, DP can suffer in social networks where specific auxiliary information, such as graph structures and friendship data, is easily available to the adversary. Important works showing the shortcomings of DP are Kifer and Machanavajjhala (2011, 2012).

Gehrke et al. (2011) present the notion of ZKP which is appealing for achieving privacy in social networks. ZKP guarantees that what can be learned

from a dataset including an individual is not more than what is learned from sampling-based aggregates computed on the dataset without that individual.

3 GRAPH SUMMARIZATION

We denote a graph as $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges connecting the nodes. We consider $\mathcal{S} \subset 2^V$ to be a set of disjoint node groups of size r or more that a social network wants to release statistics about.

Definition 1. The *$\mathcal{S}$-graph* of G is $\mathcal{G}_{G,\mathcal{S}} = (\mathcal{S}, \mathcal{E}_{\mathcal{S}})$, where

$$\mathcal{E}_{\mathcal{S}} = \{(g', g'') : g', g'' \in \mathcal{S} \text{ and } \exists v' \in g' \text{ and } \exists v'' \in g'' \text{ such that } (v', v'') \in E\}.$$

This definition says that two groups g' and g'' in $\mathcal{S}$ are connected through an edge in $\mathcal{G}_{G,\mathcal{S}}$ if there exists at least one edge in G that connects a node in g' to a node in g''.

Definition 2. The *$\mathcal{S}$-graph summarization ($\mathcal{S}$-GS)* is a function

$$\begin{aligned}
w_1 &: \mathcal{S} \longrightarrow [0,1] \\
w_2 &: \mathcal{E}_{\mathcal{S}} \longrightarrow [0,1] \times [0,1] \times [0,1] \\
w_1(g) &= \frac{|g|}{|V|} \\
w_2(g', g'') &= (x, y, z), \text{ where} \\
x &= \frac{|\{v' \in g' : \exists v'' \in g'', \text{ s.t. } (v', v'') \in E\}|}{|g'|} \\
z &= \frac{|\{v'' \in g'' : \exists v' \in g', \text{ s.t. } (v', v'') \in E\}|}{|g''|} \\
y &= \frac{|\{(v', v'') : v' \in g', v'' \in g'', (v', v'') \in E\}|}{|g'| \cdot |g''|}.
\end{aligned}$$

Throughout the chapter, we will refer to the elements of w_2 as $w_2(g', g'')[x]$, $w_2(g', g'')[y]$, and $w_2(g', g'')[z]$, or $w_2[x]$, $w_2[y]$, and $w_2[z]$, whenever g' and g'' are clear from the context. We will also use $w_2[.]$ to refer to any of three elements x, y, or z.

Example 1. Figure 1a shows a simple graph G, and $\mathcal{S}$ consisting of two groups g' and g''. Group g' has four nodes and group g'' has six nodes. There are several edges (eight of them) connecting members of g' to members of g''.

Figure 1b shows how g' and g'' are represented by a node each in $\mathcal{G}_{G,\mathcal{S}}$. The nodes and the edge connecting them in $\mathcal{G}_{G,\mathcal{S}}$ are labeled by w_1 and w_2 measures, respectively, as described above. Specifically, we have $w_1(g') = \frac{4}{4+6} = 0.4$ and $w_1(g'') = \frac{6}{4+6} = 0.6$. Since three of four nodes in g' and all the six nodes in g'' are connected with some nodes in the other group, we have $w_2(g', g'') = (\frac{3}{4}, \frac{8}{4\times 6}, \frac{6}{6}) = (0.75, 0.33, 1)$.

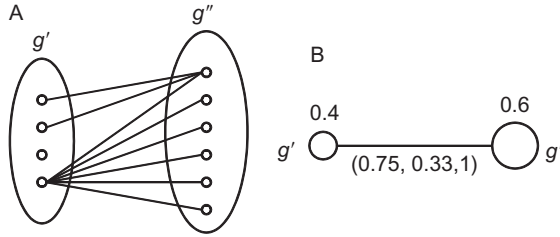


FIGURE 1 A graph and its summarization.

4 BACKGROUND ON ϵ-ZERO-KNOWLEDGE PRIVACY

ZKP introduced by Gehrke et al. (2011) is a privacy framework that is stronger than *DP*). ZKP is especially desirable in social networks where we need to protect not only the participation of a connection but also easy to find evidence of the participation, as, for example, the evidence given by other connections that were influenced by the connection.

ZKP is defined in relation with classes of sampling-based aggregate information. The class of sampling-based aggregation represents our tolerance for information release. For example, we can say that we are only comfortable to release the average age of a population computed on a $\sqrt{n}$ random sample. ZKP uses the notion of a simulator from zero knowledge and says that a simulator with the acceptable aggregate information can essentially compute whatever an adversary can compute by accessing the result of the mechanism (Gehrke et al., 2011). We describe ZKP in the following using a setting of graphs.

Let G be a graph. We denote by G_{-*} a graph obtained from G by removing a piece of information (for example an edge). G and G_{-*} are called *neighboring graphs*.

Let *San* be a mechanism that operates on a graph G (the complete database), and computes a *sanitized* answer to a query. The adversary's goal in a privacy scenario is to gain information about private matters of individuals (nodes) or connections (edges) in G using this released sanitized answer. Let $Adv(San(G), z)$ denote the output of the algorithm that an adversary employs to breach privacy. The adversary can interact with mechanism *San* and may have access to some auxiliary information z. The information in z is considered to be general, and easily accessible, e.g., information about the structure of the network (graph) or the groups that individuals belong in.

Let agg be a class of randomized algorithms that first select $k = k(n)$ random samples (nodes) without replacement from G_{-*} and then compute some aggregate information. Such algorithms output an approximate answer to the query.

Let *Sim*, "the simulator," be an algorithm. We denote by $Sim(T(G_{-*}), z)$ the information that the simulator can compute given the aggregate information computed by a $T \in agg_k$. In plain language, imagine *Sim* to be a person who

can be "extremely smart and capable (ESC)" and who has access to aggregates computed by the algorithms of class *agg* on the database where the sensitive information has been removed. Also, assume that the simulator also has access to background information z.

On the other hand, imagine the adversary to be a person who is also ESC, and has access to $San(G)$ as well as background information z. ZKP assures an individual that the participation in the network does not jeopardize her/his privacy. ZKP provides this guarantee by sanitizing the query answers such that the information that the adversary could extract from the output (sanitized answer) is computationally indistinguishable from the information that could be computed using sampling-based aggregates calculated on the network data that misses the IOI's sensitive information. That is, the adversary is not better off than some simulator even though he has access to the output of mechanism *San* computed on the whole database.

Definition 3. [Zero-knowledge privacy (Gehrke et al., 2011)] The mechanism *San* is *ϵ-zero-knowledge private with respect to agg* if there exists a $T \in agg$ such that for every adversary Adv, there exists a simulator *Sim* such that for every G, every $z \in \{0,1\}^*$, and every $W \subseteq \{0,1\}^*$, the following hold:

$$Pr[Adv(San(G), z) \in W] \leq e^{\epsilon} \cdot Pr[Sim(T(G_{-*}), z) \in W]$$
$$Pr[Sim(T(G_{-*}), z) \in W] \leq e^{\epsilon} \cdot Pr[Adv(San(G), z) \in W],$$

where probabilities are taken over the randomness of *San* and Adv, and T and *Sim*.

By this definition, ZKP guarantees that any additional information that an adversary can obtain about an individual by having access to the output of the mechanism is virtually not more than what can be computed by a simulator using some sampling-based (approximate) aggregates even without access to the mechanism and the sensitive data.

Note that the selection of k—the number of random samples—in *agg* algorithms is very important and it should be chosen so that with high probability very few of the nodes connected with the node whose information has to be private will be chosen. We will often index *agg* by k as agg_k to stress the importance of k. To satisfy the ZKP definition, a mechanism should use $k = o(n)$, say $k = \sqrt{n}$ or $k = \sqrt[3]{n^2}$, where n, the number of nodes in the database, is sufficiently large (see Gehrke et al., 2011). DP is a special case of ZKP where $k = n$.

As it will be illustrated in the upcoming sections, the specifications of algorithm $T \in agg$, e.g., sample size, are only used to compute the level of privacy needed in the ZKP mechanism. We stress that the ZKP mechanism is the only algorithm applied on the data. The simulator is only an abstract notion.

Achieving ZKP. Let $f : \mathbf{G} \to \mathbb{R}^m$ be a function that produces a vector of length m from a graph database. For example, given $\mathcal{G}_{G,\mathcal{S}}$, f produces the results of the $\mathcal{S}$-GS functions, i.e., w on edges.

We consider the L_1-Sensitivity to be defined as follows.

Definition 4 (*L_1-Sensitivity*). For $f : \mathbf{G} \to \mathbb{R}^m$, the L_1-sensitivity of f is

$$\Delta(f) = \max_{G',G''} ||f(G') - f(G'')||_1$$

for all neighboring graphs G' and G''.

Another essential definition is that of "sample complexity."

Definition 5 (*Sample Complexity (Gehrke et al., 2011)*). A function $f : Dom \to \mathbb{R}^m$ is said to have (δ, β)**-sample complexity** with respect to agg if there exists an algorithm $T \in agg$ such that for every $D \in Dom$ we have

$$Pr[||T(D) - f(D)||_1 \leq \delta] \geq 1 - \beta.$$

T is said to be a (δ, β)-sampler for f with respect to agg.

This definition bounds the probability of error between the randomized computation (approximation) of function f and the expected output of f. Basically, functions with low sample complexity (smaller δ and β) can be computed more accurately using random samples from the input data.

When the released information, as typical, is real numbers, the ZKP mechanism *San* achieves the privacy by adding noise to each of the numbers independently.

Let $Lap(\lambda)$ be the zero-mean Laplace distribution with scale λ, and variance $2\lambda^2$. The scale of Laplace noise in ZKP is properly calibrated to the sample complexity of the function that is to be privately computed. The following proposition expresses the relationship between the sample complexity of a function and the level of ZKP achieved by adding Laplace noise to the outputs of the function.

Proposition 1 (Gehrke et al., 2011). *Suppose $f : \boldsymbol{G} \to [a, b]^m$ has (δ, β)-sample complexity with respect to agg. Then, mechanism*

$$San(G) = f(G) + (X_1, \ldots, X_m),$$

where $G \in \boldsymbol{G}$, and $X_j \backsim Lap(\lambda)$ for $j = 1, \ldots, m$ independently, is

$$\ln\left((1-\beta)e^{\frac{\Delta(f)+\delta}{\lambda}} + \beta e^{\frac{(b-a)m}{\lambda}}\right)$$

–ZKP with respect to agg.

5 ZKP MECHANISM FOR GRAPH SUMMARIZATION

In this section, we design a ZKP mechanism to release a graph summarization. Let $\mathcal{G}_{G,\mathcal{S}} = (\mathcal{S}, \mathcal{E}_{\mathcal{S}})$ be the $\mathcal{S}$-graph for a graph G. Let f be a function that takes a graph $\mathcal{G}_{G,\mathcal{S}}$ as input and produces $c = |\mathcal{S}| + 3 \cdot |\mathcal{E}_{\mathcal{S}}|$ numbers, or differently said, a c-dimensional vector corresponding to w_1 and w_2 aggregates for the groups and the connecting edges.

Let $f = [f_1, \ldots, f_t]$ be the vector (or subvector) that is to be privately released. We apply a separate San_i (ZKP) mechanism, for $i \in [1, t]$, to each of the elements of f. Let us assume that each San_i provides ϵ_i-ZKP for f_i with respect to agg_{k_i}, where $k_i = k(n)/t$ and $n = |V|$. Then, based on the following proposition, f will be $(\sum_{i=1}^{t} \epsilon_i)$-ZKP with respect to $agg_{k(n)}$, where $k(n) = \sum_{i=1}^{t} k_i$.

Proposition 2 (Sequential Composition (Gehrke et al., 2011)). *Suppose San_i, for $i \in [1, n]$, is an ϵ_i-ZKP mechanism with respect to agg_{k_i}. Then, the mechanism resulting from composing*[1] *San_i's is $\left(\sum_{i=1}^{n} \epsilon_i\right)$-ZKP with respect to $agg_{(\sum k_i)}$.*

In this chapter, we consider edge (connection) privacy. We note that node privacy will not be considered here, since, as it is widely known (cf. Hay et al., 2009; Kifer and Machanavajjhala, 2011), it results in too noisy output with practically no utility.

5.1 Edge (Connection) Privacy

Consider $\mathcal{G}_{G,\mathcal{S}}$ and $\mathcal{G}_{G_{-e},\mathcal{S}}$, where $\mathcal{G}_{G_{-e},\mathcal{S}}$ is a neighboring graph of G obtained from G by removing edge e. In the edge privacy scenario, the total number of nodes (groups) and the size of each group are identical in $\mathcal{G}_{G,\mathcal{S}}$ and $\mathcal{G}_{G_{-e},\mathcal{S}}$. Therefore, the sensitivity of any w_1 function, including the ones in f, is zero, that is, $\Delta(w_1) = 0$.

On the other hand, removing an edge in G can change by at most 1 the numerator of each element x, y, and z in w_2 measures of $\mathcal{G}_{G_{-e},\mathcal{S}}$. Note that this change affects only one w_2 measure in the whole graph $\mathcal{G}_{G_{-e},\mathcal{S}}$. Therefore, the sensitivities of the elements of any w_2 function, including the ones in f, are

$$\Delta(w_2[x]) = \frac{1}{r} \quad \Delta(w_2[y]) = \frac{1}{r^2} \quad \Delta(w_2[z]) = \frac{1}{r},$$

where r is the minimum group size in $\mathcal{S}$.

In the following sections, the ZKP mechanisms are separately designed for w_1 and w_2 functions in f.

5.1.1 ZKP Mechanism for w_1

Suppose $w_1(g)$ is an element of f, where g is a group in $\mathcal{G}_{G,\mathcal{S}}$. Let $San = w_1(g) + Lap(\lambda)$ be a ZKP mechanism which adds random noise selected from $Lap(\lambda)$ distribution to the output of $w_1(g)$ in order to achieve ZKP. Our goal here is to come up with the right λ to achieve a predefined level of ZKP.

Based on the definition of ZKP, one should first know the sample complexity of the w_1 function. For this, without change in semantics, we will express w_1 so that it computes an average rather than a fraction of two counts. Then, using the

1. A set of computations that are separately applied on *one* database and each provides ZKP in isolation also provides ZKP for the set.

Hoeffding inequality (Mitzenmacher and Upfal, 2005), we compute the sample complexity of w_1.

Expressing w_1. We assume that in addition to the regular node attributes (if any), we have $|\mathcal{S}|$ new boolean attributes, one for each possible group. We denote these new attributes by uppercase B's indexed by the group id. A node v in graph G will have $B_g(v) = 1$ if v belongs to group g, and $B_g(v) = 0$, otherwise. We have that,

Proposition 3.

$$w_1(g) = \frac{\sum_{v \in V} B_g(v)}{|V|}.$$

Therefore, $w_1(g)$ can be viewed as the average value of attribute B_g over all nodes in G.

ZKP mechanism. Let $G = (V, E)$ be a graph enriched with boolean attributes as explained above. We would like to determine the value of $\lambda > 0$ for $Lap(\lambda)$ distribution which is to be used to add random noise to a $w_1(g)$ measure included in f. For this, first we compute the sample complexity of w_1 to be able to use Proposition 1 and establish an appropriate value for λ.

Let T be a randomized algorithm in agg_k, the class of randomized algorithms that operates on an input graph G. To randomly sample a graph G, algorithm T uniformly selects $k = k(n)/t$ random nodes from V, reads their attributes, and retrieves all the edges[2] incident to these k sample nodes.[3] From the sampled nodes and their incident edges with other sampled nodes, we consider $\mathcal{G}_{G',\mathcal{S}'} = (\mathcal{S}', \mathcal{E}'_{\mathcal{S}})$. Then, T approximates the value of $w_1(g)$ using $\mathcal{G}_{G',\mathcal{S}'}$. Since we have expressed $w_1(g)$ for a group g as an average, based on the Hoeffding inequality we have

$$Pr[|T(g) - w_1(g)| \leq \delta] \geq 1 - 2e^{-2k\delta^2}.$$

From this and Definition 5, we have that w_1 has $\left(\delta, 2e^{-2k\delta^2}\right)$-sample complexity with respect to agg_k.

Now we make the following substitutions in the formula of Proposition 1: $\beta = 2e^{-2k\delta^2}$, $\Delta(w_1(g)) = 0$, $b - a = 1$, and $m = 1$ and obtain that mechanism *San* is

$$\ln\left(e^{\delta/\lambda} + 2e^{1/\lambda - 2k\delta^2}\right) - \text{ZKP}$$

with respect to agg_k.

2. Clearly, only nondangling incident edges, whose both end nodes have been sampled, will be retrieved.
3. For other possible methods of graph sampling see, for example, Gehrke et al. (2011).

Similar to DP, one can set λ, the Laplace noise scale, to be proportional to "the error" as measured here by the sum of the sensitivity and sampling error, and inversely proportional to the ZKP privacy level

$$\lambda = \frac{\Delta(w_1) + \delta}{\epsilon} = \frac{1}{\epsilon} \cdot \frac{1}{\sqrt[3]{k}}.$$

Regarding δ, we can consider for instance a sample size $k = \sqrt[3]{n^2}$ and have $\delta = \frac{1}{\sqrt[3]{k}}$.

From all the above, the privacy level obtained will be

$$\begin{aligned} \ln\left(e^{\delta/\lambda} + 2e^{1/\lambda - 2k\delta^2}\right) &= \ln\left(e^{\epsilon} + 2e^{\epsilon\sqrt[3]{k} - 2\sqrt[3]{k}}\right) \\ &\leq \ln\left(e^{\epsilon} + 2e^{-\sqrt[3]{k}}\right) \\ &\leq \epsilon + 2e^{-\sqrt[3]{k}}. \end{aligned}$$

Thus, we have that by adding $Lap\left(\frac{1}{\epsilon \cdot \sqrt[3]{k}}\right)$ noise, mechanism *San* will be $\left(\epsilon + 2e^{-\sqrt[3]{k}}\right)$-ZKP with respect to agg_k. Of course, the privacy achieved is in fact better than this because of the above inequalities. We address finding of the exact λ given a ZKP privacy level and sample complexity in Section 7.

Example 2. Let graph G be a social graph with 100 million participants/nodes ($|V| = n = 100,000,000$), and g', g'' be two groups. Suppose the requested output vector is

$$f = \langle w_1(g'), w_2(g', g'')[x], w_2(g', g'')[y], w_2(g', g'')[z], w_1(g'') \rangle$$

and suppose that the minimum group size in $\mathcal{S}$ is $r = 5000$. Assume we would like to have for f a ZKP mechanism expressed with respect to an acceptable agg_k, where

$$k(n) = \sqrt[3]{100,000,000^2} = 215,443.$$

To privately release the first output in f, a randomized algorithm T can uniformly select

$$k_1 = 215,443/5 = 43,089$$

nodes and approximate the value of $w_1(g')$ using these samples. Let (δ_1, β_1) be the sample complexity of $w_1(g')$ where

$$\begin{aligned} \delta_1 &= \frac{1}{\sqrt[3]{k_1}} = \frac{1}{\sqrt[3]{43,089}} = 0.0285 \\ \beta_1 &= 2e^{-2k_1\delta_1^2} = 2e^{-2*43089*(0.0285)^2} = 7.97 * 10^{-31}. \end{aligned}$$

The sensitivity of f is

$$\Delta(f) = \frac{1}{r} + \frac{1}{r^2} + \frac{1}{r} = 0.0004.$$

Now, if we would like to use a mechanism which is 0.1 ZKP, we add random noise selected from a Laplace distribution with scale

$$\lambda_1 = \frac{\Delta(f) + \delta_1}{\epsilon} = \frac{0.0004 + 0.0285}{0.1} = 0.289$$

to the actual value of $w_1(g')$. With this noise scale, the ZKP privacy level of the mechanism is precisely

$$\epsilon_1 \leq \left(\epsilon + 2e^{-\sqrt[3]{k_1}}\right) = \left(0.1 + 2e^{-35.06}\right) \approx 0.1$$

with respect to agg_k.

5.1.2 ZKP Mechanism for w_2

Suppose the function $w_2(g, g')[.]$ is an element of f, where g and g' are groups in $\mathcal{G}_{G,\mathcal{S}}$. Let $San = w_2(g, g')[.] + Lap(\lambda)$ be a ZKP mechanism that adds random noise selected from $Lap(\lambda)$ distribution to $w_2(g, g')[.]$. To come up with the right λ first we compute the sample complexity of the w_2 function.

Expressing $w_2[x]$ and $w_2[z]$. To express the x or z elements of the w_2 function, we introduce $|\mathcal{S}|$ new boolean node attributes, each corresponding to a group. We denote these new attributes by B' indexed by the group id. A node v will have $B'_g(v) = 1$ if v has an edge with some node in group g, and $B'_g(v) = 0$, otherwise. Now for each pair of groups g and g', we can show the following proposition.

Proposition 4.

$$w_2(g, g')[x] = \frac{\sum_{v \in g} B'_{g'}(v)}{|g|}$$
$$w_2(g, g')[z] = \frac{\sum_{v \in g'} B'_g(v)}{|g'|}.$$

Hence, the x (or z) elements of $w_2(g, g')$ can be viewed as the average value of attribute $B'_{g'}$ (or B'_g) over the subset of nodes in G that are in g' (or g).

Expressing $w_2[y]$. To express y in w_2, we introduce $|\mathcal{S}|$ new node attributes, each corresponding to a group. We denote these new attributes by B'' indexed by the group id. Each attribute B''_g is a boolean vector of dimension $|g|$, where each dimension corresponds to a node in g. A node v will have $B''_g(v)[u] = 1$, where $u \in g$, if (v, u) is an edge in graph G, and $B''_g(v)[u] = 0$, otherwise. For each pair of groups g and g' we can show that

Proposition 5.

$$w_2(g, g')[y] = \frac{\sum_{v \in g, u \in g'} B''_{g'}(v)[u]}{|g| \cdot |g'|}$$
$$= \frac{\sum_{v \in g', u \in g} B''_g(v)[u]}{|g| \cdot |g'|}.$$

Therefore, the y measure in $w_2(g, g')$ can be viewed as the average of $B''_{g'}(v)[u]$'s or $B''_g(v)[u]$'s.

ZKP mechanism. Let $G = (V, E)$ be a graph enriched with boolean attributes as explained above. We would like to determine the value of $\lambda > 0$ for the $Lap(\lambda)$ distribution which will add random noise to $w_2(g, g')[.]$.

Let T be a randomized algorithm in agg_k. Algorithm T randomly samples graph G by uniformly selecting $k = k(n)/t$ random nodes from V and retrieving all the incident edges. With this sampling, the nodes in the groups of $\mathcal{G}_{G,\mathcal{S}}$ and the edges between them are randomly sampled as well. We call this sampled $\mathcal{S}$-graph $\mathcal{G}'_{G,\mathcal{S}} = (\mathcal{S}', \mathcal{E}'_{\mathcal{S}})$. Let us assume that we have a sample of each group and edges between groups and the size of a sample group g is k_g. Then, algorithm T approximates w_2 using the data from group samples. For the sample complexity of the elements of w_2, since we expressed them as averages, we can use the Hoeffding inequality as follows:

$$Pr[|T(g, g')[x] - w_2(g, g')[x]| \leq \delta] \geq 1 - 2e^{-2k_g\delta^2}$$
$$Pr[|T(g, g')[z] - w_2(g, g')[z]| \leq \delta] \geq 1 - 2e^{-2k_{g'}\delta^2}$$
$$Pr[|T(g, g')[y] - w_2(g, g')[y]| \leq \delta] \geq 1 - 2e^{-2(k_g \times k_{g'})\delta^2}.$$

Let us focus first on $w_2[x]$ ($w_2[z]$ is similar). Now we make the following substitutions in the formula of Proposition 1: $\beta = 2e^{-2k_g\delta^2}$, $\Delta(w_2(g, g')[x]) = 1/r$, $b - a = 1$, and $m = 1$. From this, we have that mechanism *San* is

$$\ln\left(e^{1/r+\delta/\lambda} + 2e^{1/\lambda - 2k_g\delta^2}\right) - \text{ZKP}$$

with respect to agg_k.

Again, one can set λ, the Laplace noise scale, to be proportional to "the error" as measured by the sum of the sensitivity and sampling error, and inversely proportional to ϵ

$$\lambda = \frac{\Delta(w_2)[x] + \delta}{\epsilon} = \frac{1}{\epsilon}\left(\frac{1}{r} + \frac{1}{\sqrt[3]{k_g}}\right).$$

Regarding δ, we can consider for instance a sample size $k = \sqrt[3]{n^2}$ and have $\delta = \frac{1}{\sqrt[3]{k_g}}$.

From all the above, the privacy level obtained will be

$$\begin{aligned}\ln\left(e^{1/r+\delta/\lambda} + 2e^{1/\lambda - 2k_g\delta^2}\right) &= \ln\left(e^{\epsilon} + 2e^{\epsilon/1/r+1/\sqrt[3]{k_g}-2\sqrt[3]{k_g}}\right) \\ &\leq \ln\left(e^{\epsilon} + 2e^{-\sqrt[3]{k_g}}\right) \\ &\leq \epsilon + 2e^{-\sqrt[3]{k_g}}.\end{aligned}$$

Thus, we have that by adding noise randomly selected from the $Lap\left(\frac{1}{\epsilon}\left(\frac{1}{r}+\frac{1}{\sqrt[3]{k_g}}\right)\right)$ distribution to $w_2[x]$, *San* will be $\left(\epsilon+2e^{-\sqrt[3]{k_g}}\right)$-ZKP with respect to agg_k.

By substituting for the proper sensitivity and sample complexity, similar computations can be carried out for a *San* mechanism for $w_2[y]$.

Example 3. Let us consider Example 2 again with the output vector

$$f = \langle w_1(g'), w_2(g', g'')[x], w_2(g', g'')[y], w_2(g', g'')[z], w_1(g'')\rangle.$$

To privately release the second output in f, a randomized algorithm T can again uniformly select

$$k_2 = k(n)/5 = \sqrt[3]{100,000,000^2}/5 = 43,089$$

nodes and approximate the value of $w_2(g', g'')[x]$. The actual value of function $w_2(g', g'')[x]$ is computed on G. Suppose that the minimum group size in $\mathcal{S}$ is $r = 5000$ and the size of the sample group corresponding to g' in $\mathcal{G}'_{G,\mathcal{S}}$ is $k_{g'} = 30,000$. Let (δ_2, β_2) be the sample complexity of $w_2(g', g'')[x]$ where

$$\delta_2 = \frac{1}{\sqrt[3]{k_{g'}}} = \frac{1}{\sqrt[3]{30,000}} = 0.0322$$

$$\beta_2 = 2e^{-2k_{g'}\delta_2^2} = 2e^{-2*30,000*(0.0322)^2} = 1.92 * 10^{-27}.$$

The sensitivity of f is

$$\Delta(f) = \frac{1}{r} + \frac{1}{r^2} + \frac{1}{r} = 0.0004.$$

Now, if we would like to use a mechanism which is 0.1 ZKP, we can add random noise selected from a Laplace distribution with scale

$$\lambda_2 = \frac{\Delta(f) + \delta_2}{\epsilon} = \frac{0.0004 + 0.0322}{0.1} = 0.322$$

to the actual value of $w_2(g', g'')[x]$.

With this noise scale, the ZKP privacy level of the mechanism is precisely

$$\epsilon_2 \leq \left(\epsilon + 2e^{-\sqrt[3]{k_{g'}}}\right) = \left(0.1 + 2e^{-31.07}\right) \approx 0.1$$

with respect to agg_k.

6 EVALUATION

We focus on a single output $w_1(g)$ to evaluate our approach (the evaluation based on w_2 is similar). In our methods, the amount of noise added to the output is

independent of the database, and it only depends on the aggregates we compute and their sensitivities. Therefore, the following analysis is valid for any database.

6.1 Parameters Affecting Noise Scale

Considering the formula of noise scale $\lambda = \frac{\Delta(f)+\delta}{\epsilon}$, the sampling error δ is an important factor specifying λ. This error in turn has reverse connection with the sample size and the size of the database graph. Recall that throughout the chapter, we considered the error to be $\delta = \frac{1}{\sqrt[3]{k}}$, where k is the sample size with values for example $k = \sqrt[3]{n^2}$.

Figure 2 illustrates the relationship between the noise scale λ and the sample size k and the database size n. In this figure, we assumed that the output vector f has five elements and the ZKP-level ϵ is 0.1. Each curve in the figure corresponds to a sample size, namely, $k = \sqrt[3]{n^2}$ and $k = \sqrt[4]{n^3}$. The figure shows that as the graph size decreases from one billion to one million, the noise scale increases nonlinearly to the amounts that are not practical in our setting. Therefore, we conclude that ZKP mechanisms are more practical in very big databases with sufficiently large sample size.

6.2 The Noise

The analysis in this section provides a better understanding of the amount of noise which is added to the output. We consider w_1 function.

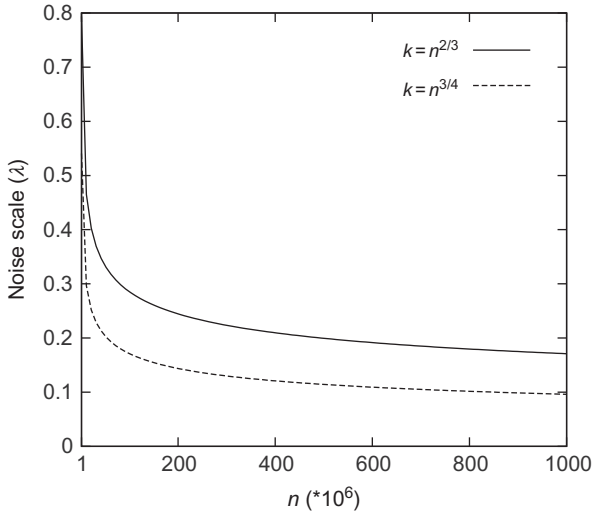


FIGURE 2 Relationship between noise scale and database size.

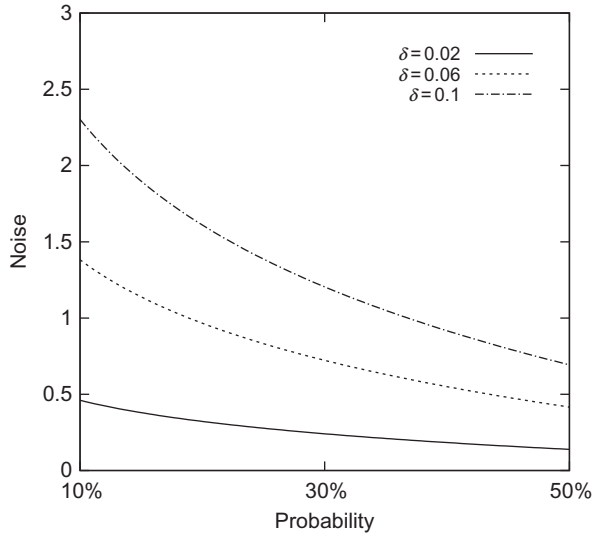


FIGURE 3 Probability versus noise.

We first compute the cumulative distribution function of Laplace distribution in an interval $[-z, z]$ as follows:

$$Pr(-z \leq x \leq z) = \int_{-z}^{z} \frac{1}{2\lambda} e^{-|x|/\lambda} dx = 1 - e^{-z/\lambda}.$$

Therefore, $Pr(|x| \geq z) = e^{-z/\lambda}$. Let $pr = Pr(|x| \geq z)$. Value z for a specified cumulative probability pr can be calculated using the above equation as

$$z = -\lambda \cdot \ln(pr) = -\frac{\delta}{\epsilon} \cdot \ln(pr).$$

Figure 3 illustrates the minimum absolute noise z as a function of cumulative probability pr for three different values of δ when $\epsilon = 0.1$. Each point (pr, z) on the curve for a given δ means that

pr *percent of the time the random noise has an absolute value of at least* z.

For example, for $\delta = 0.02$, we have that 50% of the time the absolute value of noise is at least 0.14, and 30% of the time it is at least 0.24. These values of δ are practical as our outputs are fractions.

7 FROM PRIVACY LEVEL TO NOISE SCALE

In this section, we address the problem of computing the noise scale based on the required privacy. For a given privacy level ϵ, the right value for λ can be

computed using Proposition 1 and the sample complexity of the function. For this, we need to solve the following equation with respect to λ:

$$\ln\left((1-\beta)\cdot \mathrm{e}^{(\Delta(f)+\delta)/\lambda} + \beta \cdot \mathrm{e}^{((b-a)m/\lambda)}\right) = \epsilon.$$

The sample complexity (δ, β) of the function and the sensitivity $\Delta(f)$ are computed as described in Section 5. Suppose $b-a$ and m are both equal to 1 (as in Section 5). Thus, by assigning δ a value (which depends on k), λ is the only variable in this equation. By setting $x = \mathrm{e}^{1/\lambda}$, we have the polynomial equation

$$(1-\beta)x^{\Delta(f)+\delta} + \beta x - \mathrm{e}^{\epsilon} = 0$$

which can be solved for x using various methods (cf. Goedecker, 1994; Jenkins, 1975), and finally, we have $\lambda = \frac{1}{\ln x}$.

Example 4. Let us consider Example 2 again with the output vector

$$f = \langle w_1(g'), w_2(g', g'')[x], w_2(g', g'')[y], w_2(g', g'')[z], w_1(g'')\rangle.$$

Suppose that we would like to design a (0.1)-ZKP mechanism for $w_1(g')$.

To compute the corresponding noise scale λ_1, we use the above polynomial equation. We assume that the minimum group size in $\mathcal{S}$ is $r = 5000$. The sensitivity is $\Delta(f) = \frac{1}{r} + \frac{1}{r^2} + \frac{1}{r} = 0.0004$, and we consider $k_1 = k(n)/5 = \sqrt[3]{100,000,000^2}/5 = 43,089$ (as in Example 2), i.e., $\delta = \frac{1}{\sqrt[3]{k_1}} = \frac{1}{\sqrt[3]{43,089}} = 0.0285$. We have that $w_1(g')$ has a sample complexity of

$$(\delta, \beta) = (\delta, 2\mathrm{e}^{-2k_1\delta^2}) = (0.0285, 7.08 * 10^{-31}).$$

Now if we plug all the values in the equation

$$(1-\beta)x^{\Delta(f)+\delta} + \beta x - \mathrm{e}^{\epsilon} = 0$$

we have

$$(1-7.08*10^{-31})x^{0.0004+0.0285} + (7.08*10^{-31})x - \mathrm{e}^{0.1} = 0$$

which has root $x = 31.731745$. This results in a noise scale λ_1 that is very close to (albeit slightly lower than) what we computed in Example 2. Therefore, setting the noise scale to be proportional to $\Delta(f)+\delta$ and inversely proportional to ϵ is a good enough approximation for achieving $\epsilon -$ ZKP.

8 PRIVATE PROBABILISTIC A-GS

In this section, we consider graphs with probabilistic edges. Such graphs are very common in modeling influences in social networks (cf. Bhagat et al., 2012; Budak et al., 2011; Kollios et al., 2013; Pfeiffer and Neville, 2011).

8.1 Probabilistic Graphs

We will denote a *probabilistic* graph by $G = (V, E)$, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and additionally, assigned to each edge $e \in E$, there is an existence probability $p(e) \in [0, 1]$. A probabilistic graph defines a probability distribution over a set of deterministic (regular) graphs called *possible instances* (PIs). Let $\mathcal{PI}(G)$ (or simply $\mathcal{PI}$ when G is clear from the context) be the set of all PIs of a probabilistic graph G and $PI_i(G)$ (or simply PI_i) denote one single PI. The existence probability of each PI is computed as

$$p(\text{PI}) = \prod_{e \in E(\text{PI})} p(e) \cdot \prod_{e \notin E(\text{PI})} (1 - p(e)). \tag{1}$$

8.2 Probabilistic Graph Summarization

We define the summarization of a probabilistic graph G in a similar way as for deterministic graphs. We have a set of disjoint groups of nodes, and any two groups g and g' are connected in the summary graph if at least one edge connects a node from g to some node in g'. We denote the probabilistic summary graph corresponding to G as $\mathcal{S}$-GS.

Computing the w_1 measure does not change in the probabilistic case as it is based on the existence of nodes and their attribute values which none is probabilistic. However, due to probabilistic edges, the numerators of x, y, and z of the w_2 measure are computed differently. That is, instead of computing the exact value, their *expected values* over the set of possible instances will need to be computed.

For this, let X, Y, and Z be random variables representing the x, y, and z measures, respectively. To compute $E[X]$ or $E[Z]$, we have the following theorems:

Theorem 1. *Let g and g' be two groups in a probabilistic summary graph, and let $E_{v_j} = \{e_1, \ldots, e_{n_j}\}$ be the set of edges connecting a node $v_j \in g$ to the nodes of g'. We have that*

$$E[X(g, g')] = E[X] = \frac{\sum_{v_j \in g} \left(1 - \prod_{e \in E_{v_j}} (1 - p(e))\right)}{|g|}.$$

For $E[Y]$, we have that,

Theorem 2. *Let V_g be the set of nodes in a group g and $E_{gg'} = V_g \times V_{g'}$ be the set of all possible edges between two groups g and g' in a probabilistic summary graph. We have that*

$$E[Y(g, g')] = E[Y] = \frac{\sum_{e_i \in E_{gg'}} p(e_i)}{|g| \cdot |g'|}.$$

Proofs for Theorems 1 and 2 can be found in Hassanlou et al. (2013).

8.3 Zero-Knowledge Private Probabilistic A-GS

We focus on edge (connection) privacy in this section. Let $\mathcal{G}_{G,\mathcal{S}} = (\mathcal{S}, \mathcal{E}_{\mathcal{S}})$ be the probabilistic summary graph corresponding to a graph G. Let f be a subvector that is to be privately released.

As stated before, the w_1 elements in f are computed and privatized as illustrated in Section 5.1.1. For the elements of the $E[w_2]$ functions in f, we need to view them as averages to be able to use the Hoeffding inequality in the process of privatization. We do this by defining new synthetic attributes.

Expressing $E[Y]$. For each node $v \in V$, we assume to have $|\mathcal{S}|$ new attributes called P'', each indexed by a group id. Each attribute P''_g is a vector of dimension $|g|$, where each dimension corresponds to a node in g. For a node v, we have $P''_g(v)[u] = p(e_{vu})$, where u is a node in g, and $p(e_{vu})$ is the probability of the edge between v and u. Clearly, $P''_g(v)[u] = 0$ if there is no edge between v and u in G.

For each pair of groups g and g', we have the following proposition:

Proposition 6.

$$E[Y(g,g')] = \frac{\sum_{v\in g, u\in g'} P''_{g'}(v)[u]}{|g|\cdot|g'|} = \frac{\sum_{v\in g', u\in g} P''_{g}(v)[u]}{|g|\cdot|g'|}.$$

Note that, with this expression, $E[Y]$ is the average of the elements of attribute $P''_{g'}$ over the nodes of g, or vice versa.

Expressing $E[X]$ or $E[Z]$. To be able to view $E[X]$ or $E[z]$ functions in f as averages, for each node $v \in V$, we consider $\mathcal{S}$ new synthetic attributes called P', each indexed by a group id. For each attribute P'_g, we compute the attribute value as

$$P'_g(v) = 1 - \prod_{u\in g}(1 - P''_g(v)[u]) = 1 - \prod_{u\in g}(1 - p(e_{vu})).$$

Now for each pair of groups g and g' we have the following proposition.

Proposition 7.

$$E[X(g,g')] = \frac{\sum_{v\in g} P'_{g'}(v)}{|g|}$$

$$E[Z(g,g')] = \frac{\sum_{v\in g'} P'_{g}(v)}{|g'|}.$$

Clearly, $E[X(g,g')]$ ($E[Z(g,g')]$) is now the average of $P'_{g'}$ (P'_g) attribute over the nodes of g (g').

ZKP mechanism. Let $G = (V, E)$ be a probabilistic graph augmented with synthetic attributes P's and P''s. To compute the sample complexity, a randomized algorithm T, in agg_k, samples graph G by uniformly selecting $k = k(n)/t$

random nodes from V and all their incident edges. Then, T approximates $E[w_2[.]]$ using the data from sample groups. Since we redefined the elements of $E[w_2[.]]$ as averages, we have the following inequalities for their sample complexities using the Hoeffding inequality:

$$Pr[|T(g,g')[x] - E[w_2(g,g')[X]]| \leq \delta] \geq 1 - 2e^{-2k_g\delta^2}$$
$$Pr[|T(g,g')[z] - E[w_2(g,g')[Z]]| \leq \delta] \geq 1 - 2e^{-2k_{g'}\delta^2}$$
$$Pr[|T(g,g')[y] - E[w_2(g,g')[Y]]| \leq \delta] \geq 1 - 2e^{-2(k_g \times k_{g'})\delta^2}.$$

It can be verified that the sensitivities of $E[w_2[.]]$ functions are similar to the regular case. Thus, by plugging the above parameters in Proposition 1, we have the following for the *San* mechanism of $E[X]$, where r is the minimum group size in $\mathcal{S}$ and k_g is the size of a sample group g.

Proposition 8. *By adding noise randomly selected from Lap* $\left(\frac{1}{\epsilon}\left(\frac{1}{r} + \frac{1}{\sqrt[3]{k_g}}\right)\right)$ *distribution to the output of* $E[X]$*, San will be* $\left(\epsilon + 2e^{-\sqrt[3]{k_g}}\right)$*-ZKP with respect to* agg_k.

A similar *San* mechanism can be proposed for $E[Y]$ by substituting for the sensitivity and sample complexity.

9 CONCLUSIONS

We addressed ZKP for graph summarization. We focused on group connection measures that are supported by virtually all the social-graph software products. Our techniques are crucial to be applied on summary graphs before public release of the information. We focused on ZKP mechanisms for edge privacy and introduced methods to compute the ZKP parameters. Furthermore, we presented an approach to achieve ZKP for the release of graph-summarization for probabilistic data. The upshot is that ZKP is quite useful for protecting not only the participation of a connection but also the evidence of its participation. However, from a utility point of view, ZKP can only be applied meaningfully on big social graphs.

REFERENCES

Bhagat, S., Goyal, A., Lakshmanan, L.V.S., 2012. Maximizing product adoption in social networks. In: WSDM, pp. 603–612.

Blum, A., Dwork, C., McSherry, F., Nissim, K., 2005. Practical privacy: the sulq framework. In: PODS, pp. 128–138.

Budak, C., Agrawal, D., Abbadi, A.E., 2011. Limiting the spread of misinformation in social networks. In: WWW, pp. 665–674.

Chester, S., Kapron, B.M., Ramesh, G., Srivastava, G., Thomo, A., Venkatesh, S., 2011. k-anonymization of social networks by vertex addition. In: ADBIS, pp. 107–116.

Chester, S., Kapron, B.M., Ramesh, G., Srivastava, G., Thomo, A., Venkatesh, S., 2013. Why Waldo befriended the dummy? k-Anonymization of social networks with pseudo-nodes. Soc. Netw. Anal. Min. 3 (3), 381–399.

Dwork, C., 2006. Differential privacy. In: ICALP (2), pp. 1–12.

Dwork, C., 2008. Differential privacy: a survey of results. In: TAMC, pp. 1–19.

Dwork, C., 2010. Differential privacy in new settings. In: SODA, pp. 174–183.

Dwork, C., McSherry, F., Nissim, K., Smith, A., 2006. Calibrating noise to sensitivity in private data analysis. In: TCC, pp. 265–284.

Gehrke, J., Lui, E., Pass, R., 2011. Towards privacy for social networks: a zero-knowledge based definition of privacy. In: TCC, pp. 432–449.

Gephi, 2009. An Open Source Software for Exploring and Manipulating Networks. http://gephi.org/ (Accessed on 10 January 2013).

Goedecker, S., 1994. Remark on algorithms to find roots of polynomials. SIAM J. Sci. Comput. 15 (5), 1059–1063. ISSN 1064-8275 (print), 1095-7197 (electronic).

Hassanlou, N., Shoaran, M., Thomo, A., 2013. Probabilistic graph summarization. In: WAIM, pp. 545–556.

Hay, M., Li, C., Miklau, G., Jensen, D., 2009. Accurate estimation of the degree distribution of private networks. In: ICDM, pp. 169–178.

Jenkins, M.A., 1975. Algorithm 493: zeros of a real polynomial [c2]. ACM Trans. Math. Softw. 1 (2), 178–189. ISSN 0098-3500. doi:10.1145/355637.355643. http://doi.acm.org/10.1145/355637.355643.

Kifer, D., Machanavajjhala, A., 2011. No free lunch in data privacy. In: SIGMOD Conference, pp. 193–204.

Kifer, D., Machanavajjhala, A., 2012. A rigorous and customizable framework for privacy. In: PODS, pp. 77–88.

Kollios, G., Potamias, M., Terzi, E., 2013. Clustering Large Probabilistic Graphs. IEEE Trans. Knowl. Data Eng. 25 (2), 325–336. ISSN 1041-4347.

Liu, K., Terzi, E., 2008. Towards identity anonymization on graphs. In: SIGMOD Conference, pp. 93–106.

Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M., 2007. L-diversity: privacy beyond K-anonymity. ACM Trans. Knowl. Discov. Data 1 (1). ISSN 1556-4681. http://dx.doi.org/10.1145/1217299.1217302.

Mitzenmacher, M., Upfal, E., 2005. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, New York, NY. ISBN 0521835402.

Narayanan, A., Shmatikov, V., 2009. De-anonymizing social networks. In: IEEE Symposium on Security and Privacy, pp. 173–187.

Netdriller, 2011. A Powerful Social Network Analysis Tool. http://alhajj.cpsc.ucalgary.ca/group/NetDriller/index.html (Accessed on 10 January 2013).

NodeXL, 2006. Network Overview, Discovery and Exploration for Excel. http://nodexl.codeplex.com/ (Accessed on 10 January 2013).

Pajek, 1997. Program for Large Network Analysis. http://pajek.imfm.si/doku.php?id=pajek (Accessed on 10 January 2013).

Pfeiffer, J.J., Neville, J., 2011. Methods to determine node centrality and clustering in graphs with uncertain structure. In: ICWSM.

Rastogi, V., Hay, M., Miklau, G., Suciu, D., 2009. Relationship privacy: output perturbation for queries with joins. In: PODS, pp. 107–116.

Chapter 7

Distributed Confidence-Weighted Classification on Big Data Platforms

Nemanja Djuric[*,1], **Mihajlo Grbovic**[*], **Slobodan Vucetic**[†]
[*] *Yahoo Labs, Sunnyvale, California, USA*
[†] *Temple University, Philadelphia, Pennsylvania, USA*
[1] *Corresponding author: e-mail: nemanja@yahoo-inc.com*

ABSTRACT
Recent advent of applications that generate terabytes of data on a daily basis brought numerous challenges to researchers in machine learning community, since classification algorithms designed for standard computers are incapable of addressing these large-scale problems due to memory and time constraints. As a result, there exists an evident need for novel methods that can handle such tasks. In this chapter, we present AROW-MR, a linear Support Vector Machine solver for efficient training of confidence-weighted classifiers. We propose to use MapReduce to train models in a distributed manner, obtaining large improvements in both accuracy and training time. We discuss implementation on distributed platforms such as Hadoop and Spark, and present an extensive empirical evaluation of the method using both synthetic and real-world data with nearly billion examples. The results validate large potential of Big Data platforms for development of complex, effective machine learning algorithms.

Keywords: MapReduce, Hadoop, Linear classification, Confidence-weighted classification, Online learning, Distributed platforms

1 INTRODUCTION

Explosive growth in data size, data complexity, and data rates, triggered by emergence of high-throughput technologies such as remote sensing, social networks, crowdsourcing, or computational advertising, in recent years has led

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00007-1

to an increasing availability of data sets of unprecedented scales, with billions of high-dimensional data examples stored on hundreds of terabytes of memory. This advent of large-scale applications has brought forward a clear need for computational approaches that can efficiently learn from Big Data problems (Bizer et al., 2012; Labrinidis and Jagadish, 2012; Lohr, 2012). Emerging conferences that specifically address the Big Data issues, as well as the number of recent publications related to large-scale tasks, underline the significance of the Big Data field. Moreover, recently introduced "Big Data Research and Development Initiative" by the United States (Mervis, 2012), as well as similar initiatives by the European Union (European Commission, 2014) and other world governments aimed at providing support for these efforts, clearly indicate globally recognized, strategic importance, as well as future potential and impact of Big Data-related research.

With the emergence of extremely large-scale data sets, researchers in machine learning and data mining communities are faced with numerous challenges related to the sheer size of the problems at hand, as many well-established classification and regression approaches were not designed and are not suitable for such memory- and time-intensive tasks. The inadequacy of standard machine learning tools in this new setting has led to investment of significant research efforts into the development of novel methods that can address such challenges. Classification tasks are of particular interest, as the problem of classifying input data examples into one of finite number of classes can be found in many areas of machine learning. However, state-of-the-art nonlinear classification methods, such as Support Vector Machines (SVMs) (Cortes and Vapnik, 1995), are not applicable to truly Big Data due to very high time and memory overhead, which are in general superlinear and linear in the data size T, respectively, significantly limiting their use when solving large-scale problems. Several methods have been proposed to make SVMs more scalable, ranging from algorithmic speed-ups (Kivinen et al., 2002; Nandan et al., 2014; Platt, 1998; Rai et al., 2009; Severyn and Moschitti, 2010; Tsang et al., 2005; Vishwanathan et al., 2003), to parallelization approaches (Chang et al., 2007; Graf et al., 2004; Zhu et al., 2009). However, scalability of SVM training is inherently limited as nonlinear SVMs are characterized by linear growth of model size with training data size T (Steinwart, 2003). This led to an increased interest in linear SVM models (Djuric et al., 2013b; Fan et al., 2008; Gentile, 2002; Li et al., 2002; Shalev-Shwartz et al., 2007; Wang et al., 2011; Yu et al., 2012), which have constant memory and $\mathcal{O}(T)$ training time. These linear models provide a scalable alternative to nonlinear SVMs, albeit with a certain drop in prediction accuracy.

Unfortunately, even linear time complexity may not be sufficiently efficient for modern data sets stored across petabytes of memory space, requiring researchers to develop and adopt new machine learning approaches in order to address extremely large-scale tasks. Significant research efforts culminated

in several highly influential frameworks for solving parallelizable problems that involve data sets which cannot be loaded on a single machine. These frameworks for parallel computations include MapReduce (Dean and Ghemawat, 2008, 2010), AllReduce (Agarwal et al., 2011), Spark (Zaharia et al., 2012), GraphLab (Low et al., 2010, 2012), Pregel (Malewicz et al., 2010), and others. MapReduce in particular has become very popular framework in industry, with companies such as Yahoo, Google, and Facebook spearheading its use in commercial systems (Borthakur et al., 2011; Shvachko et al., 2010).

Unlike other distributed frameworks that assume frequent communication and shared memory between the computation nodes (e.g., Agarwal et al., 2011; Low et al., 2010), MapReduce framework, and its open-source implementation called Hadoop, allows limited communication overhead between the nodes, which results in very strong fault-tolerance and guaranteed consistency. These favorable properties led to development of parallelizable variants of popular machine learning algorithms, such as perceptron (Gesmundo and Tomeh, 2012), *k*-means, logistic regression (LR), principal component analysis (Böse et al., 2010; Chu et al., 2007; Lin et al., 2011), and others. However, the proposed classification methods mostly rely on iterative training and two-way communication between the computation nodes (Chu et al., 2007; Gesmundo and Tomeh, 2012). This may impose significant costs during training as it does not closely follow the computational paradigm of MapReduce, which derives its reliability from the high level of autonomy of computation nodes. Recently, a technology called Spark (Zaharia et al., 2012) has been proposed to address these problems by maintaining data in the operating memory between the training iterations, significantly reducing latency due to I/O and deserialization costs.

In this chapter, we describe an efficient linear SVM learner with sublinear training time, capable of fully employing the MapReduce framework to significantly speed up the training. The algorithm extends ideas behind confidence-weighted (CW) linear classifiers (Crammer et al., 2009; Dredze et al., 2008) and proposes to train a number of SVM models on each of the mappers in an online fashion. Then, following completion of the map phase, the local CW models are sent to the reducer that optimally combines local classifiers to obtain a single model, more accurate than any of the individual ones. Compared to the CW algorithms, the proposed method, named AROW-MapReduce (AROW-MR), allows significantly more efficient training of accurate SVMs on extremely large data sets due to the distributed training. We validate our approach on real-world, large-scale problem of ad latency prediction with nearly 1 billion data examples, where AROW-MR achieved higher accuracy and faster training than the baseline approaches.

The chapter is organized as follows. In Section 2, we describe the CW classifiers, shown to achieve state-of-the-art performance on a number of real-world applications. In Section 3, we overview popular frameworks for distributed learning, namely MapReduce, AllReduce, and Spark. Then, in Section 4, we

describe a distributed variant of CW classifiers which can be used to efficiently train highly accurate models on large-scale problems. In Section 5, we validate our approach on synthetic data set and show that the method outperforms the baseline approaches on an industry-scale data from the computational advertising domain. Lastly, we give the concluding remarks in Section 6.

2 CLASSIFICATION WITH LINEAR SVM MODELS

In this section we briefly review linear SVM methods and introduce recently proposed CW classifiers. We detail the CW algorithm, proposed by Dredze et al. (2008), followed by the description of Adaptive Regularization of Weights (Crammer et al., 2009), an improved CW model shown to significantly outperform the original CW approach.

2.1 Linear SVM Classifiers

Support Vector Machine (Cortes and Vapnik, 1995) has grown to prominence in the past decade and is widely considered to be the state-of-the-art classification method. Given a data set $\mathcal{D}$ of size T, $\mathcal{D} = \{(\mathbf{x}_t, y_t), t = 1, \ldots, T\}$, where $\mathbf{x}_t$ is a D-dimensional feature vector describing the t^{th} training example and y_t is a corresponding binary label, $y \in \{-1, 1\}$, SVM separates positively from negatively labeled examples by finding a separating hyperplane $\mathbf{w}$ that maximizes margin between the two classes, achieved through solving the following quadratic objective function,

$$\min_{\mathbf{w}} \|\mathbf{w}\|^2 + \frac{1}{T}\sum_{t=1}^{T} \ell(\mathbf{w}^{\mathrm{T}}\mathbf{x}_t, y_t), \tag{1}$$

where $\mathbf{w}^{\mathrm{T}}\mathbf{x}_t$ is the prediction margin, and $\ell(\mathbf{w}^{\mathrm{T}}\mathbf{x}_t, y_t) = \max(0, 1 - y_t\mathbf{w}^{\mathrm{T}}\mathbf{x}_t)$ is the hinge-loss function that returns loss incurred by the prediction (e.g., positive loss for incorrect classification, $\text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_t) \neq y_t$, and 0 in the case of correct classification with prediction margin that is larger than 1).

The objective function (1) is convex; thus, there exists a unique solution to the problem that can be found by standard convex optimization methods. However, solving (1) exactly requires time superlinear in the training size T (Joachims, 2006), which makes such approach infeasible on the large-scale data sets commonly found in practice. To mitigate this problem, a number of efficient, approximate linear SVM solvers have been proposed, which reduce the time complexity to $\mathcal{O}(T)$ (Joachims, 2006; Shalev-Shwartz et al., 2011). In the following section, we describe such method, CW classifier, shown to learn highly accurate linear SVM models in time linear to the training set size. Unlike similar linear solvers, CW method explicitly models classification probability, thus allowing more flexible and accurate modeling of the problem at hand.

2.2 CW Classification

First described by Dredze et al. (2008), the CW algorithm is a linear SVM classifier that, in addition to the prediction margin for the new data example, also outputs probability of the correct classification. This is achieved by maintaining a multivariate Gaussian distribution over the separating hyperplanes $\mathbf{w}$,

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{2}$$

and during the training procedure both mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the distribution are learned. In this way, a more expressive and informative model is found, giving us information about noise in each of the individual features, as well as about the relationships between features.

Let us assume that a trained CW model, with known mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, is given. For an example $(\mathbf{x}, y)$ from data set $\mathcal{D}$, described by feature vector $\mathbf{x}$ and binary label y, this induces a Gaussian distribution over the prediction margin $\hat{y}$ as follows,

$$\hat{y} \sim \mathcal{N}\left(y\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}, \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{x}\right). \tag{3}$$

Then, using Eq. (3), we can compute the probability of correct classification by employing the equation for the normal cumulative distribution function to obtain the following expression,

$$\mathbb{P}(\mathrm{sign}(\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}) = y) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{y\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}}{\sqrt{2\mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}\mathbf{x}}}\right)\right). \tag{4}$$

The CW classifier is learned online, and the current model is updated each round after observing one training example. During training, our belief about the classifier before the t^{th} training iteration, expressed through the current mean $\boldsymbol{\mu}_{t-1}$ and the current covariance matrix $\boldsymbol{\Sigma}_{t-1}$, is updated so that the new example $(\mathbf{x}_t, y_t)$ is correctly classified with probability larger than some user-defined parameter η. In addition, we impose an additional constraint that our new belief after iteration t is not too far from our belief before the iteration t. More formally, the stated requirements yield the following optimization problem,

$$\begin{aligned} &(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = \underset{\boldsymbol{\mu}, \boldsymbol{\Sigma}}{\arg\min}\, D_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \parallel \mathcal{N}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})\right) \\ &\text{subject to } \mathbb{P}\left(y_t\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}_t \geq 0\right) \geq \eta, \end{aligned} \tag{5}$$

where D_{KL} is the Kullback–Leibler (KL) divergence. Since (5) is nonconvex, the authors of Dredze et al. (2008) solve an approximate convex problem and derive closed-form updates for the parameters of the Gaussian distribution.

As formulated in (5), we seek such an update of the classification model so that the new training example is correctly classified with certain probability η. In Crammer et al. (2009), the authors point out that this may be suboptimal for noisy data sets. More specifically, once the learning algorithm observes a noisy example, the update would modify the current model so that the noise is

correctly classified, which could have an adverse effect on the generalization performance of the classifier. To address this issue, a new CW formulation is proposed (Crammer et al., 2009), called Adaptive Regularization of Weights (AROW). In this approach, the following problem is solved,

$$\begin{aligned}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t) = &\arg\min_{\boldsymbol{\mu},\boldsymbol{\Sigma}} D_{\mathrm{KL}}\big(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \parallel \mathcal{N}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})\big) \\ &+ \lambda_1\big(\max(0, 1 - y_t \boldsymbol{\mu}^{\mathrm{T}} \mathbf{x}_t)\big)^2 + \lambda_2(\mathbf{x}_t^{\mathrm{T}} \boldsymbol{\Sigma} \mathbf{x}_t).\end{aligned} \tag{6}$$

We can see that at each training step, the old and the new belief are still constrained to be close, as measured by the KL divergence. However, unlike in the CW algorithm which aggressively updates the model in order to accommodate new examples, in AROW formulation the aggressiveness of maximization of margin and minimization of uncertainty for the new example are controlled by the regularization parameters λ_1 and λ_2, respectively. As shown by Crammer et al. (2009), after finding the derivative of the objective function with respect to the parameters, update equations for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be written in a closed form,

$$\begin{aligned}\boldsymbol{\mu}_t &= \boldsymbol{\mu}_{t-1} + \alpha_t y_t \boldsymbol{\Sigma}_{t-1} \mathbf{x}_t, \\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_{t-1} - \beta_t \boldsymbol{\Sigma}_{t-1} \mathbf{x}_t \mathbf{x}_t^{\mathrm{T}} \boldsymbol{\Sigma}_{t-1},\end{aligned} \tag{7}$$

where α_t and β_t are computed as

$$\begin{aligned}\alpha_t &= \beta_t \max(0, 1 - y_t \boldsymbol{\mu}_{t-1}^{\mathrm{T}} \mathbf{x}_t), \\ \beta_t &= (\mathbf{x}_t^{\mathrm{T}} \boldsymbol{\Sigma}_{t-1} \mathbf{x}_t + r)^{-1},\end{aligned} \tag{8}$$

and $r = 1/(2\lambda_1)$, for $\lambda_1 = \lambda_2$. Online AROW training is initiated with a zero-vector $\boldsymbol{\mu}_0$ and an identity matrix $\boldsymbol{\Sigma}_0$, and it further proceeds to observe training examples and iteratively update the model parameters following (7) and (8).

3 MAPREDUCE FRAMEWORK FOR DISTRIBUTED COMPUTATIONS

With the recent explosive growth of data set sizes, analysis and knowledge extraction from modern large-scale data sets using a single machine are becoming increasingly intractable. In particular, training time of popular classification and regression methods (e.g., SVMs, classification trees) is at best linear in training set size, which may be too expensive for problems with billions of examples. To address this pressing issue, a number of frameworks for distributed learning on clusters of computation nodes has been introduced, offering different levels of parallelization, node independence, and reliability (Agarwal et al., 2011; Dean and Ghemawat, 2008, 2010; Low et al., 2010, 2012; Malewicz et al., 2010; Zaharia et al., 2012). In this section, we describe such a framework

which has become ubiquitous in the industry, called MapReduce. In addition, we discuss related AllReduce distributed framework, which is utilized in the Vowpal Wabbit (VW) software package, an open-source machine learning library for state-of-the-art distributed learning, as well as Spark framework, a recently proposed technology that generalizes MapReduce and is becoming increasing popular both in the industry and in the academic setting (Xin et al., 2013a,b; Zaharia et al., 2012).

MapReduce framework (Dean and Ghemawat, 2008, 2010), implemented as an open-source platform Hadoop[1], consists of two distinct phases called *map* and *reduce*, which constitute one MapReduce *job*. In the map phase, computing nodes that are referred to as mappers read parts of the data set (possibly stored on multiple computers) and perform some action (e.g., filtering, grouping, counting, sorting) with final results being sent to a computing node referred to as a reducer in a form of ordered (*key*, *value*) pairs. In the reduce phase, reducer performs a summary operation on the data received from the mappers, where the received data is sorted by their key values. There may be multiple mappers and reducers, and the framework guarantees that all values associated with a specific key will appear in one and only one reducer. Note that the limited communication between computation nodes in MapReduce framework, which is allowed only from mappers to reducers, ensures high independence of mappers and significant fault tolerance of the framework. Even in the case of mapper failure the entire job is not significantly affected as remaining mappers are not aware of the failure, which can be fixed by a simple restart of the failed node.

We elucidate MapReduce abstraction using a simple example, illustrated in Fig. 1. Given a text document, we may want to find how many times each word appears in order to represent the document using a bag-of-words scheme, which can be achieved using several mappers and a single reducer. Each mapper reads a part of the document in a streaming fashion (in our example, each mapper scans a single line of the text) and outputs (w, 1) as soon as the word w is found. When the mappers finish outputting (*key*, *value*) pairs, the reducer starts reading these pairs sorted by their key (i.e., sorted alphabetically in our example). Then, on the reducer side, we initialize the word count variable to 0 and read one pair at a time, adding all values associated with the same key as the ordered pairs are received.

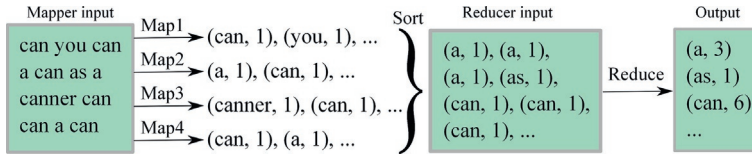


FIGURE 1 Example of word counting task solved using the MapReduce framework.

1. http://hadoop.apache.org/, accessed February 2015.

Once a key that is different from the one associated with the current count is read (e.g., we observed word "as" after three observations of word "a"), reducer outputs the word and its total count and resets the count variable to compute the number of occurrences of the new word. In this way, there is no need to store individual words or their counts in the operating memory, which significantly lowers memory costs of the reducer.

Although we used a very simple example to illustrate the idea behind MapReduce, this framework is very powerful and can be used to solve complex, large-scale tasks in machine learning domain. There are several ways of utilizing MapReduce paradigm for distributed learning:

- read the training data using multiple mappers, and learn the model on a reducer in an online learning manner;
- maintain a global model that is used by all mappers to compute partial training updates, which are aggregated on the reducer and used to update the global model (note that this approach requires running multiple, consecutive MapReduce jobs to ensure convergence of the model);
- learn a local model on each of the mappers, and combine the mapper-specific models into a global one on a reducer.

For the first option, distributed learning takes the same amount of time as learning on a single machine, with the benefit that there is no need to store the data on a single machine. The second option is typically used for batch learning (Chu et al., 2007; Gesmundo and Tomeh, 2012; Lin et al., 2011), where each mapper computes partial gradient using the current model, while the reducer sums the partial gradients and updates the model. A new MapReduce job is then instantiated, with the updated model used by all mappers for the next round of gradient calculation. Thus, one job is analogous to one gradient descent step of the model optimization. Since learning process may require several iterations to converge, multiple MapReduce jobs need to be ran one after another, which may be ineffective and costly as starting and ending each job requires nonnegligible time costs. In such cases, much better alternative is the Spark framework which is designed and optimized for iterative training of machine learning algorithms, which we discuss in more detail in Section 3.2. In contrast, the third approach ensures more robust learning and very limited communication overhead as only a single job is run, and we utilize this approach to propose an efficient and accurate classifier in Section 4.

3.1 AllReduce Framework

The MapReduce abstraction allows for very limited interaction between the computation nodes, which ensures very robust and fault-tolerant system. In the following, we introduce significantly less-constrained framework called AllReduce (Agarwal et al., 2011), which is utilized by the popular VW software

package[2] (Langford et al., 2009). Unlike MapReduce, AllReduce framework assumes communication between mappers as well, while the reducers are not used. In particular, each mapper maintains a copy of the global model. Then, when computing the update step for the current model, partial update step computed on one mapper is communicated to all other mappers. Once every mapper receives a message from all other mappers, the aggregated update step is performed on all computation nodes simultaneously, each updating their local model copy with exactly the same update step. As a result, copies of the model that are stored locally at each mapper remain identical throughout the training. A typical implementation of AllReduce is done by imposing a tree structure on the computation nodes which limits the number of exchanged messages between the nodes such that the partial messages are aggregated up the tree and then broadcasted down to all mappers.

Disadvantage of the AllReduce framework is that the mappers need to run truly concurrently. However, it is common for large clusters to run many independent jobs initiated by different users, each requiring different amount of resources on the computation nodes. Consequently, for higher number of mappers there may be no guarantee that all of them will be available for concurrent execution. Furthermore, as we have observed in practice, due to the fact that all nodes are required to send their updates before the next learning iteration starts, AllReduce learning will stall if any individual mapper fails once the job has started, which may present a significant limiting factor for a user attempting to solve extremely large-scale problems.

3.2 Spark Framework

Despite several attempts to parallelize iterative training of machine learning algorithms using MapReduce (e.g., Böse et al., 2010; Chu et al., 2007; Lin et al., 2011), the framework remains inherently unsuitable for implementation of iterative batch training due to large costs related to start-up process of an iteration. In particular, each training run is implemented as a single MapReduce job, which is required to load the entire data from the disk in order to process it, repeatedly incurring the same I/O and communication costs at each iteration. This results in significant recurring costs, which slow down the training and render MapReduce framework a suboptimal choice for batch training of machine learning methods.

To address the issues with excessive overhead when repeatedly accessing the same data, a distributed framework called Spark[3] has been recently proposed by Zaharia et al. (2012), built around a distributed data abstraction termed Resilient Distributed Datasets (RDDs). A critical aspect of RDD is *data persistence*,

2. https://github.com/JohnLangford/vowpal_wabbit, accessed February 2015.
3. https://spark.apache.org/, accessed February 2015.

where, unlike in the MapReduce paradigm, computation nodes are capable of maintaining loaded data in operating memory between iterations, thus avoiding recurring start-up costs and significantly speeding up iterative processing of data. Another key contribution of Spark is an introduction of the concept of *data lineage*. More specifically, let us assume that the original data was processed by the mappers in some way (e.g., a user filtered all rows with NULL fields), and each mapper now maintains a modified version of the original data. Then, for the current state of the data on a particular computation node, a sequence of coarse-grained operations that led to that particular state is saved (i.e., sequence of *maps*, *filters*, *joins*, and other operations that operate on the entirety of the data). If a computation node fails, we can apply the saved sequence of operations only on a subset of the original data that is necessary to fully recreate the lost chunk of data from the failed node, thus avoiding expensive replication of the intermediate states of the data as a way to enforce fault tolerance. The RDD abstraction results in faster iterative data processing, and the authors report up to 20× faster execution of k-means and LR when compared to their Hadoop implementations (Zaharia et al., 2012).

Interestingly, it can be shown that Spark is a generalization of MapReduce and also subsumes many other distributed frameworks, such as Pregel or SQL (Zaharia et al., 2012). However, in the case of distributed online training of machine learning algorithms, as opposed to distributed batch training, Spark does not exhibit advantage over MapReduce. During online training, the data is loaded and observed only once, and the persistence aspect of Spark, which brings most of the speed-up to the framework, does not come into play. Thus, for our use case which involves online training of classification models, both Spark and MapReduce are equally effective. As a result, we choose to employ Hadoop as our distributed platform, as it is a more mature project than Spark at the moment, with much more resources and documentation to help in development and prototyping of novel distributed algorithms. In the following section we describe implementation details of distributing CW classifiers using the MapReduce framework.

4 CW CLASSIFICATION USING MAPREDUCE

In this section we present a distributed AROW algorithm, which can be used to efficiently train very accurate linear classifiers on large-scale data. Let us assume that we have M mappers, and on each mapper an AROW model is trained using only a subset of the whole data set $\mathcal{D}$, by running the algorithm described in Section 2. More specifically, on the m^{th} mapper an AROW model is trained using a data set $\mathcal{D}_m \subset \mathcal{D}$ such that $\bigcup_{m=1,\ldots,M} \mathcal{D}_m = \mathcal{D}$ and $\mathcal{D}_i \cap \mathcal{D}_j = \varnothing, i \neq j$. We denote the trained AROW parameters for the m^{th} mapper as $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$, which are sent to the reducer after the completion of the map stage. During the reduce stage, we learn the final, aggregated parameters $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ such that the

multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ is an optimal combination of M multivariate Gaussian distributions learned on mappers.

More formally, let us define the following objective function $\mathcal{L}$ that is to be minimized on the reducer,

$$\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[D_{\mathrm{KL}}^{\mathrm{S}}\big(\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \parallel \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\big)], \tag{9}$$

where the expectation is taken over the distributions over hyperplanes that separate the data set $\mathcal{D}$, and $D_{\mathrm{KL}}^{\mathrm{S}}$ is the symmetric KL divergence, defined as

$$D_{\mathrm{KL}}^{\mathrm{S}}(A\|B) = \frac{1}{2}\big(D_{\mathrm{KL}}(A\|B) + D_{\mathrm{KL}}(B\|A)\big). \tag{10}$$

As can be seen from (9), the reducer computes aggregated parameters $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$ such that the expected symmetric KL divergence between the aggregated Gaussian distribution and hyperplane distributions, drawn from the probability distribution over separating hyperplane distributions for the data set $\mathcal{D}$, is minimized. We note that the proposed method can be viewed as a generalization of the averaging CW model used for large-scale data sets, briefly discussed in Dredze et al. (2008). Given the mapper-specific parameters $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m, m = 1, \ldots, M$, empirical estimate of the objective function $\mathcal{L}$ can be expressed as follows,

$$\mathcal{L} = \sum_{m=1}^{M} \mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big)\, D_{\mathrm{KL}}^{\mathrm{S}}\big(\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \parallel \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big), \tag{11}$$

where we define $\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big)$, or probability of the m^{th} distribution over the separating hyperplanes, as the fraction of the training set used to train the m^{th} AROW classifier. We refer to the final CW classification model as AROW-MR.

4.1 Reducer-Side Optimization of AROW-MR

The optimization function (11) is convex; thus, there exists a unique set of $(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$ parameters that minimize $\mathcal{L}$. In this section, we derive update equations for AROW-MR parameters, the mean and the covariance matrix of the aggregated Gaussian distribution over the separating hyperplanes.

In order to solve (11), we compute the first derivatives of the objective function $\mathcal{L}$ with respect to the parameters of the aggregated Gaussian distribution. After finding the derivative of $\mathcal{L}$ with respect to $\boldsymbol{\mu}_*$ and equating the resulting expression with 0, we obtain the following update rule for mean $\boldsymbol{\mu}_*$,

$$\begin{aligned}\boldsymbol{\mu}_* = &\Big(\sum_{m=1}^{M} \mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big)\,(\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_m^{-1})\Big)^{-1} \\ &\Big(\sum_{m=1}^{M} \mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big)\,(\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_m^{-1})\boldsymbol{\mu}_m\Big).\end{aligned} \tag{12}$$

In order to compute the update rule for covariance matrix, we find the derivative of $\mathcal{L}$ with respect to $\boldsymbol{\Sigma}_*$ and equate the resulting equation with 0. After derivation, we obtain the following expression,

$$\begin{aligned}\boldsymbol{\Sigma}_*\Big(\sum_{m=1}^{M}\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\big)\,\boldsymbol{\Sigma}_m^{-1}\Big)\boldsymbol{\Sigma}_* = \\ \sum_{m=1}^{M}\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\big)\,\big(\boldsymbol{\Sigma}_m+(\boldsymbol{\mu}_*-\boldsymbol{\mu}_m)(\boldsymbol{\mu}_*-\boldsymbol{\mu}_m)^{\mathrm{T}}\big).\end{aligned} \tag{13}$$

Equation (13) is a Riccati equation of the form $\mathbf{XAX}=\mathbf{B}$, solved with respect to matrix $\mathbf{X}$ with matrices $\mathbf{A}$ and $\mathbf{B}$ given. After finding the decomposition of matrix $\mathbf{A}$ as $\mathbf{A}=\mathbf{U}^{\mathrm{T}}\mathbf{U}$ (e.g., using the Cholesky decomposition), we can compute $\mathbf{X}$ in a closed form using the following steps,

$$\begin{aligned}\mathbf{XAX}&=\mathbf{B},\\ \mathbf{XU}^{\mathrm{T}}\mathbf{UX}&=\mathbf{B},\\ \mathbf{UXU}^{\mathrm{T}}\mathbf{UXU}^{\mathrm{T}}&=\mathbf{UBU}^{\mathrm{T}},\\ (\mathbf{UXU}^{\mathrm{T}})^2&=\mathbf{UBU}^{\mathrm{T}},\\ \mathbf{UXU}^{\mathrm{T}}&=\mathbf{U}^{0.5}\mathbf{B}^{0.5}(\mathbf{U}^{\mathrm{T}})^{0.5},\\ \mathbf{X}&=\mathbf{U}^{-0.5}\mathbf{B}^{0.5}(\mathbf{U}^{\mathrm{T}})^{-0.5}.\end{aligned} \tag{14}$$

By matching the elements of Eq. (14) with the elements of Eq. (13), we can find the closed-form solution for the covariance matrix $\boldsymbol{\Sigma}_*$. In particular, if we compute the Cholesky decomposition of the sum on the left-hand side of Eq. (13) as follows,

$$\sum_{m=1}^{M}\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\big)\,\boldsymbol{\Sigma}_m^{-1}=\mathbf{U}^{\mathrm{T}}\mathbf{U}, \tag{15}$$

the resulting closed-form expression for the $\boldsymbol{\Sigma}_*$ matrix can be written as

$$\boldsymbol{\Sigma}_*=\mathbf{U}^{-0.5}\Big(\sum_{m=1}^{M}\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)\big)\,\big(\boldsymbol{\Sigma}_m+(\boldsymbol{\mu}_*-\boldsymbol{\mu}_m)(\boldsymbol{\mu}_*-\boldsymbol{\mu}_m)^{\mathrm{T}}\big)\Big)^{0.5}(\mathbf{U}^{\mathrm{T}})^{-0.5}. \tag{16}$$

Then, in order to find the optimal parameters $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$, Eqs. (12) and (16) are solved iteratively until convergence (we empirically found that only a few iterations are sufficient for the optimization procedure to converge), where we initialized $\boldsymbol{\mu}_*$ to the average of mapper-specific mean vectors $\boldsymbol{\mu}_m$, with $m=1,\ldots,M$, and $\boldsymbol{\Sigma}_*$ to an identity matrix. Pseudocode given in Algorithm 1 summarizes training steps of the AROW-MR algorithm.

Algorithm 1 AROW-MapReduce (AROW-MR)

Inputs: D-dimensional data set $\mathcal{D}$, regul. parameter r, no. of mappers M;
Outputs: Parameters of AROW-MR model, vector $\boldsymbol{\mu}_*$ and matrix $\boldsymbol{\Sigma}_*$;

1. **Map:** Train the m^{th} AROW classifier on local subset $\mathcal{D}_m$ of the data set $\mathcal{D}$ in order to obtain $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$, where $m = 1, \ldots, M$:
   (1) Initialize $\boldsymbol{\mu}_0$ to zero-vector and $\boldsymbol{\Sigma}_0$ to identity matrix;
   (2) Observe a data point $(\mathbf{x}_t, y_t)$ from the local subset and iteratively update local model using Eqs. (7) and (8):
   $$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \alpha_t y_t \boldsymbol{\Sigma}_{t-1}\mathbf{x}_t,$$
   $$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t-1} - \beta_t \boldsymbol{\Sigma}_{t-1}\mathbf{x}_t\mathbf{x}_t^{\mathrm{T}}\boldsymbol{\Sigma}_{t-1},$$
   where $\alpha_t = \beta_t \max(0, 1 - y_t\boldsymbol{\mu}_{t-1}^{\mathrm{T}}\mathbf{x}_t)$,
   $\beta_t = (\mathbf{x}_t^{\mathrm{T}}\boldsymbol{\Sigma}_{t-1}\mathbf{x}_t + r)^{-1}$;
   (3) Once the local data subset is exhausted, send the final model parameters $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ computed locally to the reducer;
2. **Reduce:** Combine the local AROW classifiers into an aggregated AROW classifier using Eqs. (12), (15), and (16):
   (1) Compute $\mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big), m = 1, \ldots, M$, as a size of the subset $\mathcal{D}_m$ divided by a size of the entire data set $\mathcal{D}$;
   (2) Compute matrix $\mathbf{U}$ of the Cholesky decomposition:
   $$\textstyle\sum_{m=1}^{M} \mathbb{P}\big(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)\big)\ \boldsymbol{\Sigma}_m^{-1} = \mathbf{U}^{\mathrm{T}}\mathbf{U};$$
   (3) Initialize matrix $\boldsymbol{\Sigma}_*$ to the identity matrix and vector $\boldsymbol{\mu}_*$ as
   $$\boldsymbol{\mu}_* = M^{-1} \textstyle\sum_{m=1}^{M} \boldsymbol{\mu}_m;$$
   (4) Iteratively compute aggregated parameters until convergence:
      (4a) Update mean vector $\boldsymbol{\mu}_*$ using Eq. (12);
      (4b) Update covariance matrix $\boldsymbol{\Sigma}_*$ using Eq. (16);
3. **Output:** Aggregated AROW-MR parameters $\boldsymbol{\mu}_*$ and $\boldsymbol{\Sigma}_*$.

Let us discuss the time complexity of AROW and its distributed version AROW-MR. Assuming a D-dimensional data set $\mathcal{D}$ of size T, complexity of AROW training amounts to $\mathcal{O}(TD^2)$. On the other hand, complexity of AROW-MR is $\mathcal{O}(TD^2/M + MD^2 + D^3)$, where the first term is due to local AROW training on mappers and the second and the third term are due to reducer optimization, which involves summation over M matrices of size $D \times D$ and Cholesky decomposition of the result, respectively. For large-scale data, for which we assume $T \gg M$, we can conclude that AROW-MR offers efficient training with significantly lower time overhead when compared to the AROW algorithm.

5 EXPERIMENTS

In this section we present the results of empirical evaluation of the AROW-MR algorithm. We first validated our method on a synthetic data, then explored its performance on a real-world, industry-scale task of ad latency prediction.

5.1 Validation on Synthetic Data

In order to better characterize the proposed distributed algorithm, in the first set of experiments we compared AROW and AROW-MR algorithms on synthetic data. We used the *waveform* data set generator, available from the UCI Repository (Bache and Lichman, 2013), where we labeled the first and the second class as being positive and the third class as being negative. We generated 50,000 training examples and 5000 test examples and set regularization parameters $\lambda_1 = \lambda_2 = 0.05$ through cross-validation. We repeated experiments 10 times, where in each iteration we split the training set into M disjoint subsets of equal sizes and used one subset to train a local AROW model on one mapper, setting the number of mappers M from 2 to 200 in increments of 5 in order to better evaluate the effect of higher levels of parallelization.

In addition to AROW-MR, we report the results of the original AROW which used the entire training data set (denoted by AROW, which was not affected by the number of mappers), as well as the results of a local AROW model trained on a single mapper, denoted by AROW-single, for which the number of training examples decreased as the number of mappers was increased (i.e., number of training examples for each local model was $50{,}000/M$). We included AROW-single results to illustrate the performance of local AROW models that are eventually combined on the reducer to obtain AROW-MR model. Experiments on synthetic data were run in Matlab, on Lenovo IdeaPad Y500 with 2.4 GHz Intel Core i7 with 8 GB of DDR3 memory. The training times and mean accuracy are shown in Fig. 2a and b, respectively, where the error bars represent intervals of two standard deviations (we omitted error bars for accuracy of AROW-single as the standard deviation was around 0.5 and error bars would clutter the figure).

As illustrated in Fig. 2a, distributed training resulted in a significant speed-up in training time. We can see that AROW-MR training is an order of magnitude faster than training of AROW, while at the same time achieving higher accuracies, as shown in Fig. 2b. Interestingly, the training time did not decrease further as we increased the number of mappers beyond a certain point. Although the mapper time continued to drop (as shown by the bottom-most line labeled AROW-single), this was countered by longer time spent to solve the optimization problem (11) in the reduce phase due to larger M. This validates the known result that for certain problems "too much parallelization or distribution can have negative consequences" (Hughes and Hughes, 2004), and that the level of parallelization should be determined after deeper analysis and understanding of the problem being solved.

Furthermore, we can see in Fig. 2b that the accuracy of AROW-MR initially increased as the number of mappers increased, statistically significantly outperforming AROW. The accuracy of local AROW models trained on each mapper (shown as AROW-single) dropped steadily with the increase of the number of mappers, which was expected as less training examples were used. Interestingly,

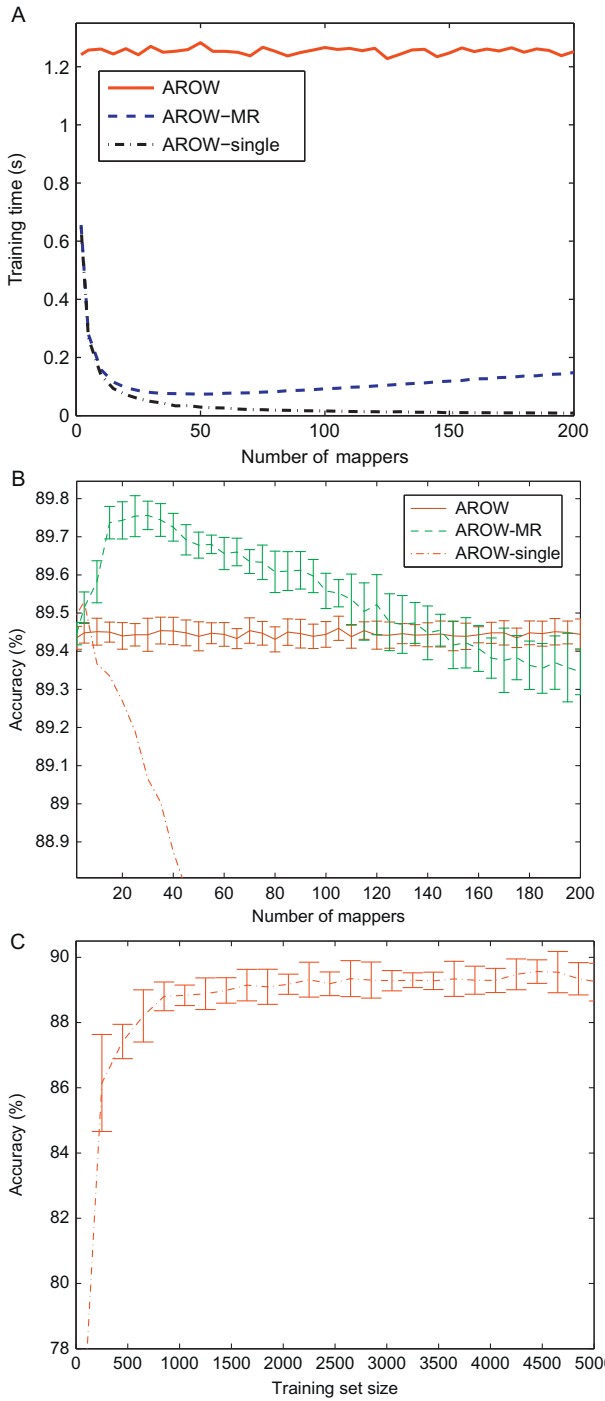


FIGURE 2 (a, b) Performance of AROW and AROW-MR on the *waveform* data trained on 50,000 examples (AROW-single illustrates performance of a single mapper of AROW-MR). (c) Accuracy of AROW as a function of the training set size.

as the number of mappers further increased, we can see that the accuracy of AROW-MR started decreasing when M surpassed 40, until it reached the accuracy of AROW for $M = 140$. This decrease is due to the fact that there were too few training examples on mappers for the local models to be close to convergence, which in turn affected accuracy of the aggregated model.

We further investigate the convergence rate of AROW as a function of the number of training points and give the results in Fig. 2c. Note that the results in the figure correspond to the results of AROW-single, shown in Fig. 2a and b (i.e., AROW-single with 50 mappers corresponds to AROW with 1000 training points, as 50,000 training points/50 mappers = 1000 training points per mapper), where Fig. 2c illustrates AROW performance in much more detail. The figure shows that AROW converges to accuracy of 89.5% at around 5000 points, which corresponds to AROW-single with 10 mappers in Fig. 2b. On the other hand, for training sizes below 500 the results indicate that the performance drops quickly, reaching only 82% accuracy for 250 training points. Nevertheless, we can see that the corresponding AROW-MR with 200 mappers reached significantly higher accuracy of 89.35%. This result suggests that the proposed AROW-MR approach is very robust, combining a number of suboptimal models into an aggregated model that yielded state-of-the-art classification performance, in addition to significantly faster training time.

5.2 Ad Latency Problem Description

Having validated the proposed approach on the synthetic data, in this section we compare performance of AROW-MR and the baseline methods on large-scale, industrial task of predicting ad latency. However, before moving on to the discussion of empirical results, we first introduce this important problem in online advertising, as well as the large-scale data set used in the experiments.

Over the previous decade, income generated by Internet companies through online advertising has been growing steadily at amazing rates, with the total revenue reaching a record $36.6 billion in the United States in 2012 alone.[4] This burgeoning, highly competitive market consists of several key players: (1) advertisers, companies that want to advertise their products; (2) publishers, websites that host advertisements; and (3) intermediate players that connect advertisers to publishers. In some cases such clear segmentation is not possible, and certain companies can be found in more than one role (e.g., Yahoo or Google may provide both the products and the advertising space). Typically, advertiser designs an image of the advertisement, called a creative, specifying size and dimension requirements of the image to be shown on websites. This is then sent to the intermediate companies which have contracts with publishers, and which decide when and to whom the ads will be shown in order to maximize profits.

4. news.yahoo.com/us-internet-ad-revenue-grows-15-percent-2012-153752947, February 2015.

In order to retain existing and attract new users, publishers aim at improving user experience by minimizing page load times. In addition, equally important task for publishers is to ensure that the ads are delivered on time. Considering that ad latency time accounts for a significant percentage of the overall load time, improving ad load times would directly benefit both the user experience and the website revenue. Thus, correctly predicting ad latency time, and using this information to decide if and which ad should be shown to a user, is an extremely important problem in online advertising where an additional latency of several milliseconds could result in a significant loss of revenue. In this section we considered ad latency data set consisting of nearly 1.3 billion ad impressions, for which 21 features were measured at serve time, along with ad latency given in milliseconds. Features can be divided into several groups:

- *Time-specific features* originate from the timestamp of an impression, using time of day and day of week as ad impression features.
- *Advertiser-specific features* include the advertiser's account ID, size of the advertisement (2-D of the creative and its size in kilobytes), as well as the creative ID of a specific image used by the advertiser.
- Publisher's website can be partitioned into several regions, where each region has several spaces on which the ad can be shown. Further, ad can be placed at several different positions in the space (e.g., top, bottom). Thus, *publisher-specific features* include region ID, space ID, position, ad network used to serve the ad, serve type, hostname, as well as collocation center used to serve the ad.
- Lastly, *user-specific features* include user's device type, operating system, and browser. We also used user's geographic location (i.e., state and city), user's physical distance from the collocation center serving the ad, user's connection speed (e.g., broadband, dial-up), as well as Internet service provider used by the user.

In Table 1 we give an overview of the data set features, as well as the cardinalities for discrete features (note that we omitted business-sensitive information, which is marked with the "×" symbol).

We can represent the latency prediction problem as a binary classification by thresholding the value of ad latency. An ad is considered late (i.e., labeled positive as $y = 1$) if the time period from the moment when the web page loads to the moment when the ad renders is longer than k milliseconds, and not late otherwise (i.e., labeled negative as $y = -1$). Value of k can be selected depending on a product or ad campaign requirements, and we omit the specific value used in the experiments as it represents a sensitive information.

5.3 Validation on Ad Latency Data

In order to evaluate performance of the classification algorithms, we randomly split the data set into training set, consisting of 997,055,154 examples, and

TABLE 1 Description of Features Used to Represent Ad Impressions in the Ad Latency Data

Type	Feature Name (*Cardinality*)
Time	Hour of the day (*24*)
	Day of the week (*7*)
Advertiser	Account ID (**x**)
	Creative ID (**x**)
	Ad size (dimensions) (*33*)
	Ad size in kilobytes (*real-valued*)
Publisher	Region ID (**x**)
	Space ID (location on the page) (**x**)
	Ad position (*28*)
	Hostname (**x**)
	Ad network (**x**)
	Serve type (**x**)
	Collocation center (**x**)
User	Device (*15*)
	Operating system (*22*)
	Browser (*100*)
	Connection speed (*10*)
	U.S. state (*50*)
	City (*574*)
	ISP (**x**)
	Distance to colloc. center (*real-valued*)

Business-sensitive feature cardinalities are replaced with a "**x**" sign.

nonoverlapping testing set with 279,026,428 labeled examples. For the ad latency prediction task, the publishers prefer low false-positive rate (FPR or 1-specificity), ensuring that very few ads are wrongly classified as late, thus minimally hurting revenue. At the same time, the publishers prefer high true-positive rate (TPR or sensitivity), which improves user experience and increases profit. As these two goals are often conflicting, the optimal strategy is to maximize the area under the receiver operating characteristic (ROC) curve, referred to as the

AUC (Fawcett, 2004). Thus, unlike in the experiments with the synthetic data, we report AUC as a measure of performance. Given a predicted margin for the t^{th} example, $\hat{y}_t \in \mathbb{R}$, a binary classification prediction is found as $\text{sign}(\hat{y}_t - \theta)$, where different values of threshold θ result in different predictions and different TPR and FPR values. We can obtain an ROC curve by sliding the threshold θ from $-\infty$ to ∞ and plotting the resulting TPR and FPR against each other in a 2-D graph.

The experiments were conducted using MapReduce on Apache Hadoop, with AROW-MR mapper and reducer implemented in Perl. Performance of AROW-MR was compared to AROW algorithm which was run on a single machine, as well as to the LR implemented in highly scalable VW package, run both on a single machine and in a distributed manner using AllReduce on Hadoop. We ensured that Hadoop scheduled exactly M mappers by storing the data in M gzip-compressed files. The files compressed using gzip codec are nonsplittable (unlike, for example, bz2-compressed files), and such setup thus resulted in exactly one mapper per a compressed file.

We first compared the performance of AROW learned on a single machine with AROW-MR learned in a distributed manner. Similar to experiments presented in Section 5.1, we increased the number of mappers to evaluate the effects of different levels of parallelization. The results are given in Table 2. We can see that the running time of AROW-MR drastically improved over the nondistributed AROW (trained using a single mapper). While it took 17 days for AROW to train, we were able to train accurate AROW-MR models in less than an hour. Expectedly, average mapper time decreased and reducer time increased as the number of mappers was increased, as each mapper was trained on smaller partition of the data and reducer was required to combine higher number of local models. Interestingly, we can also see that the results in Table 2 validate the results obtained on synthetic data regarding performance gains with increasing levels of parallelization. In particular, both the accuracy and training

TABLE 2 Comparison of AROW (Given in the First Column) and the AROW-MapReduce with Various Configurations in Terms of the AUC on the Task of Ad Latency Prediction

	AROW-MR Configuration				
Number of mappers	1	100	500	1000	10,000
Number of reducers	0	1	1	1	1
Average mapper time	408 h	30.5 h	34 min	17.5 min	2 min
Reducer time	n/a	1 min	4 min	7 min	1 h
AUC	0.8442	0.8552	0.8577	0.8662	0.8621

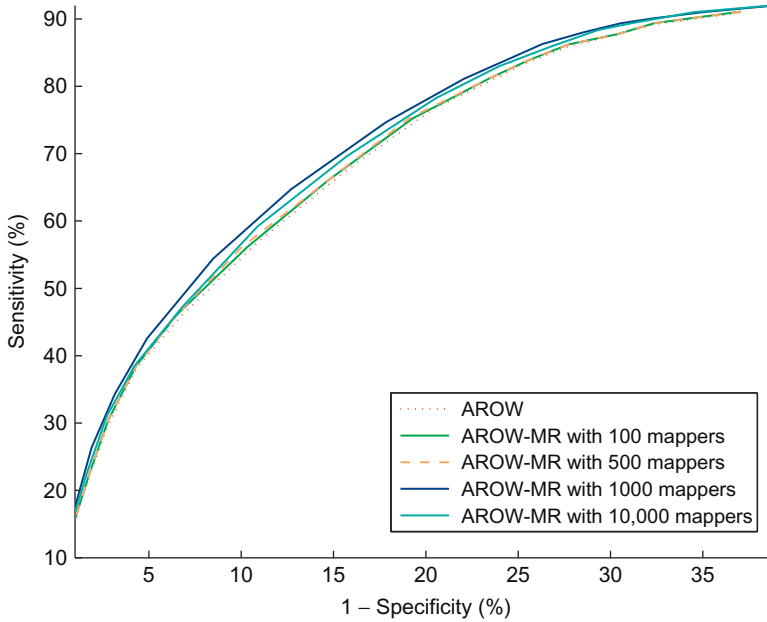


FIGURE 3 Comparison of ROC curves for AROW-MR and AROW algorithms.

time improved until we reached $M = 1000$ and dropped slightly for higher number of mappers. The detailed results are presented in Fig. 3, where we plotted ROC curves of the CW models trained using different number of mappers. We can see that the curve for nondistributed AROW results in the smallest AUC, while the level of parallelization achieved with 1000 mappers represents the best choice for the prediction task.

Next, in Table 3 we show the performance of LR model trained using VW package in both distributed (using AllReduce) and nondistributed mode (on a single machine). We can see that AROW-MR achieved higher accuracy than LR, which is a very popular approach used in large-scale classification tasks. While AROW-MR obtained AUC of 0.8662, the best AUC achieved by LR was 0.8508. Although the increase of 1.02% might not seem large, in the computational advertising domain it may result in significant increase of revenue. Interestingly, the results also indicate that more distributed training of LR actually hurts its generalization performance, which slightly dropped as the number of mappers was increased. Further, we can see that LR was trained in 6 min, while it took 24.5 min to train AROW-MR model. Although LR training is seemingly faster than training of AROW-MR, it is important to emphasize that VW package implements LR in C language while, for technical reasons, AROW-MR was implemented in Perl. Considering that Perl is not an optimal choice for mathematical computations, we expect AROW-MR training

TABLE 3 Performance of Distributed Logistic Regression

	Vowpal Wabbit configuration			
Number of mappers	1	100	500	1000
Number of reducers	0	0	0	0
Average mapper time	7 h	1 h	8 min	6 min
Reducer time	n/a	n/a	n/a	n/a
AUC	0.8506	0.8508	0.8501	0.8498

time to improve significantly if implemented in C or some other lower-level programming language.

It is also worth noting that we were not able to run the LR experiment for more than 1000 mappers. The LR implementation in VW package uses AllReduce framework, which requires all mappers to run concurrently without any node failures; otherwise, the entire training might fail. However, this is hard to guarantee in practice even for larger clusters, and it further exemplifies the advantage of the proposed algorithm over the competition. We can conclude that AROW-MR offers robust, efficient training of accurate classifiers, outperforming the existing state-of-the-art on large-scale, industry-size problems.

6 CONCLUSION

In this chapter we discussed popular Big Data platforms that allow for efficient and effective processing and analysis of terabytes of data within hours. We particularly focused on MapReduce, a popular framework which provides distributed, fault-tolerant environment for large-scale machine learning, comprising two distinct phases called *map* and *reduce*. Moreover, we showcased its practical application by introducing AROW-MR method, a distributed linear SVM solver capable of learning accurate classifiers in time sublinear in the size of a training data. Map phase of the method involves training a number of local AROW models on each mapper, followed by reduce phase which combines local classifiers to obtain a single, aggregated model more accurate than any of the individual ones. The experiments on synthetic data and real-world ad latency data set with nearly 1 billion examples suggest that AROW-MR allows for significant accuracy and training time improvements over the original AROW algorithm, indicating large potential and practical value of Big Data platforms.

ACKNOWLEDGMENTS

The authors would like to thank Vladan Radosavljevic and Hirakendu Das for helpful discussions. N.D. and S.V. would also like to acknowledge National Science Foundation support through grant IIS-0546155. This chapter is an extension of an earlier publication by the same authors (Djuric et al., 2013a).

REFERENCES

Agarwal, A., Chapelle, O., Dudík, M., Langford, J., 2011. A reliable effective terascale linear learning system. arXiv:1110.4198.

Bache, K., Lichman, M., 2013. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Bizer, C., Boncz, P., Brodie, M.L., Erling, O., 2012. The meaningful use of big data: four perspectives—four challenges. ACM SIGMOD Record 40 (4), 56–60.

Borthakur, D., Gray, J., Sarma, J.S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., et al., 2011. Apache Hadoop goes realtime at Facebook. In: International Conference on Management of Data. pp. 1071–1080.

Böse, J.H., Andrzejak, A., Högqvist, M., 2010. Beyond online aggregation: parallel and incremental data mining with online map-reduce. In: Proceedings of the 2010 Workshop on Massive Data Analytics on the Cloud. p. 3.

Chang, E., Zhu, K., Wang, H., Bai, H., Li, J., Qiu, Z., Cui, H., 2007. PSVM: parallelizing support vector machines on distributed computers. Adv. Neural Inform. Process. Syst. 20, 16.

Chu, C., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G., Ng, A.Y., Olukotun, K., 2007. Map-reduce for machine learning on multicore. Adv. Neural Inform. Process. Syst. 19, 281.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn. 20 (3), 273–297.

Crammer, K., Kulesza, A., Dredze, M., 2009. Adaptive regularization of weight vectors. Adv. Neural Inform. Process. Syst. 22, 414–422.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

Dean, J., Ghemawat, S., 2010. MapReduce: a flexible data processing tool. Commun. ACM 53 (1), 72–77.

Djuric, N., Grbovic, M., Vucetic, S., 2013a, Distributed confidence-weighted classification on MapReduce. In: IEEE International Conference on Big Data. pp. 458–466.

Djuric, N., Lan, L., Vucetic, S., Wang, Z., 2013b. BudgetedSVM: a toolbox for scalable SVM approximations. J. Mach. Learn. Res. 14, 3813–3817.

Dredze, M., Crammer, K., Pereira, F., 2008. Confidence-weighted linear classification. In: Proceedings of the International Conference on Machine Learning. pp. 264–271.

European Commission, 2014. Towards a thriving data-driven economy. The Commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J., 2008. LIBLINEAR: a library for large linear classification. J. Mach. Learn. Res. 9, 1871–1874.

Fawcett, T., 2004. ROC graphs: notes and practical considerations for researchers. Mach. Learn. 31, 1–38.

Gentile, C., 2002. A new approximate maximal margin classification algorithm. J. Mach. Learn. Res. 2, 213–242.

Gesmundo, A., Tomeh, N., 2012. HadoopPerceptron: a toolkit for distributed perceptron training and prediction with MapReduce. In: Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics. pp. 97–101.

Graf, H., Cosatto, E., Bottou, L., Dourdanovic, I., Vapnik, V., 2004. Parallel support vector machines: the cascade SVM. Adv. Neural Inform. Process. Syst. 17, 521–528.

Hughes, C., Hughes, T., 2004. Parallel and distributed programming using C++. Addison-Wesley, Boston, MA. pp. 31–32.

Joachims, T., 2006. Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 217–226.

Kivinen, J., Smola, A.J., Williamson, R.C., 2002. Online learning with kernels. IEEE Trans. Signal Process. 52 (8), 2165–2176.

Labrinidis, A., Jagadish, H., 2012. Challenges and opportunities with big data. Proc. VLDB Endowment 5 (12), 2032–2033.

Langford, J., Li, L., Zhang, T., 2009. Sparse online learning via truncated gradient. J. Mach. Learn. Res. 10, 777–801.

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., Kandola, J., 2002. The perceptron algorithm with uneven margins. In: International Conference on Machine Learning. pp. 379–386.

Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., Huang, T., 2011. Large-scale image classification: fast feature extraction and SVM training. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1689–1696.

Lohr, S., 2012, February 12. The age of big data. New York Times, 11.

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., Hellerstein, J.M., 2010. GraphLab: a new parallel framework for machine learning. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence. pp. 340–349.

Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M., 2012. Distributed GraphLab: a framework for machine learning and data mining in the cloud. Proc. VLDB Endowment 5 (8), 716–727.

Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G., 2010. Pregel: a system for large-scale graph processing. In: Proceedings of the ACM SIGMOD International Conference on Management of data. pp. 135–146.

Mervis, J., 2012. Agencies rally to tackle big data. Science 336 (6077), 22.

Nandan, M., Khargonekar, P.P., Talathi, S.S., 2014. Fast SVM training using approximate extreme points. J. Mach. Learn. Res. 15 (1), 59–98.

Platt, J., 1998. Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel Methods—Support Vector Learning. MIT Press, Cambridge, MA.

Rai, P., Daumé III, H., Venkatasubramanian, S., 2009. Streamed learning: one-pass SVMs. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence. pp. 1211–1216.

Severyn, A., Moschitti, A., 2010. Large-scale support vector learning with structural kernels. In: Machine Learning and Knowledge Discovery in Databases. Springer, Berlin-Heidelberg. pp. 229–244.

Shalev-Shwartz, S., Singer, Y., Srebro, N., 2007. Pegasos: primal estimated sub-gradient solver for SVM. In: Proceedings of the 24th International Conference on Machine Learning. pp. 807–814.

Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A., 2011. Pegasos: primal estimated sub-gradient solver for SVM. Math. Program. 127 (1), 3–30.

Shvachko, K., Kuang, H., Radia, S., Chansler, R., 2010. The Hadoop distributed file system. In: IEEE Symposium on Mass Storage Systems and Technologies (MSST). pp. 1–10.

Steinwart, I., 2003. Sparseness of support vector machines. J. Mach. Learn. Res. 4, 1071–1105.

Tsang, I.W., Kwok, J.T., Cheung, P.M., 2005. Core vector machines: fast SVM training on very large data sets. J. Mach. Learn. Res. 6 (1), 363.

Vishwanathan, S.V.N., Smola, A.J., Murty, M.N., 2003. SimpleSVM. In: International Conference on Machine Learning.

Wang, Z., Djuric, N., Crammer, K., Vucetic, S., 2011. Trading representability for scalability: adaptive multi-hyperplane machine for nonlinear classification. In: ACM SIGKDD Conference on Knowledge Discovery and Data Mining.

Xin, R.S., Gonzalez, J.E., Franklin, M.J., Stoica, I., 2013a. Graphx: a resilient distributed graph system on Spark. In: First International Workshop on Graph Data Management Experiences and Systems. p. 2.

Xin, R.S., Rosen, J., Zaharia, M., Franklin, M.J., Shenker, S., Stoica, I., 2013b. Shark: Sql and rich analytics at scale. In: Proceedings of the 2013 International Conference on Management of Data. pp. 13–24.

Yu, H.F., Hsieh, C.J., Chang, K.W., Lin, C.J., 2012. Large linear classification when data cannot fit in memory. ACM Trans. Knowl. Discov. Data 5 (4), 23.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I., 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. pp. 15–28.

Zhu, Z.A., Chen, W., Wang, G., Zhu, C., Chen, Z., 2009. P-packSVM: parallel primal gradient descent kernel SVM. In: IEEE International Conference on Data Mining. pp. 677–686.

Chapter 8

Big Data Applications in Health Sciences and Epidemiology

Saumyadipta Pyne[*,†,1], **Anile Kumar S. Vullikanti**[‡], **Madhav V. Marathe**[§,‖]

[*] *Bioinformatics, CR Rao Advanced Institute of Mathematics, Statistics and Computer Science, University of Hyderabad Campus, Hyderabad, India*

[†] *Public Health Foundation of India, New Delhi, India*

[‡] *Computer Science and Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, Virginia, USA*

[§] *Department of Computer Science, Virginia Tech, Blacksburg, Virginia, USA*

[‖] *Network Dynamics and Simulation Science Laboratory, Virginia Bioinformatics Institute, Virginia Tech, Blacksburg, Virginia, USA*

[1] *Corresponding author: e-mail: spyne@broadinstitute.org*

ABSTRACT

There is growing concern about our preparedness for controlling the spread of pandemics such as H1N1 Influenza. The dynamics of epidemic spread in large-scale populations are very complex. Further, human behavior, social contact networks, and pandemics are closely intertwined and evolve as the epidemic spread. Individuals' changing behaviors in response to public policies and their evolving perception of how an infectious disease outbreak is unfolding can dramatically alter normal social interactions. Effective planning and response strategies must take these complicated interactions into account. Mathematical models are key to understanding the spread of epidemics. In this chapter, we discuss a recent approach of diffusion in network models for studying the complex dynamics of epidemics in large-scale populations. Analyzing these models leads to very challenging computational problems. Further, using these models for forecasting epidemic spread and developing public health policies leads to issues that are characteristic of big data applications. The chapter describes the state of the art in computational and big data epidemiology.

Keywords: Mathematical epidemiology, Stochastic diffusion models, Public health policy planning, Epidemic surveillance, Epidemic forecasting, Agent-based simulations

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00008-3

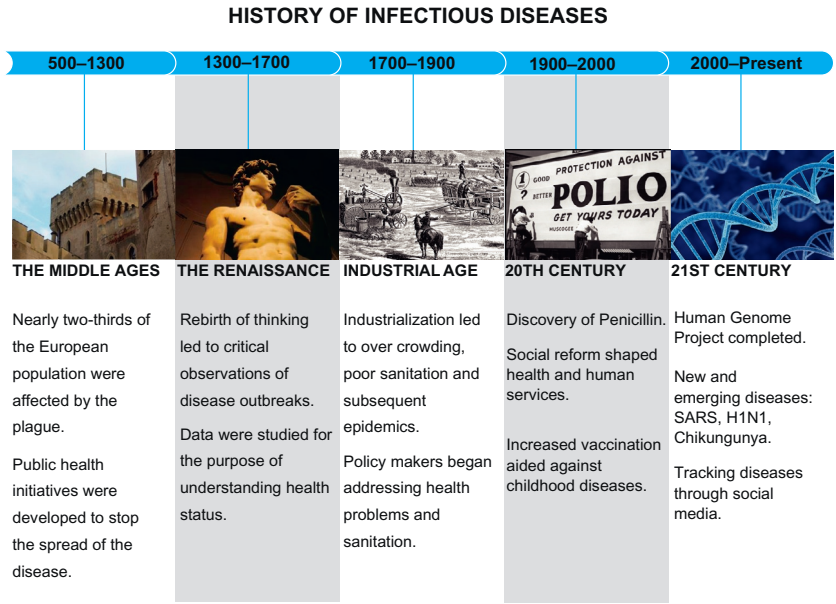


FIGURE 1 History of infectious diseases. Image courtesy: Maureen Lawrence-Kuether, Virginia Tech.

1 INTRODUCTION

Infectious diseases are the largest cause of human mortality worldwide—they account for more than 13 million deaths a year. Further, just six deadly infectious diseases—pneumonia, tuberculosis, diarrheal diseases, malaria, measles, and more recently HIV/AIDS—account for half of all premature deaths, killing mostly children and young adults. For example, malaria is said to be the primary cause of between 650,000 and 1.4 million deaths just in 2010. Key events related to the history of infectious diseases can be traced as far back as the middle ages (see Fig. 1).[1]

An epidemic is an occurrence in a community or region, of cases of an illness, specified health behavior or other health-related events in excess of what would be expected normally. The word epidemic is derived from two Greek works: *epi* (upon) and *demos* (people) meaning "upon people." A pandemic is an epidemic that spans a large portion of the world, such as the H1N1 outbreak in 2009. In contrast, an endemic disease is one wherein new infections are constantly occurring in the population. Epidemics are thought to have influenced significant historical events including the plagues in Roman times and middle ages, the fall of the Han empire in the third century in China, and the defeat of the Aztecs in the 1500s, due to smallpox outbreak. The 1918 pandemic was responsible for more deaths than those due to World War I. The last 50 years have

1. See a compilation of these events on wikipedia: http://en.wikipedia.org/wiki/List_of_epidemics.

seen epidemics caused HIV/AIDS, SARS, and influenza-like illnesses (see, e.g., Brauer et al., 2008).

Epidemiology is the study of the distribution and determinants of health-related states or events in specified populations, and the application of this study to the prevention and control of health problems (Last, 2001). By its very definition, the focus of epidemiology is on population-level issues as opposed to medical sciences, which is individual centric. Epidemiologists are primarily concerned with public health, which includes the design of studies, evaluation and interpretation of public health data, and the maintenance of data collection systems. They are interested in studying questions such as (1) What is the spatial extent of the outbreak? (2) How does the disease progress and how can we control it? (3) How did the disease originate and how does this compare to past outbreaks? Such questions have been asked repeatedly throughout human history. The 2014 Ebola outbreak in West Africa is an example of an epidemic that has caused thousands of deaths, along with significant social and economic impact on the affected countries.[2]

An important concern in epidemiology is to control an outbreak. With the advent of modern science, pharmaceutical measures have been widely used to control and prevent outbreaks. For example, vaccines have become a critical method of controlling, preventing, and possibly eradicating infectious diseases in host populations. Despite their success, nonpharmaceutical methods, such as quarantining and social distancing, continue to play a central role in controlling infectious disease outbreaks; they are especially important during an outbreak caused by an emerging pathogen. In such a case, pharmaceutical methods are usually not available.

Public health authorities have made significant strides in reducing the burden of infectious diseases. Nevertheless, infectious diseases continue to be an important source of concern. A number of global trends further amplify these concerns: (i) increased urbanization, (ii) increased global travel, (iii) denser urban regions, (iv) climate change, and (v) increased older and immunocompromised population. Many of the changes that we see around us are, to a large extent, anthropogenic and are happening at a scale wider and faster than ever before in human history. Further, new pathogens are emerging regularly, which raises the importance of societies' need to understand and be prepared to systematically address the challenge of emerging infectious diseases at different levels.

1.1 Mathematical and Big Data Computational Epidemiology

Mathematical models play an important role in epidemiology. Their importance is further highlighted by controlled physical experiments used to understand scientific phenomena being much harder and often impossible in epidemiology due to ethical and practical reasons. The first mathematical model in epidemiology

2. See http://www.nytimes.com/interactive/2014/07/31/world/africa/ebola-virus-outbreak-qa.html.

is credited to Daniel Bernoulli in 1760 (Brauer et al., 2008). Using mathematical techniques, Bernoulli established that variolation[3] could help increase the life expectancy in the French population. Another systematic and data-driven investigation of the spread of epidemics was by John Snow, a British physician, who analyzed a cholera outbreak in London in 1854 and attributed it to a source of contaminated water (Brauer et al., 2008).

The early 1900s saw seminal advances in mathematical epidemiology. In 1911, Sir Ronald Ross discovered the malaria vector and transmission due to a species of mosquitoes and later developed a spatial model for the spread of the disease. One of the most significant results from his model was that the spread of malaria could be controlled by reducing the population of malaria below a "threshold"—this is the first instance of the concept of an epidemic threshold. Kermack and McKendrik extended this to develop the first general epidemic model, popularly known as the SIR model, involving ordinary differential equations (ODEs) based on a mass–action model (Brauer et al., 2008; Newman, 2003); we discuss this later in Section 2.

Big Data Computational Epidemiology is an emerging interdisciplinary area that uses computational models and big data for understanding and controlling the spatiotemporal diffusion of disease through populations. Figure 2 illustrates the four key components of variety, velocity, volume, and veracity in epidemiology. The role of computation and big data in epidemiological science has been progressively increasing over the last 20 years. There are several reasons for this change. First, mathematical models have become increasingly complex; it is virtually impossible to solve these models without the use of computers. Second, there is an increasing acceptance of networked models—these models represent the underlying population as a complex interaction network and study the spread of diseases over these interaction networks. These network models are more natural than the traditional compartmental models, but this comes at an increased computational and data cost. For example, the network models we discuss later in Section 6 integrate over 34 public and commercial data sets, leading to over 100 GB of input data for constructing a population model for the USA, with over 300 million people and 100 million locations. In turn, analysis of such data sets requires very powerful computing resources, and novel high-performance computing approaches (Barrett et al., 2008; Bisset et al., 2009a). Often, many parameters of the models are not known, especially in the early days of an epidemic (e.g., the transmission probability), necessitating parametric studies. Additionally, most policy questions involve comparison and evaluation of different interventions to control epidemics, such as vaccinations and school closures. Individual behavior can have significant impact on the

3. Interestingly, it appears that the idea of *variolation* to control smallpox was known to Indians and Chinese as early as the eighth century AD. This method involved exposing people to material from smallpox scabs from infected individuals; see http://en.wikipedia.org/wiki/Inoculation for more details.

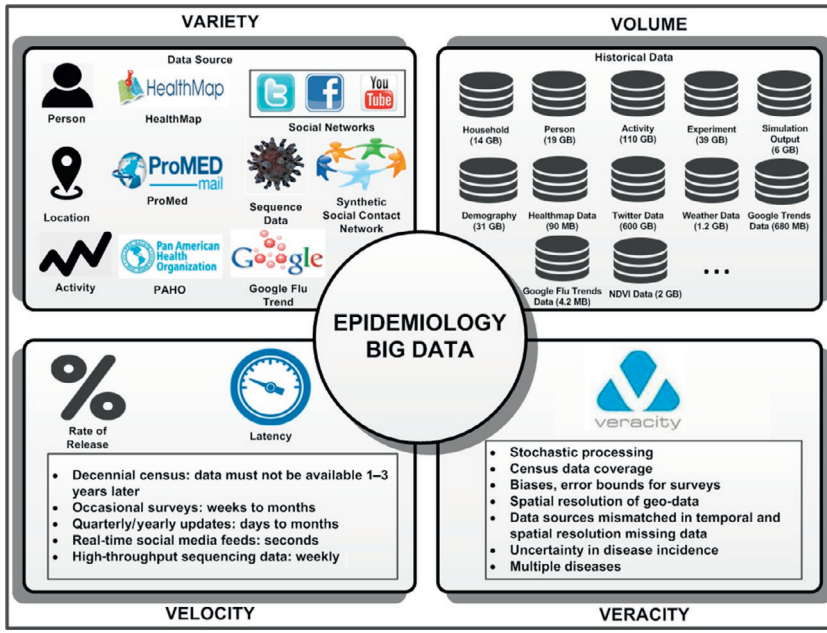


FIGURE 2 Big data issues in epidemiology: variety, volume, velocity, and veracity. Image courtesy: S.M. Shamimul Hasan, Virginia Tech.

disease dynamics (Funk et al., 2010; Lipsitch et al., 2009) and further increase the number of instances that need to be done. Third, epidemiologists are collecting increased amounts of data from the field. New methods are being developed for disease surveillance and detection. Although extremely valuable, the data need to be managed at all levels. Computational methods for data management, including methods to collect, store, clean, organize, search, fuse, and analyze data, are all important. Finally, with the advent of the world wide web, there is a growing demand for developing web-based tools that can be accessed by epidemiologists in a pervasive manner. An example of this is the HealthMap tool, which collects data about disease incidence all over the world, through news and social media data (Fig. 3a). Another tool is Google's FluTrends, which infers flu incidence from search queries, e.g., related to flu medication, news, etc. (Fig. 3b). Overall, big data play a large and important role in epidemiology (Salathé et al., 2012).

1.2 Organization and Outline

This chapter describes the basic concepts in computational epidemiology, with a focus on networked epidemiology. We discuss different kinds of mathematical models for epidemiology in Section 2. Some of the common computational

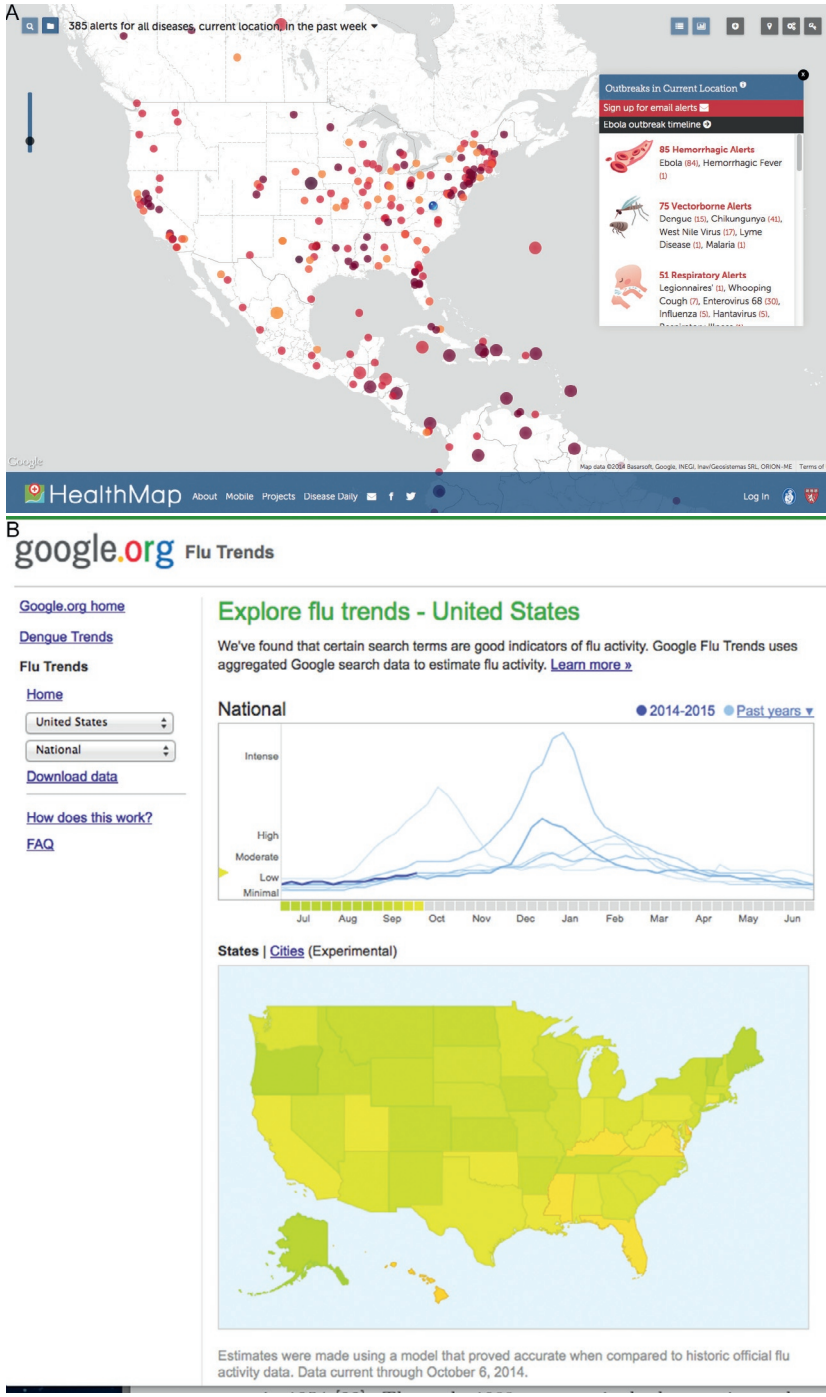


FIGURE 3 Screenshot from (a) HealthMap (http://www.healthmap.org/en/) and (b) Google Flu Trends (http://www.google.org/flutrends/), taken on October 7, 2014.

problems of analysis and inference are discussed in Sections 3 and 4, respectively. Section 5 discusses concepts in molecular epidemiology, disease surveillance, and phylodynamics and outlines important big data challenges in these fields. Section 6 discusses a computational framework for networked epidemiology. This includes a synthetic information environment for organizing and reasoning about diverse data sets, high-performance computing-based modeling environments, and pervasive web-based decision support tools.

2 MATHEMATICAL FRAMEWORK FOR EPIDEMIOLOGY

We start with a very commonly used model, popularly known as the SIR model, involving ODEs based on a mass–action assumption; we refer to Brauer et al. (2008) and Newman (2003) for more details. A population of size N is divided into three states: susceptible (S), infective (I), and removed or recovered (R) (Fig. 4). The following process describes the system dynamics: each infected person can infect any susceptible person (independently) with probability β and can recover with probability γ. Let $S(t)$, $I(t)$, and $R(t)$ denote the number of people who are susceptible, infected, and recovered states at time t, respectively. Let $s(t) = S(t)/N$, $i(t) = I(t)/N$, and $r(t) = R(t)/N$; then, $s(t) + i(t) + r(t) = 1$. Under the "complete mixing" assumption that each individual is in contact with everyone in the population, the dynamics of the SIR model can be described by a system of differential equations (Fig. 4b).

One of the classic results in the SIR model is that there is an epidemic which infects a large fraction of the population, if and only if $R_0 = \beta/\gamma > 1$; we refer the reader to Dimitrov and Meyers (2010) for a derivation of this result. R_0 is known as the "reproductive number" and is key in characterizing an epidemic. At the start of an epidemic, much of the public health effort is focused on estimating R_0 from observed infections (Lipsitch et al., 2003). Further, public health decision making for controlling an epidemic is often translated to reducing R_0—this can be done in many ways, such as by reducing β through better protective measures, such as washing and face masks, in the case of flu (Dimitrov and Meyers, 2010). While the notion of R_0 has given a lot of insights into the spread of epidemics, it has several limitations, especially in practice. For instance, during the SARS outbreak in 2003, R_0 was estimated to be in the range [2.2, 3.6]. However, despite its spread over many countries, it

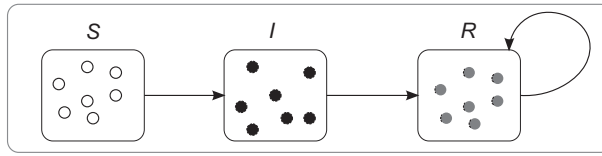


FIGURE 4 The SIR model. (a) The entire population is split into compartments corresponding to the S, I, and R states. Any two individuals in the population can come in contact. (b) Coupled differential equations for the SIR model.

did not cause a very large outbreak for multiple reasons. First, the R_0 estimates were based on infections in crowded hospital wards, where a complete mixing assumption is reasonable. But this does not hold in other settings. Second, as the epidemic spread, people altered their behavior and started mixing less, which is not taken into account in the static definition of R_0.

There has been a lot of work in developing more complex compartmental models in order to address some of these issues—compartments represent subpopulations with very similar characteristics and are well mixed within themselves so that the dynamics can be expressed in terms of differential equations over these groups. See Newman (2003) and Brauer et al. (2008) and the references therein for more details. Overall, compartmental models have been immensely successful over the last 100 years and have become a workhorse of mathematical epidemiology. They are relatively easy to extend and quick to build and can either be solved analytically or numerically quite efficiently, building on the well-developed theory of ODEs.

In contrast to compartmental models, *networked epidemiology* considers epidemic spread on a specific undirected contact network $G(V, E)$ on a population V—each edge $e = (u, v) \in E$ implies that individuals (also referred to as nodes) $u, v \in V$ come into contact.[4] Let $N(v)$ denote the set of neighbors of v. For instance, in the graph in Fig. 5, we have $V = \{a, b, c, d\}$ and $E = \{(a, b), (a, c), (b, d), (cd)\}$. Node a has b and c as neighbors, so $N(a) = \{b, c\}$. The SIR model on the graph G is a dynamical process in which each node is in one of S, I, or R states. Infection can potentially spread from u to v along edge $e = (u, v)$ with a probability of $\beta(e, t)$ at time instant t after u becomes infected, conditional on node v remaining uninfected until time t—this is a

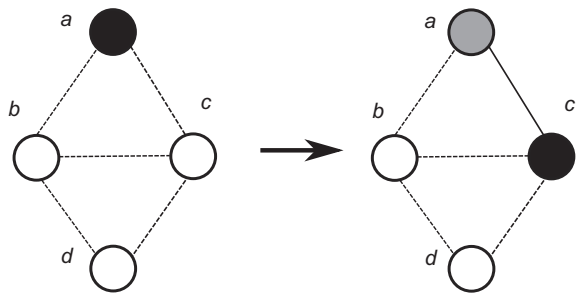


FIGURE 5 The SIR process on a graph. The contact graph $G = (V, E)$ is defined on a population $V = \{a, b, c, d\}$. The node colors white, black, and gray represent the Susceptible, Infected, and Recovered states, respectively. Initially, only node a is infected, and all other nodes are susceptible. A possible outcome at $t = 1$ is shown, in which node c becomes infected, while node a recovers. Node a tries to independently infect both its neighbors b and c, but only node c gets infected—this is indicated by the solid edge (a, c). The probability of getting this outcome is $(1 - \beta((a, b), 1))\beta((a, c), 1)$.

4. Note that though edge e is represented as a tuple (u, v), it actually denotes the set $\{u, v\}$, as is common in graph theory.

discrete version of the rate of infection for the ODE model discussed earlier. We let $I(t)$ denote the set of nodes that become infected at time t. The (random) subset of edges on which the infections spread represents a disease outcome and is referred to as a *dendrogram*. This dynamical system starts with a configuration in which there are one or more nodes in state I and reaches a fixed point in which all nodes are in states S or R. Figure 5 shows an example of the SIR model on a network.

The time series $(|I(t)|, t = 0, 1, \ldots)$ is referred to as an *epidemic curve* corresponding to a stochastic outcome; this is a very commonly used quantity related to an epidemic. The *total number of infections* for an outcome is given by $\sum_t |I(t)|$. The *peak value* is $\max_t |I(t)|$, while the *time to peak* is the largest time t that maximizes $|I(t)|$. These quantities are illustrated in Fig. 6. Also observe that the epicurve is not very smooth; this is in contrast with the epicurve in an ODE model. A popular variant of the SIR model is the SIS model, in which an infected node switches to state S after the infectious duration. Some of the notation used in the paper is summarized in Table 1.

Such a network model captures the interplay between the three components of computational epidemiology: (i) individual behaviors of agents; (ii) unstructured, heterogeneous multiscale networks; and (iii) the dynamical processes on these networks. It is based on the hypothesis that a better understanding of the characteristics of the underlying network and individual behavioral adaptation can give better insights into contagion dynamics and response strategies. Although computationally expensive and data intensive, network-based epidemiology alters the types of questions that can be posed, providing qualitatively different insights into disease dynamics and public health policies. It also allows policy makers to formulate and investigate potentially novel- and context-specific interventions. Such epidemic models are instances of a more general dynamical systems framework, called *graph dynamical systems* (GDSs) (Mortveit and Reidys, 2007). Therefore, the theory of GDS can be used as a

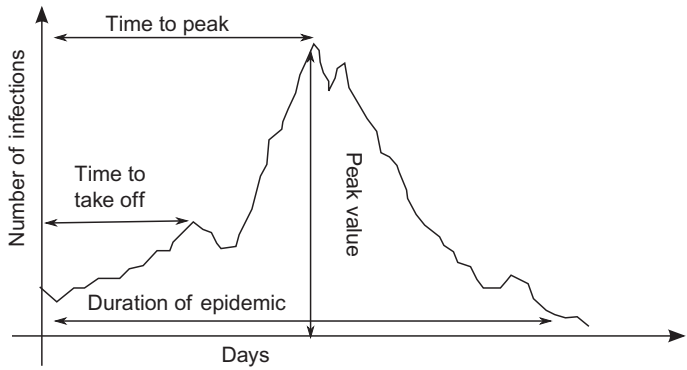


FIGURE 6 Different quantities associated with the dynamics of epidemics. See discussion in Section 2 for definitions of these quantities.

TABLE 1 Summary of Some of the Notations Used in the Chapter (See Marathe and Vullikanti, 2013 for More Discussion)

$G(V, E)$	Social contact network on a population V, with each edge $e = (u, v) \in E$ representing a contact
$N(v)$	Set of neighbors of node $v \in V$ in graph G
$\beta(e, t)$ for $e = (u, v)$	Transmission probability: the probability that the infection spreads from u to v per time unit of contact, if u is infected and v is not
$I(t)$	The set of nodes infected at time t
$(\|I(t)\|, t = 0, 1, \ldots)$	Epidemic curve
$\max_t \|I(t)\|$	Peak value of an epidemic
$\text{argmax}_t \|I(t)\|$	Time to peak

foundation for computational epidemiology; this leads to four basic classes of problems, which are motivated from the analysis of dynamical systems:

1. *Analysis problems.* This includes characterizing the outbreak size, duration of the epidemic, and other properties of epidemics. There has been a lot of work on analytical results in terms of network properties, e.g., Marathe and Vullikanti (2013), Easley and Kleinberg (2010), Newman (2003), Pastor-Satorras and Vespignani (2001), Alon et al. (2004), Chung and Lu (2002), Ganesh et al. (2005), and Wang et al. (2003), and on computational tools for such analysis, e.g., Perumalla and Seal (2011), Carley et al. (2006), Parker and Epstein (2012), Eubank (2002), Ferguson et al. (2006), Longini et al. (2005), Germann et al. (2006), and Chao et al. (2010). We discuss these problems in more detail in Section 3.
2. *Inference problems.* As discussed earlier, the networked SIR model is determined by the network, initial conditions, and epidemic model. In general, we may have partial information about some of these components; e.g., edges of the graph might not be completely known, or parameters of disease spread are unknown. Some of the results on such inference problems include Nishiura (2011a), Neil et al. (2005), Nsoesie et al. (2011), Shah and Zaman (2010), Netrapalli and Sanghavi (2012), and Gomez-Rodriguez et al. (2010). These are discussed in more detail in Section 4.
3. *Optimization problems.* This includes problems of controlling the spread of epidemics, e.g., by vaccination or quarantining, correspond to making changes in the node functions or removing edges so that the system converges to configurations with few infections, e.g., Borgs et al. (2010), Aspnes et al. (2006), and Eubank et al. (2004a). These are very challenging stochastic optimization problems, and beyond the scope of this chapter. We refer the reader to Marathe and Vullikanti (2013) for some pointers to this area.

4. *Forecasting and situational assessment problems.* These include problems of determining quantities, such as the number of infections over time, or the peak, and identifying the people who might be infected, given partial information of the outbreak till some time. This is a very active area of research (see, e.g., Chakraborty et al., 2014a; Ginsberg et al., 2009). See Section 5.2 for more details.

3 DYNAMICS AND ANALYSIS PROBLEMS

The structure of the underlying contact graph G has a significant impact on the disease dynamics (Newman, 2003). A fundamental question is to characterize the conditions under which there is a "large" outbreak, which is defined as one which infects $\Theta(n)$ individuals, where $n = |V|$. We mention a few of the main results of this kind for the SIR and SIS models. These are somewhat involved to derive here, and we give references for more details. A common approach is to try to characterize the dynamics in terms of the degree distribution of the graph. The simplest case is when G is a complete graph, with a uniform probability β of the disease spread from u to v, and $\tau(u) = 1$. The classical result by Erdős-Renyi (Marathe and Vullikanti, 2013; Newman, 2003) implies that there is a large outbreak with $\Theta(n)$ infections, if and only if $\beta > 1/n$. The main technique is a branching process analysis. We note that this result for the complete graph is a discrete analogue of the characterization in terms of R_0, with $n\beta$, the expected number of infections caused by a node, being the equivalent of R_0. The existence of such a threshold for disease spread has been shown to exist in other well-structured graph classes, such as lattices and random regular graphs (Newman, 2003). It is much more challenging to prove such statements in heterogeneous graphs with less symmetry. Pastor-Satorras and Vespignani (2001) use techniques from statistical mechanics to show that in scale-free network models, under mean-field assumptions the threshold for epidemics propagation is 0, i.e., no matter how small the probability of infection is, there would be a large outbreak. Two settings for which rigorous results are known, without using any mean-field assumptions, are (i) the Chung-Lu model (Chung and Lu, 2002), which is a random graph model in which the probability of an edge (i, j) is proportional to $w_i \cdot w_j$, for a given weight sequence $\mathbf{w}$; and (ii) classes of expander graphs (Alon et al., 2004).

4 INFERENCE PROBLEMS

In general, not all elements of the network model are known. For instance, the network might be partially known. Similarly, the transmission probability and the initial configuration, such as the source of the infection might not be known. Partial information in the form of observed infections might be available through public health surveillance. At the onset of an outbreak (e.g., in every flu season), recurring problems for public policy planners include determining where the

epidemic might have started, characteristics of the specific disease strain, and if there will be a large epidemic or pandemic. These are significant data and computational challenges, where statistical and machine learning approaches have an important role.

General abstractions of this problem are to determine the transmission probability or other disease characteristics, or the probability that the source of the epidemic is a node v, conditioned on observed surveillance data and some assumptions about the prior distribution, or to find a maximum likelihood estimator for them. A related class of problems is to infer the underlying contact network properties, based on such observations. These are very challenging problems because (i) surveillance data on who is infected are limited and noisy—only a fraction of the infected people are detected; (ii) the underlying contact graph is only partially known and is dynamic. These problems are the "inverse" of the "forward" problems of simulating the spread of an epidemic in a social network.

There has been a lot of work related to these problems, especially in the statistics and data mining literature (Neil et al., 2005; Nishiura, 2011a; Nsoesie et al., 2011). An active area of research is based on Bayesian approaches, e.g., using spatial scan statistics for detecting anomalous outbreaks (Neil et al., 2005). These techniques rely on detecting spatial clusters from syndromic surveillance data, without using any specific properties about the epidemic process. In contrast, a model-based approach relies on the characteristics of the SEIR process, and Nsoesie et al. use a combination of simulations and supervised classification techniques to predict epidemic curves and infer underlying disease parameters for an ongoing outbreak (Nsoesie et al., 2011).

One of the problems that have been studied is inferring the source of an infection, given the network and observed infections. This is generally formalized as a likelihood maximization problem. Shah and Zaman (2010) develop an efficient maximum likelihood approach for determining the source of an outbreak in trees for an SI model of disease and then extend it to general graphs by assuming spreading in a Breadth First Search (BFS) order. They also develop a heuristic, called rumor centrality, which allows faster estimation of the source probability. The related problem of inferring the contact network based on the disease spread information has been studied by Netrapalli and Sanghavi (2012) and Gomez-Rodriguez et al. (2010), who present maximum likelihood estimators for a minimal network on which a given set of observed infections can occur.

5 DISEASE SURVEILLANCE, MOLECULAR EPIDEMIOLOGY, AND PATHOGEN PHYLODYNAMICS

The emergence of new pathogens has led to new, critical challenges that are faced by the hitherto restricted interfaces of humans, animals, and the environment (Patz et al., 2004). Many of such interfaces around the world have been identified as hot spots of emerging zoonotic infectious diseases, in which an array of

ever-increasing human activities such as farming, mining, logging, and hunting are being conducted against the backdrop of significant wildlife biodiversity, of course, with concomitant microbial diversity (Morse et al., 2012). Perhaps the regions in which environmental changes have been occurring most rapidly, and where people and wildlife live in close proximity, are located mostly in the developing world, and which, ironically, have limited resources and capacity to monitor and respond to zoonotic outbreaks.

Morse et al. (2012) described a three-stage model for typical dynamics of transmission of zoonotic diseases from a localized hot spot to a global pandemic. In preemergence stage 1, a pathogen is localized in its natural animal reservoir but could be driven by certain major ecological or environmental disturbances, which increases the likelihood of its coming to contact with, and spilling over to, new species including livestock or humans that are in increasingly close contact. For instance, the recent outbreak of Ebola has been brought back to focus the issue of "growing human activity and deforestation in previously untouched forests."[5] The 2002 SARS outbreak in China was traced to hunting and trading of bats that are natural reservoirs of SARS-like corona viruses (Li et al., 2005).

In stage 2, while the disease spills over to humans, it remains localized within a geographical region with outcomes varying between a few cases to a moderate or even large outbreak, but possibly with limited interpersonal transmission, as, for instance, in the case of the recent Ebola outbreak. A large-scale pandemic occurs in stage 3 and is quite rare. It is usually attributed to global travel and trade and characterized by sustained interpersonal transmission and international spread of reservoir hosts or vectors. To address the problem of zoonosis, clearly the most desirable objective is to tackle and limit the transmission of a zoonotic pathogen, if possible, in stage 1. Unfortunately, the global trend in emerging infectious diseases has seen increasing incidence since 1940, despite controlling for effects of higher reporting (Jones et al., 2008). It was found that 60% of known human infectious diseases, and more than 75% of emerging infections, have zoonotic origin (King et al., 2006; Taylor et al., 2001). In fact, 71.8% of zoonotic pathogens originated in wildlife species. Approximately 80% of the identified reservoirs are mammalian, and almost half of the observed 1000 pathogens noted in livestock and pets are zoonotic. Indeed, almost 80% of the viruses and 50% of the bacteria that infect humans are of zoonotic origin. A conservative estimate puts the number of viruses present in vertebrate species at 1 million, which suggests that more than 99.9% of the viruses are currently unknown (Lipkin, 2009). While such staggering numbers are a matter of great concern, it also presents a perspective to the researchers so that they may be prepared to approach the problem in a comprehensive and systematic manner. Data on such pathogens and their evolution help in disease surveillance and

5. http://www.washingtonpost.com/news/morning-mix/wp/2014/07/08/how-deforestation-and-human-activity-could-be-to-blame-for-the-ebola-pandemic/.

forecasting and lead to significant big data challenges, as illustrated in Fig. 2. We briefly summarize these aspects below.

5.1 Disease Surveillance

Recently, the concept of "One Health" (Coker et al., 2011) has received much attention led by the combined initiatives of FAO and the UN, WHO, and the World Organization for Animal Health. Importantly, the need for a concerted, interdisciplinary approach has been felt in public health research, bringing together teams of epidemiologists, clinicians, veterinarians, microbiologists, wildlife biologists, ecologists, data analysts, and public health officials. Apparently, a global strategy seems to be emerging for monitoring and biosurveillance, partly driven by concerns of bioterrorism. However, the coverage or efficiency of public health surveillance systems worldwide is still far from uniform.

Notable efforts in this direction are made with the help of social media (Lipkin, 2013). ProMED (Program for Monitoring Emerging Infectious Diseases) was created in 1994. In a bottom-up manner, a network of reader at the grassroots submits entries that are then curated by an expert panel that sends ProMED-mail with commentary in five languages to a listserv exceeding 60,000 subscribers in 185 different countries. A private subscription-based service GPHIN (Global Public Health Intelligence Network) receives information on global outbreaks by scanning worldwide news in nine languages. In 2001, GPHIN joined the WHO Global Outbreak Alert Response Network (GOARN), which included additional verification services to its reporting. HealthMap.org, launched in 2006, is a user-friendly map-based online display system for real-time crowd-sourced submission of reports of worldwide georeferenced observations from news media, either through its web site or through a number of smart-phone-based apps (e.g., Outbreaks Near Me). Figure 3 shows screenshots of HealthMap and Google FluTrends, which are two tools for disease surveillance.

Typically, automated surveillance mechanisms would generate data with the big data characteristics of high volume, velocity, and variety. Researchers in big data and Computational Statistics must design robust models and algorithms to address the analytical needs of working with such data. A variety of methodological advances have been made to analyze data from social networks, sensor networks, data streams, GIS data, and associated platforms now routinely used for monitoring and surveillance (Cooper et al., 2006). Computational frameworks for biosecurity involve clustering, classification, prediction, outlier analysis, univariate and multivariate stream analysis, data fusion, social network analysis, text and web data mining, spatial and spatiotemporal modeling, and visual analytics, among others.

Many public health researchers and policy experts are currently involved in designing programs and strategies to be prepared for emerging (and reemerging) diseases. Systematic identification of new hot spots; modeling and mapping

of ecological niches; characterization of human–animal interfaces by exposure types and frequency; creation of host–pathogen databases; identification of key taxonomic groups; and phylodynamic analysis of host, epidemiological, and molecular data all lead to a pool of information that could be useful for prediction and prevention of zoonotic outbreaks. As an example of this approach, the PREDICT project of the Emerging Pandemic Threats (EPT) program launched by US Agency for International Development (USAID) in 2009 aims to facilitate predictive modeling for the identification of the most likely regions, hosts, and human–animal interfaces for forthcoming emergence of zoonoses (Morse et al., 2012).

As described by Morse et al. (2012), PREDICT aims to collect timely and reliable data based on Internet surveillance of reports of unusual health events occurring in countries with hot spots. Further, it conducts analysis to test if the corresponding pathogen is likely to emerge and spread in the social systems that exist in those hot spots. PREDICT combines risk modeling with targeted wildlife field sampling for selected locations, interfaces, and host taxa. In this effort, it is aided by interdisciplinary experts, computerized data collection and analysis, and active partnership with local and national governments. The program currently collaborates with 20 African, Asianm and South American countries, and in just a few years, it has detected hundreds of novel viruses in the hundreds of thousands of samples collected from tens of thousands of animals from these locations (Morse et al., 2012).

While programs such as PREDICT can demonstrate the benefits of local capacity-building efforts (Schwind et al., 2014) as part of international efforts to counter zoonotic threats, it also underscores the critical need for statistical and computational capability to work with massive data sets of high volume, velocity, and variety. To create sophisticated models for forecasting, the researchers must consider high-dimensional data on a large number of parameters: socioeconomic drivers (e.g., population growth and density, mixing patterns, migration, trade, agricultural practices, sanitation, age, diet, vaccination, drug and antibiotic use, cultural norms, occupational exposures, nutritional and immunological status), wildlife diversity, contact frequency, relatedness of host species, relatedness of microbial species present in host, evolvability of pathogens, host–pathogen coevolution, and several general factors such as capacities for reporting and response on the ground, geographical, and ecological conditions, and so on. An array of tools designed for predictive analytics, ranging from neural networks and tree models and regression models based on the observed parameters, to complex representations such as Bayesian network models that can capture their underlying dependence structure, are gaining in popularity (Buczak et al., 2013; Cooper et al., 2006).

Serological surveillance is routinely conducted worldwide in order to determine the prevalence of a disease in a population. From a statistical perspective, cross-sectional or longitudinal serological surveys provide useful data in terms of levels of antibodies in serological samples owing to past infections. Thus,

parametric or nonparametric models are generally used to separate the susceptible and the infected subpopulations based on seroprevalence data, within specific age groups or otherwise, allowing inference on such parameters as the prevalence and the rate of infection, as described in Hens et al. (2012).

5.2 Forecasting

An important emerging topic in big data computational epidemiology is epidemic forecasting. The basic idea here is to combine big data obtained from nontraditional sources, including social media, wikipedia, electronic health records, crowd-sourcing as well as surveillance systems with causal and statistical models to *nowcast* and *forecast* prevalence of diseases in the host population. Just like weather systems and markets, epidemics are complex systems; as a result forecasting the future incidence rates is challenging. See Nsoesie et al. (2013), Nishiura (2011a), Ohkusa et al. (2011), Hall et al. (2007), and Tizzoni et al. (2012) for a recent survey and early work on this topic. Shaman et al. have used Bayesian ensemble methods to develop surprisingly high-quality forecasts for flu prevalence in US regions (Shaman and Karspeck, 2012; Shaman et al., 2010a,b). An important contribution of their work was to elucidate the role of weather (humidity) on the incidence rates. In a recent paper (Chakraborty et al., 2014b), as a part of the IARPA-funded EMBERS project, we analyzed the generation of robust quantitative predictions about temporal trends of flu activity, using several surrogate data sources for 15 Latin American countries. We also recently participated in the CDC flu challenge and used our models to develop real-time flu forecasts for the nine US regions. An important consideration in both cases was to produce predictions in real time before the event and compare these forecasts retrospectively using curated data sets from CDC and PAHO. See https://www.federalregister.gov/articles/2013/11/25/2013-28198/announcement-of-requirements-and-registration-for-the-predict-the-influenza-season-challenge. Recently, other agencies have also put out similar competitions, e.g., a similar competition for chikungunya was initiated by DARPA (see https://www.innocentive.com/ar/challenge/9933617). The challenges have helped focus renewed interest on developing innovative techniques for infectious disease forecasting.

5.3 Molecular Epidemiology

In the "postgenomic" era, a significant gain in the predictive value of such modeling comes from incorporating molecular data, which are used along with the traditionally available clinical, pathological, and epidemiological data. Classically, Koch's postulates, formulated by Koch and Loeffler in 1884, have served as the gold standard criteria to establish a causative relationship between a microbe and a disease. They require that the agent must be present in all cases of the disease, it must be specific, and it must be isolated and cultivated as

culture, which, if inoculated in a healthy host is, must be sufficient to reproduce the original disease. With the advent of molecular methods, these criteria were modified to allow sequence-based identification of microbial pathogens by Fredricks and Relman. Recently, "metagenomic Koch's postulates" were proposed such that metagenomic markers (individual sequence reads, assembled contigs, genes, or genomes) could be used to detect viruses, most of which are not easy to culture (Mokili et al., 2012). An integrative approach to establish causation was described by Lipkin (2013) in terms of possible, probable, and confirmed causal relationships.

Systematic identification of the causative agent can play a key role in epidemiology; it can help not only with estimation of the basic reproductive number (R_0) but also with inference of the probable routes and dynamics of transmission, and even indicate possible intervention strategies. Traditional approach to pathogen discovery has involved an established array of techniques ranging from *in vitro* and *in vivo* culturing to platforms such as immunohistochemistry, electron microscopy, and serology (Morse et al., 2012). In the 1990s, the advent of high-throughput "omic platforms revolutionized rapid, cost-effective molecular data generation, enabled by important methodological advances in statistics and bioinformatics, paved the way for unbiased, data-driven discoveries". Multiplex PCR arrays, mass spectrometry, microarrays, and high-throughput sequencing are now routinely used for microbial analysis. As costs (money, time need for analysis) of such analyses continue to fall, while their efficiency (multiplexing, throughput) continues to rise, routine screening of samples for prospective pathogen discovery could soon become a viable strategy to preempt the emergence of zoonotic disease in a systematic manner.

High density of primer sequences and the increasing robustness of the technology allow nucleotide-based microarrays (or "chips") to probe for a large number of potential viruses very cheaply and rapidly. In fact, with a flexibly designed set of probes, and the compactness of 8–10 million spots, the possibility of having a single chip that can test for all the known viruses, at least for narrowing down to the level of genus or family, for the price of just $20 or less, is not far-fetched (Vaidyanathan, 2011). The ViroChip was used to detect the presence of coronavirus family in the respiratory samples of SARS in less than 24 h and without the need for culturing the virus (Chen et al., 2011). Besides the pan-viral ViroChip, there are also the pan-microbial microarrays such as GreeneChip (Palacios et al., 2007) and LLMDA (Gardner et al., 2010). The trend is to use components that can make such platforms in the future even more specific in their detection capacity and smaller in size so that they could be more effective and easily deployed in the field. Parallel and distributed computing, on the other hand, could make search for agents and their biomarkers more robust and efficient during outbreaks, e.g., Fox et al. (2013).

The real game-changer in the field of novel microbe hunting has been, however, due to the advances in high-throughput sequencing (HTS) over the past decade (see reviews by Loman et al., 2012; Metzker, 2009). Currently,

there are multiple HTS platforms available, each with their own strengths and weaknesses. The main reason that makes HTS so suitable for pathogen discovery is that unlike previous methods like the consensus PCR or microarrays which are dependent on the primer sequences, it allows unbiased identification of both known and unknown agents, including highly divergent and novel pathogens. A list of viral pathogens that were detected by HTS is given in Chiu (2013).

The first HTS platform to be thus used was the 454 Life Sciences pyrosequencer, which is still in use today as the Roche GS FLX Titanium and GS Junior systems. However, the more recent and currently the most widely used HTS systems are from Illumina, which include the massively parallel HiSeq 2000, and the faster but lower throughput MiSeq (Firth and Lipkin, 2013). The selection of an HTS system for pathogen discovery is determined by various factors. First, as the pathogen genomes, especially viral genomes, are relatively small compared to that of hosts, the amount of template required for input must not be prohibitive. In particular, in tissue or fecal samples, the ratio of viral to background material is often quite low. This leads to steps of careful steps toward enriching the agent (or host depletion) during sample preparation for HTS (Firth and Lipkin, 2013). The two key HTS parameters for pathogen discovery are the read length and the depth of coverage. While reads must be long enough (at least 100–300 bases) to allow unambiguous identification of the agent as distinctively as possible from the host or background sequences, the read depth (i.e., the number of sequence reads per run), on the other hand, must be reasonably high to allow easy assembly of the microbial genome and thus enable sensitive detection. Illumina platforms generate almost 10–1000 times more data per run with much smaller read lengths (HiSeq 600 Gb, 150 bases; MiSeq 3 Gb, 250 bases) as compared to GS FLX (0.7 Gb, up to 700 bases) but they also take significantly longer runtime (HiSeq 11 days, MiSeq 27 h, GS FLX 8 h) (Chiu, 2013; Firth and Lipkin, 2013). Clearly, running time and costs are critical factors, especially during an outbreak. Fortunately, the per-base cost of sequencing has fallen steadily, from $5000 per megabase in 2001 using classical dideoxy methods to $0.50 per megabase in 2012 using the Illumina platform (Lipkin, 2013). A virus that might have taken a week to sequence a decade ago (say, SARS-coronovirus in 2003) now requires at most a day to finish. Further, access to HTS labs is also gradually increasing worldwide.

Increasingly, it is not so much the constraints of data acquisition that are of concern to HTS labs across the globe as the challenge of efficient analysis of the large amounts of generated data. Traditionally, the computational pipelines for pathogen detection have involved the key steps of (a) computational subtraction of the host or background sequences using reference databases, and (b) BLAST homology search of the target agent against all known microbes. If BLASTn search by nucleotide homology does not work well for identification of distantly related or novel pathogens, then protein-level similarity with translated BLASTx query is used to explore further, especially for long reads exceeding 150

bases (Firth and Lipkin, 2013). More divergent relationships are explored with frameworks such as PHI-BLAST or HMMER (Finn et al., 2011). In the recent years, several analytical pipelines, such as PathSeq, CaPSID, and READSCAN, were developed specifically for HTS-based pathogen discovery, e.g., Kostic et al. (2011), Borozan et al. (2012), and Naeem et al., (2013). New pipelines, such as SURPI (Naccache et al., 2014), are getting more scalable and efficient by taking advantage of cloud-based technologies (in "fast mode," SURPI detects pathogens analyzing data with 7500 million reads in 11 min to 5 h).

As only less than 1% of the world's microbial diversity has been cultivated in the lab, an entire vista has been opened up by the recent advances in the field of metagenomics, which is a culture-independent approach for "functional and sequence-based analysis of collective microbial genomes contained in environmental samples" (Riesenfeld, 2004). Viral metagenomics is getting increasingly popular for a wide range of investigations which include not only environmental research areas, such as marine ecology or agricultural biotechnology, but also diagnostics of human diseases (Barzon et al., 2011). While it was originally based on cloning methods for analyzing double-stranded DNA genomes, the use of HTS now allows metagenomic analysis of all kinds of genomes, including that of single-stranded DNA and RNA (Mokili et al., 2012). Typically, a pipeline for metagenomic analysis consists of a sample preparation step, which helps in removal of nonviral nucleic acids, followed by HTS, and finally, bioinformatics analysis whereby the genomic sequences of individual microbes present in the collection are distinguished.

While the presence of individual microbes is detected post hoc from a mixture thereof in metagenomic analysis, an alternative and interesting approach is to first single out the agents and then amplify the nucleic acids and sequence them individually. Single-cell analysis (SCA), followed by metagenomics using HTS, can significantly improve the sensitivity of microbial detection (Blainey, 2013). In SCA, individual cells are first isolated from the sample, captured separately (say, in microfluidic chips), and then sent for metagenomic or metatranscriptomic analysis. This obviates the need for post hoc computational separation of the microbial genomes. While multiple projects in single-cell genomics have been conducted with bacterial species, recently single viral genomics (SVG) was conducted by Allen et al. (2011). Using platforms such as flow cytometry, single virions were selectively isolated before sequencing was carried out. Interestingly, by incorporating an intermediate reverse-transcription step prior to HTS, the SVG approach could be particularly useful in the genomic analysis of viral heterogeneity, especially for RNA viruses, which have much higher mutation rates during replication (and absence of error-correction), and thus form viral quasispecies (Holmes, 2009). With their high mutation and replication rates, and under selective pressures from the use of drugs and vaccines, RNA viruses can quickly evolve to use new cell receptors or routes of transmission in unexpected host species (Firth and Lipkin, 2013).

5.4 Pathogen Phylodynamics

A systematic understanding gained through sustained and continuous investigation of viral genetic diversity should be a key aspect of advanced biosurveillance efforts in global preparedness for disease outbreaks. While the overall genomic collection of the discovered viral pathogens continues to grow, simultaneously, the pathogens keep on evolving and there are both new emergence as well as reemergence of infectious diseases. In particular, the evolvability of RNA viruses, given their high mutation rates and large population sizes, allows such evolution to take place in actual timescales of human observation. The field of phylodynamics, therefore, seeks to combine data from both phylogenetics and epidemiological dynamics (Grenfell et al., 2004). This combined approach can allow more effective prediction of the epidemiological impact of newly emerged or evolved viruses. Further, Holmes and Grenfell (2009) suggest simultaneous collection of data of different types such as (a) spatiotemporal dynamics of the disease , (b) viral genome sequences, (c) contact networks of susceptible host individuals, and (d) the immune history of the individuals in contact networks, which, taken together, "are key for understanding both the dynamics of epidemic spread and the evolutionary pressures that shape virus diversity." An example of such integrative analysis is given in Cottam et al. (2008).

Clearly, such integrative analysis has the hallmark of Big Data Analytics where data have large size, are of diverse types, and generated dynamically. By analyzing possible interactions between the viral and immune dynamics within the infected hosts, one can determine the trajectory of pathogenic adaptations in real time. Algorithms to model normal immunologic profiles in the population can help in both outbreak prediction and detection (Pyne et al., 2009, 2014; Ray and Pyne, 2012). It can provide critical insights into the spatiotemporal spread of specific infections in given populations, help in estimation of R_0, and inferring viral phenotypes (e.g., virulence, transmissibility) (Volz et al., 2013). Under different phylodynamic processes, the viral phylogenetic tree shapes assume different spatiotemporal patterns depending on the strength of the immune-driven selection (Grenfell et al., 2004). Finally, phylodynamics also allows viral diversity to be studied statistically using the coalescent theory, which links phylogenetic structures with ecological processes, often to reconstruct demographic histories of infected individuals, or to estimate the parameters of infectious disease dynamics (Pybus and Rambaut, 2009; Suchard and Rambaut, 2009). For this purpose, parallel algorithms and high-performance computing have been used to support computationally intensive Bayesian frameworks to run on large data sets, e.g., Suchard and Rambaut (2009) and Ayres (2012). New platforms like OutbreakTools provide special formats to represent epidemiologic and sequence data together to allow statistical analysis, simulation, and visualization (Jombart et al., 2014). Owing to the HTS platforms, presently more than 16,000 human and avian isolates have been completely sequenced in the Influenza Genome Sequencing Project. In the near future, programs for discovery of

pathogens and prediction of their transmission dynamics in host populations thus clearly seem to belong to the field of big data analytics.

6 HIGH-PERFORMANCE SYNTHETIC INFORMATION ENVIRONMENTS AND TOOLS

As discussed in Section 2, the key components of the the network-based SIR models of epidemic spread are the contact network and the disease spread model. The latter was discussed in Section 5. We now discuss how contact networks are modeled. These are very complex to analyze and simulate, and we discuss HPC tools for using such models.

6.1 Synthetic Networks for Epidemiology

Real data for large-scale social contact networks are not easily available, at least in the public domain, because of privacy and proprietary issues. As discussed earlier, a common approach to deal with lack of realistic data is to use simplified stochastic models, which match aggregate properties, e.g., degree distribution (Chung and Lu, 2002; Newman, 2003). However, as argued by Li et al. (2004), and many other researchers subsequently, there are significant limitations to such simplified models. We describe first-principles-based methods for generating synthetic regional and national scale social contact networks (Barrett et al., 2005; Eubank et al., 2004b). Unlike simple random graph techniques, these methods use real world data sources and combine them with behavioral and social theories to synthesize networks. This work builds on the TRANSIMS modeling environment (Barrett et al., 2000, 2001; Beckman et al., 1996) and has since been extended (Barrett et al., 2009). This approach integrates over a dozen public and commercial data sets and involves four broad steps discussed below; see Barrett et al. (2005, 2009); Eubank et al. (2004b) for more details, and a discussion on structural differences between synthetic networks and simple random networks.

The first step involves the creation of a synthetic urban population by integrating a variety of public and commercial data sets, while preserving their privacy and maintaining statistical indistinguishability. The synthetic population is a set of synthetic people and households, geographically located, each associated with demographic variables drawn from any of the demographics available in the input data set. Each synthetic individual is created in the context of a household with other synthetic individuals, and each household is located geographically. The second step involves the construction of a set of activity templates for households, based on several thousand responses to an activity or time-use survey. These activity templates include the sort of activities each household member performs and the time of day they are performed. Locations are selected for all activities, following capacity constraints, and models from transportation literature. Detailed route plans are assigned to individuals in the

third step, based on the locations where activities are performed and the road network that connects the locations. Finally, detailed movement patterns are constructed using a cellular automata-based microsimulation for individuals over the transportation infrastructure.

The resulting synthetic population has a spatial resolution of few meters and a temporal resolution of a minute, across a large urban region. This information can be captured in the form of a *dynamic social contact network*, represented by a vertex and edge labeled bipartite graph G_{PL}, where P is the set of people and L is the set of locations. If a person $p \in P$ visits a location $\ell \in L$, there is an edge $(p, \ell, label) \in E(G_{PL})$ between them, where *label* is a record of the type of activity of the visit and its start and end times. Note that it is impossible to build such a network for any region large enough to be epidemiologically significant solely by collecting field data, although such data can be incorporated into the synthetic population creation process. The use of generative models to build such networks is a unique feature of this work.

Recently, researchers have included other forms of data and information to extend the basic methodology described above. Examples include (i) using information from airline data to construct network-based representation of cities across the globe—in this each node corresponds to a city and edge corresponds to number of travelers that go between the two cities as measured by air transport data (Colizza et al., 2006); (ii) representation of smaller subnetworks (aka micronetworks), using either survey data or data collected using sensors (Mossong et al., 2008; Salathé et al., 2010); and (iii) use of LandScan data in conjunction with census and other sources of population information to construct resolved networks that are not as accurate but can be constructed easily for several cities as well as countries (Ferguson et al., 2005; Longini et al., 2005).

6.2 Individual and Collective Behaviors

A primary goal of an epidemiologist is to control the spread of infectious disease through the application of interventions, guided by public policy. Policy-based interventions induce a behavioral change in individuals. At the same time, individuals self-impose behavioral changes in response to their perception of the current state of disease incidence. Both of these modifying factors imply that the underlying social network is constantly changing. In fact, individual behaviors, public policies, disease dynamics, and the social contact networks that they generate interact and coevolve as the outbreak progresses. The recent SARS epidemic served as an excellent example of how individual behavior as well as public policies played an important role in changing the social network.

Health scientists have developed verbal or conceptual behavioral models (Bandura, 1986; Becker, 1974) to understand the role of behaviors in public health. These models have played an important role in understanding behaviors and its relationship with diseases and maintaining a healthy life. These include

the *Health Belief Model (HBM)* (Becker, 1974), models based on the Social Cognitive Theory (SCT) (Bandura, 1986), and the Social Ecological Model (SET). Verbal social and behavioral theories have proven useful in improving public health—but are often informal. These informal models need to be represented as formal computer procedures when using computer models to study epidemics. This turns out to be challenging. The role of information is critical in how such models are developed. We have investigated behavioral models based on (i) role of information (kind of information: space, time, and networked; type of information: local or global; and completeness of information), (ii) computational considerations (efficiency of encoding, computational resources required expressiveness of the language), and (iii) scale of behavior: individual, collective, and institutional, and it is quite demanding to identify the data necessary to instantiate *in silico* behavioral models. Recent advances in social media, computational turks, online games, online surveys, and digital traces are potentially exciting new ways to make progress in this direction. Of course availability of these data sets poses new kinds of questions, including (i) design to elicit truthful behavior, (ii) biases in data due to the demographics of participants, and (iii) translating behaviors in virtual world to the real world. See Funk et al. (2009), Salathé et al. (2012), and Barrett et al. (2010) for recent discussion on this emerging topic.

6.3 High-Performance Computing Tools

As discussed earlier in Section 3, phase-space properties of epidemic models have been analyzed mathematically for a small class of random and regular graph models. Computing disease dynamics over complex social contact networks is a computationally challenging problem, motivating the need for efficient simulations to calculate the spatiotemporal disease propagation. One of the first such simulations was EpiSims (Eubank, 2002; Eubank et al., 2004b; Mniszewski et al., 2008), which had a powerful modeling environment, but was not able to scale to large networks very easily. We have developed three different tools, Episimdemics (Barrett et al., 2008; Bisset et al., 2009b), EpiFast (Bisset et al., 2009a), and Indemics (Bisset et al., 2010a,b, 2011; Deodhar et al., 2012; Ma et al., 2011a,b)— these provide a trade-off between computation speed, model realism and sophistication, and ease of introducing new behavior and interventions. All three can be executed on traditional distributed memory clusters of varying sizes. EpiSimdemics is quite general and flexible, at the cost of decreased computational speed. It is an interaction-based, highly resolved modeling and simulation system for representing and reasoning about contagion diffusion across large (300 million nodes and 1.5 billion edges) networks. EpiFast (Bisset et al., 2009a) is a reaction-diffusion model that runs on an explicit social networks. It is based on a client–serve bulk synchronous computing paradigm and runs extremely fast—a single run on a network with 10 million nodes and 500 million edges take about 40 s on a 40-core machine. Indemics

is an extension to EpiFast that integrates database computing to enable the most general set of interventions that are the easiest to create, but at the loss of computational speed (Bisset et al., 2010a,b, 2011; Deodhar et al., 2012; Ma et al., 2011a,b). This is an active area, and some of the recent work includes Perumalla and Seal (2011), Carley et al. (2006), Parker and Epstein (2012), Eubank (2002), Ferguson et al. (2006), Longini et al. (2005), Germann et al. (2006), and Chao et al. (2010).

6.4 Decision Support Environments

The epidemiological modeling tools described above are capable of providing very detailed information on spatiotemporal disease dynamics. The size and scale of the data and the expertise required to use the simulations demand a user friendly environment that provides an easy way to set up experiments and analyze the results. Recently, a number of visual analytics tools have been developed to support epidemiological research (see Livnat et al., 2010, 2012). We have built a tool called the SIBEL.[6] See http://isisdemo.vbi.vt.edu/didactic/isis.html for a demo version of SIBEL that allows a user to set up detailed factorial experiments (see Fig. 7). Using a simple interface to an underlying digital library, a user can choose from among many preconstructed instances: (i) a social contact network; (ii) a within-host disease progression model; and (iii) a set of interventions. Each intervention requires additional details such as compliance level, subpopulations to which the interventions are applied and intervention triggers. An experiment consists of sweeping one or more parameters across a user-specified range of values. After setting up the experiment, the user is provided access to the results of the simulations. A set of basic analyses are performed automatically and the results are displayed. The standard plots and epidemic curves provide very detailed information about the epidemic. Additional information such as the spatiotemporal dynamics and disease dendrogram (how the disease moved over the social network) is also available. A key aspect of this tool is its simplicity—we can train public health analysts to make effective use of the system in about 3 h. Several other groups are actively developing similar systems. They include: (i) The Biosurveillance Ecosystem (BSV) being developed by DTRA; (ii) The BARD model repository at the Los Alamos National Laboratory; (iii) The Texas Pandemic toolkit being developed at the University of Texas, Austin, http://flu.tacc.utexas.edu/; (iv) The MIDAS funded Apollo project at the University of Pittsburgh and the framework at RTI; (v) The FRED modeling framework at the University of Pittsburgh; and (vi) The EpiC framework being developed by MoBs laboratory at Northeastern University.

6. Earlier versions of SIBEL were called DIDACTIC and ISIS. The change in the name reflects new functionality.

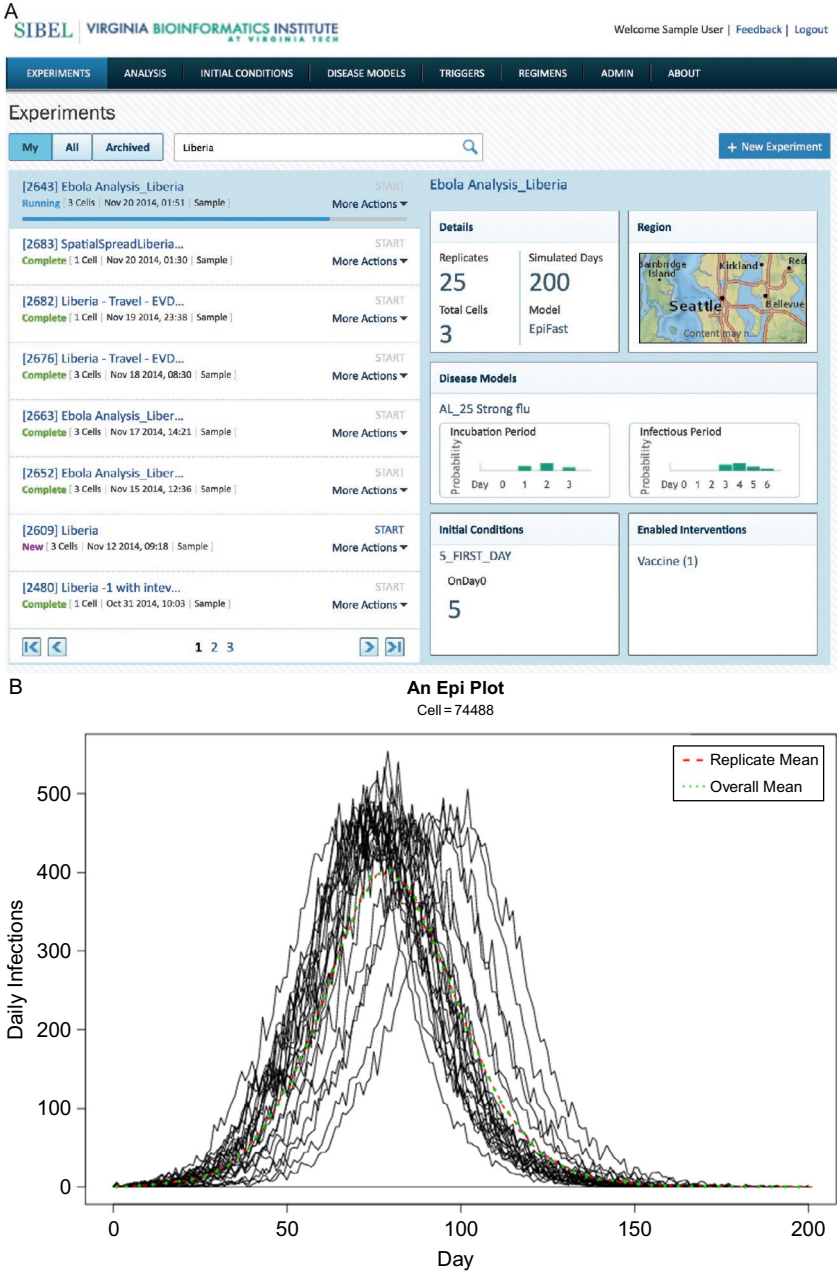


FIGURE 7 SIBEL interface (a). Example simulated epidemic curves (b).

7 SUMMARY

In summary, computational epidemiology is a new and exciting multidisciplinary area with significant challenges of large data and high-performance computing. Recent developments in high-performance pervasive computing have created new opportunities for collecting, integrating, analyzing, and accessing information related to large sociotechnical networks. The advances in network and information science that build on this new capability provide entirely new ways for reasoning and controlling epidemics. Together, they enhance our ability to formulate, analyze, and realize novel public policies pertaining to these complex systems.

There are many other important issues in epidemiology that we have not addressed here. An important area is that of vector borne diseases and zoonotic diseases. Developing computational methods to understand and control these two classes present unique new challenges that one does not have to encounter while studying human–human transmission. Another area where similar ideas have been applied is the epidemiology of noncommunicable diseases, such as obesity and diabetes, and addictions such as smoking. Further, there is a broader class of reaction–diffusion models generalizing the SIR/SIS models, which has been applied to diverse phenomena in economics, sociology, viral marketing, and political science; we refer the readers to Marathe and Vullikanti (2013) for more pointers to these topics.

ACKNOWLEDGMENTS

We thank our external collaborators and members of the Network Dynamics and Simulation Science Laboratory (NDSSL) for their suggestions and comments. S.P. was supported by Ramalingaswami-DBT Fellowship, DST-CMS, and MoS&PI India (Project No. SR/SA/MS:516/07 dated 21/04/2008). The work of A.K.S.V. and M.V.M. has been partially supported by National Science Foundation: NSF NetSE Grant CNS-1011769, NSF SDCI Grant OCI-1032677, Defense Threat Reduction Agency Grant HDTRA1-11-1-0016, DTRA CNIMS Contract HDTRA1-11-D-0016-0001, National Institute of Health Midas Grant 2U01GM070694-09, and by the Intelligence Advanced Research Projects Activity (IARPA) via the US Department of Interior (DoI) National Business Center (NBC) Contract No. D12PC000337. The US government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the US government.

REFERENCES

Allen, L.Z., Ishoey, T., Novotny, M.A., McLean, J.S., Lasken, R.S., Williamson, S.J., 2011. Single virus genomics: a new tool for virus discovery. PLoS One 6 (3), e17722. http://dx.doi.org/10.1371/journal.pone.0017722.

Alon, N., Benjamini, I., Stacey, A., 2004. Percolation on finite graphs and isoperimetric inequalities. Ann. Probab. 32 (3), 1727–1745.

Aspnes, J., Chang, K.L., Yampolskiy, A., 2006. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. J. Comput. Syst. Sci. 72 (6), 1077–1093.

Ayres, D.L., 2012. Beagle: an application programming interface and high-performance computing library for statistical phylogenetics. Syst. Biol. 61 (1), 170–173.

Bandura, A., 1986. Social Foundations of Thought and Action: A Social Cognitive Theory. Englewood Cliffs, NJ, US: Prentice-Hall, Inc.

Barrett, C.L., Beckman, R., Berkbigler, K., Bisset, K., Bush, B., Campbell, K., Eubank, S., Henson, K., Hurford, J., Kubicek, D., Marathe, M., Romero, P., Smith, J., Smith, L., Speckman, P., Stretz, P., Thayer, G., Eeckhout, E., Williams, M., 2000. TRANSIMS (TRansportation ANalysis SIMulation System). Los Alamos National Laboratory. Technical report no. LA-UR-00-1725.

Barrett, C.L., Jacob, R., Marathe, M., 2001. Formal language constrained path problems. SIAM J. Comput. 30 (3), 809–837.

Barrett, C., Smith, J.P., Eubank, S., 2005. Modern Epidemiology Modeling. Scientific American.

Barrett, C., et al., 2008. Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In: Proceedings of the ACM/IEEE Conference on High Performance Computing (SC).

Barrett, C., et al., 2009. Generation and analysis of large synthetic social contact networks. In: Proceedings of the Winter Simulation Conference, pp. 1003–1014.

Barrett, C., Bisset, K., Leidig, J., Marathe, A., Marathe, M., 2010. An integrated modeling environment to study the co-evolution of networks, individual behavior and epidemics. AI Mag. 31 (1), 75–87.

Barzon, L., et al., 2011. Applications of next generation sequencing technologies to diagnostic virology. Int. J. Mol. Sci. 12(11). 7861–7884.

Becker, M., (Ed.), 1974. The health belief model and personal health behavior. Health Education Monogr. 2 (no. 2), 324–508.

Beckman, R., Baggerly, K., McKay., M., 1996. Creating base-line populations. Transport. Res. A Policy Pract. 30, 415–429.

Bisset, K., Chen, J., Feng, X., Vullikanti, A. and Marathe, M. 2009a. EpiFast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In: Proceedings of 23rd ACM International Conference on Supercomputing (ICS′ 09). ACM Press, New York.

Bisset, K., Feng, X., Marathe, M., Yardi, S., 2009b. Modeling interaction between individuals, social networks and public policy to support public health epidemiology. In: Proceedings of the 2009 Winter Simulation Conference, pp. 2020–2031.

Bisset, K., Chen, J., Feng, X., Ma, Y., Marathe, M., 2010a. Indemics: an interactive data intensive framework for high performance epidemic simulation. In: Proceedings of the 24th ACM International Conference on Supercomputing, pp. 233–242.

Bisset, K., et al., 2010b, Indemics: an interactive data intensive framework for high performance epidemic simulation. In: Proceedings of the International Conference on Supercomputing.

Bisset, K., et al., 2011. Interaction-based hpc modeling of social, biological, and economic contagions over large networks. In: Proceedings of Winter Simulation Conference (WSC), pp. 2933–2947.

Blainey, P.C., 2013. The future is now: single-cell genomics of bacteria and archea. FEMS Microbiol. Rev. 37 (3), 407–427.

Borgs, C., Chayes, J.T., Ganesh, A., Saberi, A., 2010. How to distribute antidotes to control epidemics. Random Struct. Algorithms 37, 204–222.

Borozan, I., et al., 2012. CaPSID: a bioinformatics platform for computational pathogen sequence identification in human genomes and transcriptomes. BMC Bioinformatics 13 (206), 1–11.

Brauer, F., van den Driessche, P., Wu, J. (Eds.), 2008. Mathematical Epidemiology. Lecture Notes in Mathematics 1945. Springer Verlag, Berlin, Heidelberg, New York.

Buczak, A.L., Babin, S.M., Feighner, B.H., Koshute, P.T., Lewis, S.H., 2013. Predictive modeling of emerging infections. In: Viral Infections and Global Change, Singh, S.K. (Ed.), John Wiley & Sons, Inc., Hoboken, NJ.

Carley, K.M., Fridsma, D.B., Casman, E., Yahja, A., Altman, N., Chen, L.C., Kaminsky, B., Nave, D., 2006. Biowar: scalable agent-based model of bioattacks. IEEE Trans. Syst. Man Cybernet. A 36 (2), 252–265.

Chakraborty, P., Khadivi, P., Lewis, B., Mahendiran, A., Chen, J., Butler, P., Nsoesie, E., Mekaru, S., Brownstein, J., Marathe, M., Ramakrishnan, N., 2014a, Forecasting a moving target: ensemble models for ILI case count predictions. In: Proceedings of the SIAM International Conference on Data Mining (SDM).

Chakraborty, P., Khadivi, P., Lewis, B., Mahendiran, A., Chen, J., Butler, P., Nsoesie, E.O., Mekaru, S.R., Brownstein, J.S., Marathe, M.V., Ramakrishnan, N., 2014b. Forecasting a moving target: ensemble models for ILI case count predictions. In: Proceedings of the 2014 SIAM International Conference on Data Mining, 28 April 2014, pp. 262–270. http://dx.doi.org/10.1137/1.9781611973440.30.

Chao, D.L., Halloran, M.E., Obenchain, V., Longini Jr., I.M., 2010. FluTE, a publicly available stochastic influenza epidemic simulation model. PLoS Comput. Biol. 6 (1), e1000656.

Chen, E.C., Miller, S.A., DeRisi, J.L., Chiu, C.Y., 2011. Using a pan-viral microarray assay (virochip) to screen clinical samples for viral pathogens. J. Vis. Exp. 50. pii: 2536. http://dx.doi.org/10.3791/2536.

Chiu, C.Y., 2013. Viral pathogen discovery. Curr. Opin. Microbiol. 16, 468–478.

Chung, F., Lu, L., 2002. Connected components in random graphs with given degree sequences. Ann. Combinatorics 6, 125–145.

Coker, R.J., et al., 2011. Towards a conceptual framework to support one health research for policy on emerging zoonoses. Lancet Infect. Dis. 11, 326–331.

Colizza, V., Barrat, A., Barthelemy, M., Vespignani, A., 2006. The role of the airline transportation network in the prediction and predictability of global epidemics. PNAS 103 (7), 2015–2020. doi:10.1073/pnas.0510525103. http://www.pnas.org/content/103/7/2015.full.pdf+html, http://www.pnas.org/content/103/7/2015.abstract.

Cooper, G.E., et al., 2006. Bayesian methods for diagnosing outbreaks. In: Wagner, M.M., Moore, A.W., Aryel, R.M. (Eds.), Handbook of Biosurveillance. Elsevier, New York City, NY.

Cottam, E.M., et al., 2008. Integrating genetic and epidemiological data to determine transmission pathways of foot-and-mouth disease virus. Proc. R. Soc. Lond. B. 275 (1637), 887–895.

Deodhar, S., Bisset, K., Chen, J., Ma, Y., Marathe, M.V., 2012. Enhancing software capability through integration of distinct software in epidemiological systems. In: Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium, pp. 171–180.

Dimitrov, N.B., Meyers, L.A., 2010. Mathematical approaches to infectious disease prediction and control. In: Hasenbein, J.J. (Ed.), INFORMS TutORials in Operations Research 7, pp. 1–25.

Easley, D., Kleinberg, J., 2010. Networks, Crowds and Markets: Reasoning About A Highly Connected World. Cambridge University Press, New York, NY.

Eubank, S., 2002. Scalable, efficient epidemiological simulation. In: SAC '02: Proceedings of the 2002 ACM Symposium on Applied Computing. ACM, New York, NY, pp. 139–145.

Eubank, S., Kumar, V.A., Marathe, M., Srinivasan, A., Wang, N., 2004a. Structural and algorithmic aspects of massive social networks. In: ACM Symposium on Discrete Algorithms (SODA).

Eubank, S., Guclu, H., Anil Kumar, V.S., Marathe, M., Srinivasan, A., Toroczkai, Z. Wang, N., 2004b. Modelling disease outbreaks in realistic urban social networks. Nature 429, 180–184.
Ferguson, N.M., et al., 2005. Strategies for containing an emerging influenza pandemic in southeast Asia. Nature 437, 209–214.
Ferguson, N.M., et al., 2006. Strategies for mitigating an influenza pandemic. Nature 442, 448–452.
Finn, R., Clements, J., Eddy, S., 2011. HMMER web server: interactive sequence similarity searching. Nucleic Acids Res. 39, W29–W37.
Firth, C., Lipkin, W.I., 2013. The genomics of emerging pathogens. Annu. Rev. Genomics Hum. Genet. 14, 281–300.
Fox, G., Mani, D.R., Pyne, S., 2013. Parallel deterministic annealing clustering and its application to LC-MS data analysis. In: Proceedings of the IEEE International Conference on Big Data.
Funk, S., Gilad, E., Watkins, C., Jansen, V.A.A., 2009. The spread of awareness and its impact on epidemic outbreaks. PNAS 106 (16), 6872–6877.
Funk, S., Salathé, M., Jansen, V., 2010. Modelling the influence of human behaviour on the spread of infectious diseases: a review. J. R. Soc. Interface 7, 1247–1256. doi:10.1098/rsif.2010.0142.
Ganesh, A., Massoulie, L., Towsley, D., 2005. The effect of network topology on the spread of epidemics. Proc. INFOCOM. 2, 1455–1466.
Gardner, S.N., et al., 2010. A microbial detection array (MDA) for viral and bacterial detection. BMC Genomics 11, 668.
Germann, T.C., Kadau, K., Longini, I.M., Macken, C.A., 2006. Mitigation strategies for pandemic influenza in the united states. Proc. Natl. Acad. Sci. U.S.A. 103 (15), 5935–5940. doi:10.1073/pnas.0601266103.
Ginsberg, J., Mohebbi, M.H., Patel, R.S., Brammer, L., Smolinski, M.S., Brilliant, L., 2009. Detecting influenza epidemics using search engine query data. Nature 457, 1012–1014.
Gomez-Rodriguez, M., Leskovec, J., Krause, A., 2010. Inferring networks of diffusion and influence. In: Proceedings of the 16th ACM KDD.
Grenfell, B.T., et al., 2004. Unifying the epidemiological and evolutionary dynamics of pathogens. Science 303, 327–332.
Hall, I.M., Gani, R., Hughes, H.E., Leach, S., 2007. Real-time epidemic forecasting for pandemic influenza. Epidemiol. Infect. 135 (3), 372–385.
Hens, N., Shkedy, Z., Aerts, M., Faes, C., Van Damme, P., Beutels, P., 2012. Modeling infectious disease parameters based on serological and social contact data. A modern statistical perspective. Statistics for Biology and Health. Springer, New York, NY. doi:10.1007/978-1-4614-4072-7.
Holmes, E.C., 2009. The Evolution and Emergence of RNA Viruses. Oxford University Press, New York, NY.
Holmes, E.C., Grenfell, B.T., 2009. Discovering the phylodynamics of RNA viruses. PLoS Comput. Biol. 5 (10), e1000505.
Jombart, T., et al., 2014. OutbreakTools: new platform for disease outbreak analysis using the R software. Epidemics, 7, 28–34.
Jones, K.E., et al., 2008. Global trends in emerging infectious diseases. Nature 451, 990–993.
King, D.A., et al., 2006. Infectious diseases: preparing for the future. Science 313, 1392–1393.
Kostic, A., et al., 2011. PathSeq: software to identify or discover microbes by deep sequencing of human tissue. Nat. Biotechnol. 29, 393–396.
Last, J., 2001. A Dictionary of Epidemiology, fourth ed. Oxford University Press, New York.
Li, L., Alderson, D., Willinger, W., Doyle, J., 2004. A first principles approach to understanding the internet's router level topology. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 3–14.

Li, W.D., et al., 2005. Bats are natural reservoirs of SARS-like coronaviruses. Science 310 (5748), 676–679.

Lipkin, W.I., 2009. Microbe hunting in the 21st century. Proc. Natl. Acad. Sci. U.S.A. 106 (1) 6–7.

Lipkin, W.I., 2013. The changing face of pathogen discovery and surveillance. Nat. Rev. Microbiol. 11, 133–141.

Lipsitch, M., Cohen, T., Cooper, B., Robins, J.M., Ma, S., James, L., Gopalakrishna, G., Chew, S., Tan, C.C., Samore, M.H., Fisman, D., Murray, M., 2003. Transmission dynamics and control of severe acute respiratory syndrome. Science 300, 1966–1970.

Lipsitch, M., et al., 2009. Managing and reducing uncertainty in an emerging influenza pandemic. N. Engl. J. Med. 361 (2), 112–115. http://www.nejm.org/doi/pdf/10.1056/NEJMp0904380, http://www.nejm.org/doi/full/10.1056/NEJMp0904380.

Livnat, Y., Gesteland, P., Benuzillo, J., Pettey, W., Bolton, D., Drews, F., Kramer, H., Samore, M., 2010. Epinome—a novel workbench for epidemic investigation and analysis of search strategies in public health practice. Proc. Annu. Am. Med. Inform. Assoc. Sympos. 647–651.

Livnat, Y., Rhyne, T., Samore, M., 2012. Epinome: A visual-analytics workbench for epidemiology data. IEEE Comput. Graph. Appl. 32 (2), 89–95.

Loman, N.J., et al., 2012. Performance comparison of bench-top high-throughput sequencing platforms. Nat. Biotechnol. 30, 434–439.

Longini, I.M., et al., 2005. Containing pandemic influenza at the source. Science 309, 1083–1087.

Ma, Y., Bisset, K.R., Chen, J., Deodhar, S., Marathe, M.V., 2011a, Formal specification and experimental analysis of an interactive epidemic simulation framework. In: Proceedings of the 2011 International Workshop on Extreme Scale Computing Application Enablement—Modeling and Tools (ESCAPE).

Ma, Y., Bisset, K.R., Chen, J., Deodhar, S., Marathe, M.V., 2011b, Efficient implementation of complex interventions in large scale epidemic simulations. In: Proceedings of the Winter Simulation Conference.

Marathe, M., Vullikanti, A., 2013. Computational epidemiology. Commun. ACM. 56 (7), 88–96.

Metzker, M.L., 2009. Sequencing technologies—the next generation. Nat. Rev. Genet. 11 (11), 31–46.

Mniszewski, S.M., Del Valle, S.Y., Stroud, P.D., Riese, J.M., Sydoriak, S.J., 2008. EpiSimS simulation of a multi-component strategy for pandemic influenza. In: Proceedings of the 2008 Spring Simulation Multiconference, SpringSim '08. Society for Computer Simulation International, San Diego, CA, pp. 556–563. http://portal.acm.org/citation.cfm?id=1400549.1400636.

Mokili, J.L., et al., 2012. Metagenomics and future perspectives in virus discovery. Curr. Opin. Virol. 2, 63–77.

Morse, S.S., et al., 2012. Prediction and prevention of the next pandemic zoonosis. Lancet 380, 1956–1965.

Mortveit, H.S., Reidys, C.M., 2007. An Introduction to Sequential Dynamical Systems, Universitext. Springer Verlag, New York.

Mossong, J., et al., 2008. Social contacts and mixing patterns relevant to the spread of infectious diseases. PLoS Med. 5 (3), e74. http://dx.doi.org/10.1371/journal.pmed.0050074.

Naccache, S.N., et al., 2014. A cloud-compatible bioinformatics pipeline for ultrarapid pathogen identification from next-generation sequencing of clinical samples. Genome Res. 24, 1180–1192.

Naeem, R., Rashid, M., Pain, A., 2013. READSCAN: a fast and scalable pathogen discovery program with accurate genome relative abundance estimation. Bioinformatics 29 (3), 391–392.

Neil, D., Moore, A., Cooper, G., 2005. A bayesian scan statistic for spatial cluster detection. In: Proceedings of the National Syndromic Surveillance Conference.

Netrapalli, P., Sanghavi, S., 2012. Learning the graph of epidemic cascades. In: ACM SIGMETRICS.

Newman, M.E.J., 2003. The structure and function of complex networks. SIAM Rev. 45, 167–256.

Nishiura, H., 2011. Real-time forecasting of an epidemic using a discrete time stochastic model: a case study of pandemic influenza (H1N1-2009). BioMed. Eng. Online 10 (1), 15.

Nsoesie, E.O., Beckman, R., Marathe, M., Lewis, B., 2011. Prediction of an epidemic curve: a supervised classification approach. Stat. Commun. Infect. Dis. 3 (1), 5.

Nsoesie, E.O., Brownstein, J.S., Ramakrishnan, N., Marathe, M., 2013. A systematic review of studies on forecasting the dynamics of influenza outbreaks. Influenza Other Respir. Viruses 8 (3), 309–316.

Ohkusa, Y., Sugawara, T., Taniguchi, K., Okabe, N., 2011. Real-time estimation and prediction for pandemic A/H1N1(2009) in Japan. J. Infect. Chemother. 17 (4), 468–472.

Palacios, G., et al., 2007. Panmicrobial oligonucleotide array for diagnosis of infectious diseases. Emerg. Infect. Dis. 13, 73–81.

Parker, J., Epstein, J.M., 2012. A distributed platform for global-scale agent-based models of disease transmission. ACM Trans. Model. Comput. Simulation 22 (1), Article No. 2.

Pastor-Satorras, R., Vespignani, A., 2001, Apr. Epidemic spreading in scale-free networks. Phys. Rev. Lett. 86, 3200–3203. http://link.aps.org/doi/10.1103/PhysRevLett.86.3200.

Patz, J.A., et al., 2004. Unhealthy landscapes: policy recommendation on land use change and infectious disease emergence. Environ. Health Perspect. 112 (10), 1092–1098.

Perumalla, K., Seal, S., 2011. Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. Simulation 88 (7), 768–783.

Pybus, O.G., Rambaut, A., 2009. Evolutionary analysis of the dynamics of viral infectious disease. Nat. Rev. Genet. 10, 540–550.

Pyne, S., et al., 2009. Automated high-dimensional flow cytometric data analysis. Proc. Natl. Acad. Sci. U.S.A. 106.

Pyne, S., et al., 2014. Joint modeling and registration of cell populations in cohorts of high-dimensional flow cytometric data. PLoS One. 9, e100334.

Ray, S., Pyne, S., 2012. A computational framework to emulate the human perspective in flow cytometric data analysis. PLoS One. 7, e35693.

Riesenfeld, C.S., 2004. Metagenomics: genomic analysis of microbial communities. Annu. Rev. Genet. 38, 525–552.

Salathé, M., et al., 2010. A high-resolution human contact network for infectious disease transmission. PNAS 107 (51), 22020–22025. ISSN 1091-6490. http://dx.doi.org/10.1073/pnas.1009094108.

Salathé, M., et al., 2012. Digital epidemiology. PLoS Comput. Biol. 8 (7), e1002616.

Schwind, J.S., et al., 2014. Capacity building efforts and perceptions for wildlife surveillance to detect zoonotic pathogens: comparing stakeholder perspectives. BMC Public Health 14, 684.

Shah, D., Zaman, T., 2010. Rumors in a network: who is the culprit? ACM SIGMETRICS.

Shaman, J., Karspeck, A., 2012. Forecasting seasonal outbreaks of influenza. Proc. Natl. Acad. Sci. U.S.A. 109 (50), 20425–20430. http://www.pnas.org/content/109/50/20425.full.pdf+html, http://www.pnas.org/content/109/50/20425.abstract.

Shaman, J., Goldstein, E., Lipsitch, M., 2010a. Absolute humidity and pandemic versus epidemic influenza. Am. J. Epidemiol. 173 (2), 127–135.

Shaman, J., Pitzer, V.E., Viboud, C., Grenfell, B.T., Lipsitch, M., 2010b. Absolute humidity and the seasonal onset of influenza in the continental United States.. PLoS Biol. 8 (2), e1000316.

Suchard, M.A., Rambaut, A., 2009. Many-core algorithms for statistical phylogenetics. Bioinformatics 25, 1370–1376.

Taylor, L.H., et al., 2001. Risk factors for human disease emergence. Philos. Trans. R. Soc. Lond. B Biol. Sci. 356, 983–989.

Tizzoni, M., Bajardi, P., Poletto, C., Ramasco, J., Balcan, D., Goncalves, B., Perra, N., Colizza, V., Vespignani, A., 2012. Real-time numerical forecast of global epidemic spreading: case study of 2009 A/H1N1pdm. BMC Med. 10 (1), 165. ISSN 1741-7015. http://www.biomedcentral.com/1741-7015/10/165.

Vaidyanathan, G., 2011. Virus hunters: catching bugs in the field. Cell 147, 1209–1211.

Volz, E.M., et al., 2013. Viral phylodynamics. PLoS Comput. Biol. 9, e1002947.

Wang, Y., Chakrabarti, D., Wang, C., Faloutsos, C., 2003. Epidemic spreading in real networks: an eigenvalue viewpoint. In: Proceedings of SRDS.

Chapter 9

Big Data Driven Natural Language Processing Research and Applications

Venkat N. Gudivada[*,1], **Dhana Rao**[†], **Vijay V. Raghavan**[‡]
[*] *East Carolina University, Greenville, North Carolina, USA*
[†] *Marshall University, Huntington, West Virginia, USA*
[‡] *University of Louisiana at Lafayette, Louisiana, USA*
[1] *Corresponding author: e-mail: gudivadav15@ecu.edu*

ABSTRACT
Due to the inherent complexity of natural languages, many natural language tasks are ill-posed for mathematically precise algorithmic solutions. To circumvent this problem, statistical machine learning approaches are used for natural language processing (NLP) tasks. The emergence of Big Data enables a new paradigm for solving NLP problems—managing the complexity of the problem domain by harnessing the power of data for building high quality models.

This chapter provides an introduction to various core NLP tasks and highlights their data-driven solutions. Second, a few representative NLP applications, which are built using the core NLP tasks as the underlying infrastructure, are described. Third, various sources of Big Data for NLP research are discussed. Fourth, Big Data driven NLP research and applications are outlined. Finally, the chapter concludes by indicating trends and future research directions.

Keywords: Big data, Natural language processing, Statistical models

1 INTRODUCTION

Natural language processing (NLP) is an interdisciplinary domain encompassing linguistics, information retrieval, machine learning, probability and statistics. It is concerned with analyzing, understanding, and interpreting written text and spoken language as well as using natural languages for communicating with computers (Indurkhya and Damerau, 2010; Jurafsky and Martin, 2009; Manning

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00009-5

and Schutze, 1999). The research in text and spoken mediums mostly progressed rather independently.

1.1 Emergence of Big Data

The synergistic confluence of pervasive sensing, high performance computing, and gigabit networks is generating data at unprecedented levels. These massive datasets are referred to as *Big Data*. The term *Data Science* refers to a set of new algorithms and approaches for advancing science through Big Data. Dhar (2013) defines Data Science as *the systematic study of the extraction of generalizable knowledge from data*. The Data Science is a rapidly emerging field which provides innovative algorithms and workflows for analyzing, visualizing, and interpreting Big Data to enable scientific breakthroughs (Hey et al., 2009).

1.2 Big Data Driven NLP Research and Applications

Physicist Eugene Wigner's 1960 essay, *The Unreasonable Effectiveness of Mathematics in the Natural Sciences*, provides compelling examples to demonstrate the extent to which abstract mathematical concepts hold validity far beyond the contexts in which they were developed (Wigner, 1960). A sequel to this essay in the context of Big Data, *The Unreasonable Effectiveness of Data*, argues that the accurate selection of a mathematical model ceases its importance when compensated by *big enough* data (Halevy et al., 2009). This insight is particularly important for tasks that are ill-posed for mathematically precise algorithmic solutions. Such tasks encompass several problems in NLP including language modeling, part-of-speech (POS) tagging, named entity recognition (NER), and parsing.

Ill-posed tasks also abound in computer vision, autonomous vehicle navigation, image and video processing applications. Big Data enables a new paradigm for solving these problems—managing the complexity of the problem domain by building simple but high quality models by harnessing the power of massive data.

The ability to effectively process massive datasets has become integral to a broad range of academic disciplines and business enterprises. For example, currently about 16 years of video is uploaded daily to YouTube (YouTube, 2015). This data provides both opportunities and challenges for NLP research and applications.

Searching for a given speaker in YouTube videos is a difficult task given the vastness of data and query latency requirements. Localization of YouTube in 61 countries and across 61 languages adds another dimension of complexity to the speaker search task. On the other hand, IBM Watson, a question answering (QA) system with natural language interface, demonstrated how Big Data can be effectively utilized in performing complex tasks. NLP research is typically data driven and Big Data is transforming the way current NLP research is conducted.

The adage—a tool without theory is blind and a theory without tool is useless—holds in the Big Data context too. Big Data enables scientists to overcome problems associated with small data samples in several ways such as relaxing the assumptions of theoretical models, avoiding overfitting of models to *training data*, dealing with noisy training data, and providing ample *test data* to validate models.

1.3 Organization of the Chapter

The overarching goal of this chapter is to describe how Big Data is changing the course of NLP research and enabling new applications. Various NLP tasks and representative *applications* are discussed in Sections 2 and 3. These sections set the context and backdrop for the rest of the chapter. Big Data sources for NLP research are presented in Section 4. Big Data-enabled NLP research and *applications* are discussed in Section 5. NLP trends and future research directions are indicated in Section 6. Finally, Section 7 concludes the chapter.

2 NLP CORE TASKS

NLP is difficult due to the inherent ambiguity in language. The ambiguity is manifested at many processing steps including the acoustic level (e.g., difficulty in recognizing speech due to speaker dependent syllable segmentation), syntactic level (e.g., multiple parse trees, each leading to a different interpretation), semantic level (e.g., word sense ambiguity—same word has different meanings depending on the context), and discourse level (e.g., anaphora—a noun or pronoun in a sentence co-referees to some other discourse entity). Shown in Table 1 are core NLP tasks that form the foundation for NLP applications.

2.1 Statistical Language Modeling

Language modeling is a fundamental NLP task. Language models are used as building blocks in other NLP tasks and applications including speech recognition, optical character recognition, handwriting recognition, machine translation (MT), spelling correction, text summarization, and QA.

2.1.1 Probability Distribution of Strings

The purpose of statistical language modeling is to construct a probability distribution function that assigns a probability to every string in the language. In other words, it is a probability distribution over all the strings in the language. We use the terms *phrase*, *string*, and *sentence* interchangeably. The higher the probability of a sentence, the more likely the sentence is a valid construction in the language. More formally, let $\mathcal{V}$ be a finite vocabulary and $\mathcal{V}^*$ be the set of strings in the language defined by $\mathcal{V}$. For example, $\mathcal{V}$ = {language, machine, modeling, natural, translation} and $\mathcal{V}^*$ = {language, machine, modeling, natural,

translation, natural language, translation language, natural language modeling, machine translation, modeling machine translation, modeling natural language, . . . }. The probability distribution function p for this language is one that satisfies:

$$\sum_{x \in \mathcal{V}^*} p(x) = 1, \text{ and } p(x) \geq 0 \text{ for all } x \in \mathcal{V}^*$$

2.1.2 Maximum Likelihood Estimates

How do we construct or *learn* p? One way is to use training data—a sample of example sentences—and compute *maximum likelihood* estimates. Then we can use p to compute probabilities for any sentence in the language.

We use the notation $x_{[1:n]}$ to denote a sentence of length n, where x_1 is the first word, x_2 is the second word, and so on. Constructing a phrase by choosing

TABLE 1 Core Natural Language Processing Tasks

Core Task	Task Description
Language Modeling	Assigns a probability to every string in the language. It is a probability distribution over all the strings in the language. Applications include spelling correction, decoding secret codes, and word autocompletion in smart phones and other hand-held devices.
Word Segmentation	In many languages there is no explicit word delimiter. This makes extracting words from natural language text difficult. Extracting words from continuous speech is also a word segmentation problem. Word segmentation is the first step in both speech and text based NLP.
Part-of-Speech (POS) Tagging	Assigns a POS tag for every word in a document. POS tagging is the lowest level of syntactic analysis. POS tags are used in many subsequent activities such as syntactic parsing, word-sense disambiguation, and text-to-speech synthesis.
Named Entity Recognition	Identifies names of people, places, organizations, and other entities of interest in text. These results are used in other tasks such as co-reference resolution, word-sense disambiguation, semantic parsing, question answering, dialog systems, textual entailment, information extraction, information retrieval, and text summarization.
Parsing	Generates syntactic structures that depict relationships between words in a piece of text. A parse tree is one such structure. Parse trees are used in other tasks such as named entity recognition, information extraction, and machine translation.

one word at a time is viewed as a stochastic process. Consider a sequence of random variables $X_1, X_2, \ldots, X_n$, each of which takes a value randomly from the set $\mathcal{V}^*$. Let $x_{[1:n]}$ be a sentence in the language. If we assume that the words in the sentence $x_{[1:n]}$ correspond to the random variables $X_1, X_2, \ldots, X_n$, our goal is to model $p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$.

Let $q(x_1)$ denote the probability of occurrence of the word x_1. Also, let $q(x_i|x_{[1:i-1]})$ denote the probability of occurrence of the i^{th} word x_i given that the previous $i-1$ words are $x_{[1:i-1]}$. This joint probability of words is computed using the chain rule as:

$$p(x_1 x_2 \cdots x_n) = q(x_1) \prod_{i=2}^{n} q(x_i|x_{[1:i-1]}) \tag{1}$$

For example, p(natural language processing) = q(natural) $\times$ q(language | natural) $\times$ q(processing|natural language). We estimate probability distribution of q using counts of phrase occurrences in the training data. For instance:

$$q(\text{processing} \mid \text{natural language}) = \frac{count(\text{natural language processing})}{count(\text{natural language})} \tag{2}$$

For example, p(natural language) = 10^{-7} and p(natural language modeling) = 10^{-8}. Though these counts can be computed easily in the training data, there are two problems. First, there are too many possible phrases. Second, it is very unlikely that the training data will provide counts to estimate the probabilities of all possible phrases in the language reliably. The *Markov assumption* is used to overcome these problems, which is discussed next.

2.1.3 Markov Assumption

In probability theory, *Markov property* refers to memoryless property of a stochastic process. The latter has the Markov property if the probability distribution of future states of the process conditioned on both the past and present states depends only on the present state. In other words, predicting the next word in a sentence depends only on the current word, and not on the words that came before the current word. *Markov property* holds in a model if the values in any state are influenced only by the values of the immediately preceding or a small number of immediately preceding states. Hidden Markov model (HMM) is an example in which it is assumed that the Markov property holds. Using the Markov assumption, Eq. (1) is rewritten as:

$$p(x_1 x_2 \cdots x_n) \approx \prod_{i=1}^{n} p(x_i \mid x_{i-k}\ x_{(i-k)+1}\ x_{(i-k)+2} \cdots x_{i-1}),\ 1 \leq k < i \tag{3}$$

When $k = 1$, the values of the current state are dependent on just the immediately preceding state (aka *first-order Markov process*). Likewise, when

$k = 2$, the values of the current state are dependent on just the two immediately preceding states (aka *second-order Markov process*). We approximate each component in the product of Eq. (3) as:

$$p(x_i \mid x_1 x_2 \cdots x_{i-1}) \approx q(x_i \mid x_{i-k}\ x_{(i-k)+1}\ x_{(i-k)+2} \cdots x_{i-1}),\ 1 \leq k < i \quad (4)$$

2.1.4 Unigram, Bigram, and Trigram Models

Depending on the value of k chosen in Eq. (4), we get different language models. For example, if $k = 0$ we get the *unigram language model*:

$$p(x_1 x_2 \cdots x_n) \approx \prod_{i=1}^{n} q(x_i)$$

Under this model, the probability of observing a given word does not depend on the context—the words that precede the given word. Similarly, by setting $k = 1$ and $k = 2$, we get the *bigram* and *trigram* language models:

$$p(x_1 x_2 \cdots x_n) \approx \prod_{i=1}^{n} q(x_i \mid x_{i-1})$$

$$p(x_1 x_2 \cdots x_n) \approx \prod_{i=1}^{n} q(x_i \mid x_{i-2} x_{i-1})$$

In the bigram model, the probability of observing a given word depends on the immediately preceding word; in the trigram model, it depends on the two immediately preceding words.

2.1.5 Smoothing Parameter Values

Sparse data problems abound in language modeling. To circumvent these problems, parameter values are *smoothed* using linear interpolation and discounting methods. Linear interpolation methods estimate parameter values by a linear combination of maximum likelihood estimates from the unigram, bigram, and trigram models as:

$$q(x_i \mid x_{i-2} x_{i-1}) = \lambda_1\ q_{\text{ml}}(x_i \mid x_{i-2}, x_{i-1}) + \lambda_2\ q_{\text{ml}}(x_i \mid x_{i-1}) + \lambda_3\ q_{\text{ml}}(x_i)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$, $\lambda_i \geq 0, 1 \leq i \leq 3$, and q_{ml} denotes maximum likelihood estimation of parameters. Values of λ_1, λ_2, and λ_3 are determined by optimizing a function whose domain is $(\lambda_1, \lambda_2, \lambda_3)$.

2.1.6 Evaluating Language Models

Evaluation is an essential component of language modeling. An evaluation metric gives us a measure to compare different language models. Language

model parameters are learned from the *training data*. *Test data*, which is different from the training data, is used for model evaluation.

Perplexity is a measure for evaluating language models. Conceptually, the perplexity value indicates how many choices are available to select the next word x_{i+1}, given a partial sentence $x_1 x_2 \cdots x_i$. Therefore, a language model that has smaller perplexity is considered superior to the one that has a larger value. Intuitively, a good language model should assign higher probabilities to frequently observed sentences and lower values to infrequently occurring ones.

Let $s_1, s_2, s_3, \ldots, s_m$ be the m sentences and M be the total number of words in the test data. Recall that $p(s_i)$ is the probability of observing s_i in the language. Perplexity is defined as 2^{-l} where:

$$l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

Though perplexity is a useful *intrinsic* evaluation measure, it is not effective as a metric unless the test data looks just like the training data. *Extrinsic evaluation* measures compare language models based on their performance on authentic tasks such as spelling correction, machine translation, and speech recognition. Counts such as the number of misspelled words corrected, the number of words that are translated correctly, and the number of words in the speech that are correctly recognized are used in computing extrinsic evaluation measures.

2.1.7 Log Linear Models for Language Modeling

Recall that trigram language models make use of only unigram, bigram, and trigram estimates. However, there are numerous other features which can be used in language modeling. Such features include POS tags, word suffixes and prefixes, and words that occur somewhere before the current word (aka *long-range features*). A set of such features form a *feature vector*. A class of models referred to as *log-linear models* employ feature vectors in conjunction with *parameter vectors* (aka *weight vectors*) to learn the probability distribution function. With smoothing of parameter values, log-linear models have shown to perform better than trigram models. However, log-linear models entail greater computational cost.

2.1.8 Big Data for Building Language Models

Language models are at the center of simple and elegant solutions to various NLP problems. They are developed using the one-trillion word tokens of text extracted from publicly accessible Web pages (Brants and Franz, 2012). The trillion-word dataset summarizes the Web pages content by counting the number of occurrences of each word, and two-, three-, four-, and five-word sequences. These sequences are called n-grams. The frequency of occurrence of individual

words is referred to as *unigram* counts. Likewise, count of two-word sequences is called *bigram* counts; and *trigrams* correspond to the count of three-word sequences.

How a large natural language corpus data was used to elegantly solve three difficult NLP problems using simple probabilistic models is illustrated in Norvig (2009). The problems solved are spelling correction, decoding secret codes, and word segmentation. We discuss spelling correction in this section and word segmentation is discussed in the next subsection.

Spelling correction problem involves, for a given typed word w, determining what word c was most likely intended. For example, if w is "refered", then c should be "referred". The problem is essentially choosing that c that maximizes $p(c \mid w)$. The latter is computed using the Bayes's theorem as:

$$\underset{c}{\text{argmax}}\, p(c \mid w) = \underset{c}{\text{argmax}}\, p(w \mid c)p(c)$$

$p(c)$ is the probability that c is the intended word and is called the language model. $p(w \mid c)$ is the probability that word w was typed when the intended word is c. This is referred to as the *error model* or *noisy channel model*. The error model is also computed using a list of misspellings in the form of (c, w) pairs.

2.2 Word Segmentation

The script for many natural languages is based on the Roman or Latin alphabet. Word segmentation is made explicit in these languages by using space character as the word delimiter. There are many more languages whose script is based on other alphabets. For example, the Brahmic scripts are a family of Abugida writing systems. Word segmentation is a difficult problem in many of these languages as there is no explicit delimiter.

2.2.1 Approaches to Word Segmentation

Word segmentation is one of the NLP problems that has been studied since the early days of the field. A stochastic finite-state model for segmenting Chinese text is described in Sproat et al. (1994). A trainable rule-based algorithm for performing language-independent word segmentation is discussed in Palmer (1997). In a recent study, a general statistical framework—a global linear model—is applied to word segmentation (Zhang and Clark, 2011). In a related work (Wang et al., 2011), Web scale NLP using a URL word breaking case study is presented. This study unifies the state-of-the-art word breaking techniques under the Bayesian minimum risk framework.

Building statistical models for word segmentation using the *trillion-word corpus* (Brants and Franz, 2012) is demonstrated in Norvig (2009). For segmenting phrases such as *naturallanguageprocessing*, a simple n-gram look up will suffice. For larger phrases, unigram-, bigram-, and trigram-based language models are used. By making the assumption that each word is independent of the

others, it is not necessary to consider all combinations of words. They consider every possible way to split the text into a first word followed by rest of the remaining text. For each split, the best way to segment the remaining phrase is computed. The split that corresponds to the highest $p(\text{first}) \times p(\text{remaining})$ is the best.

2.3 POS Tagging

A POS refers to a category of words which have similar grammatical properties. Words that are assigned to the same POS category generally play similar roles within the grammatical structure of sentences. Assigning a correct POS label for each word in a document is called *POS tagging*.

The primary POS categories in English language are noun, pronoun, adjective, determiner, adverb, verb, preposition, conjunction, and interjection. However, NLP tasks require more granular categories. For example, the Penn Treebank tagset contains 36 POS tags and 12 other tags for punctuation and currency symbols (Marcus et al., 1993). The Penn Treebank tagset is a subset of the tags used with the original Brown corpus (Francis and Kucera, 1964). C5 is another tagset used for the British National Corpus (BNC), which has 61 tags (Leech et al., 1994). There is no generally agreed upon standard for POS tagsets.

The POS tags are also called grammatical tags, word classes, word types, and lexical categories. It is important to keep the distinction between *word types* and *word tokens*, whose distinction is similar to that of a *class* and *object* in object-oriented programming.

POS tagging is considered as the lowest level of syntactic analysis. POS tags are used for many subsequent activities including syntactic parsing (Klein, 2005) and word-sense disambiguation (Navigli, 2009). They are also used in text-to-speech synthesis to correctly pronounce words.

2.3.1 Approaches to POS Tagging

Algorithms for POS tagging fall into two broad categories: *rule based* (Brill, 1995) and *stochastic*. Overall, stochastic algorithms perform more effectively than the rule-based ones. Stochastic POS algorithms are based on supervised learning models such as HMM, log-linear model (aka Maximum Entropy Markov Model), and conditional random field (CRF) (Sutton and McCallum, 2012).

Supervised learning problems require m training examples: $(x^i, y^i), 1 \leq i \leq m$, where x^i is input (e.g., a sentence in the language $x_1 x_2 \cdots x_n$) and y^i is a label (e.g., the POS tag sequence for the sentence). The task is to learn a function f which maps inputs x to labels $f(x)$. That is, $f : \mathcal{X} \longrightarrow \mathcal{Y}$, where $\mathcal{X}$ is the set of all input sentences and $\mathcal{Y}$ is the set of all tag sequences $y_1 y_2 \cdots y_n$.

2.3.2 Generative and Noisy-Channel Models

The function f can be realized as a *conditional* or *generative* model. Under the conditional model, a probability distribution $p(y|x)$ is *learned* from training examples. Given a test input x, define:

$$f(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(x|y)$$

Generative models are based on computing joint probability—learning a distribution $p(x, y)$ from training examples. For many cases, $p(x, y)$ can be expressed as:

$$p(x, y) = p(y)p(x|y)$$

Training data is used to estimate the models for $p(y)$ and $p(x|y)$. Also, using Bayes rule, it can be shown that:

$$f(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, p(y)p(x\,|\,y)$$

Models that decompose $p(x, y)$ into $p(y)$ and $p(x\,|\,y)$ are referred to as *noisy-channel* models.

Given a sentence $x_1x_2 \cdots x_n$ and a POS tag sequence $y_1y_2 \cdots y_n$, a generative tagging model $p(x_1x_2 \cdots x_n, y_1y_2 \cdots y_n)$ assigns a probability which indicates the degree to which the POS tag sequence reflects the true tag sequence for the sentence. Now we can define the function that maps sentences to POS tag sequences as:

$$f(x_1x_2 \cdots x_n) = \underset{y_1y_2 \cdots y_n}{\operatorname{argmax}}\, p(x_1x_2 \cdots x_n, y_1y_2 \cdots y_n) \tag{5}$$

Trigram hidden Markov model (Trigram HMM) is an important generative model for the POS tagging problem. Let $y_0 = y_{-1} = *$, and $y_{n+1} = \text{STOP}$. The literals $*$ and *STOP* are special values to enable uniform treatment of the POS tags at the sentence boundaries. The term $p(x_1x_2 \cdots x_n, y_1y_2 \cdots y_{n+1})$ in Eq. (5) in the context of Trigram HMM can be written as:

$$p(x_1x_2 \cdots x_n, y_1y_2 \cdots y_n) = \prod_{i=1}^{n+1} q(y_i\,|\,y_{i-2}y_{i-1}) \prod_{i=1}^{n} e(x_i\,|\,y_i) \tag{6}$$

The parameter q is computed using the maximum likelihood estimate by counting trigrams and bigrams of POS tags similar to the process we used for language modeling. It is often useful to smooth the parameter q values using the linear interpolation method discussed in Section 2.1. The second parameter e is computed using the maximum likelihood estimate as:

$$e(x\,|\,y) = \frac{c(y \rightsquigarrow x)}{c(y)}$$

where $c(y)$ is the number of times the tag y is seen the training data, and $c(y \rightsquigarrow x)$ is the number of times the tag y is seen paired with the word x in the training

data. $e(x|y)$ is unreliable if the word x is infrequent, or even worse if the word is not seen in the training data.

The right hand side of Eq. (5) is efficiently computed using the Viterbi algorithm, which uses the dynamic programming technique. HMM taggers are simple to train and perform relatively well. They are evaluated extrinsically using tag accuracy. Even using a simple approach such as tagging a word with its most frequent tag and tagging unknown words as nouns gives 90% accuracy. This performance is referred to as the *baseline*. Current taggers achieve about 97% tag accuracy.

2.3.3 Log-Linear Models

Log-linear models for POS tagging define p as a conditional distribution: $p(t_1, t_2, \ldots, t_n|w_1, w_2, \ldots, w_n)$, where $t_1, t_2, \ldots, t_n$ is the tag sequence corresponding to the word sequence $w_1, w_2, \ldots, w_n$. The most likely tag sequence for a given word sequence is defined as:

$$t^*_{[1:n]} = \underset{t_{[1:n]}}{\text{argmax}}\, p(t_{[1:n]} \,|\, w_{[1:n]})$$

where the notation $t_{[1:n]}$ represent a tag sequence of length n and t_i is the i^{th} tag in the sequence. Using the chain rule for conditional probability and assuming independence of tags in a sequence, we have:

$$p(t_{[1:n]} \,|\, w_{[1:n]}) = \prod_{j=1}^{n} p(t_j \,|\, w_1, w_2, \ldots, w_n, t_{j-2}, t_{j-1}) \tag{7}$$

where $t_{-1} = t_0 = *$ to enable uniform processing for the first tag. The parameter p is computed using the *feature* and *parameter* vectors (Ratnaparkhi, 1996). The feature vector is defined by considering various factors such as (word, tag) pairs, spelling and contextual features in the training data. The parameter vector (aka *weights vector*) is learned from the training data using the Viterbi algorithm. Log-linear models perform relatively better than their HMM counterparts. Also, it is easy to incorporate various features such as previous word, next word, prefixes, and suffixes in a systematic manner.

2.3.4 Big Data and POS Tagging

Foster et al. (2011) describe the evaluation of a POS tagger and SVMTool on Twitter data. The latter comprises a corpus of 60 million tweets on 50 themes such as politics, business, sport, and entertainment. The corpus is collected using the public Twitter API between February and May in 2009. First they manually create a treebank of 519 syntactically annotated sentences taken from the tweets. This data is divided into a development set (269 sentences) and a test set (remaining 250 sentences). The SVMTool trained on a Wall Street Journal dataset but applied on the Twitter development set performed POS tagging at 84.1% accuracy. However, the same SVMTool when applied on Wall Street

Journal text performed the POS tagging at 96.3% accuracy. This demonstrates the problems associated with domain adaptation. These issues are addressed in Gimpel et al. (2011) and the POS tagger in this work achieved an accuracy of 92.2% on Twitter data.

2.4 Named Entity Recognition

The goal of NER task is to identify names of people, locations, organizations, and other entities of interest in text documents (Nadeau and Sekine, 2007). NER results are used in other tasks and applications including co-reference resolution, word-sense disambiguation, semantic parsing, QA, dialog systems, textual entailment, information extraction (IE), information retrieval, and text summarization. NER results are also used to enhance the POS tagging task and vice versa.

Named entities are often not simply singular words. For example, to recognize the phrase *United States of America* as an entity requires chunking multiple words as a text unit. The next step is to determine if a chunk of text is actually a named entity. Furthermore, words or phrases that differ in case but are the same entity should be recognized as such. Also, different chunks such as Professor Doe and Doe should be identified as the same entity. This requires capturing nonlocal dependencies in the entire document or across the corpus. Many NER systems require external knowledge in suitably codified form and *name lexicons*. Algorithms such as the *Viterbi* and *beam search* are used to chunk text. Maximum Entropy and HMMs are used to determine if a chunk of text is actually a named entity.

2.4.1 Approaches to NER

The three major approaches to NER are based on lexicon, rules, and machine learning. However, a NER system may combine multiple approaches (Keretna et al., 2014). *Lexicon-based* approaches utilize a lexicon or gazette constructed from external knowledge sources to match chunks of the text with entity names. They also provide a nonlocal model for resolving multiple names matching the same entity—*entity resolution*. Some preprocessing steps such as stemming, rewriting, and replacing a word with its synonyms increase entity matching rate. Lexicon-based approaches achieve better results for specific domains. However, they cannot identify new entities that are not in the lexicon.

Rule-based systems construct rules manually or automatically and use them for entity detection. Heuristics derived from the morphology or semantics of input sequence in conjunction with pattern matching are utilized in rule construction. Systems based on manually constructed rules makes it difficult to apply them to new domains.

Machine learning approaches use probability models and features that are derived from the input text. There are four subclasses: HMMs, CRFs, Support

Vector Machines (SVM), and Maximum Entropy models. These approaches require large annotated training data to estimate model parameters. Also, they do not naturally provide a nonlocal model for *entity resolution*.

2.4.2 Evaluating NER Systems

The CoNLL-2003 shared task named entity dataset is a benchmark for evaluating NER systems. It consists of eight files covering two languages: English and German. For each language, three data files are provided: training, development, and test. NER systems learn their parameters using the *training data*, parameters are tuned using the *development data*, and the systems are tested using the *test data*. NER systems performance is measured using an F-score with $\beta = 1$:

$$F_{\beta=1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

where precision is the ratio of correct results to results returned and recall is the ratio of correct results to the number of results that should have been returned. Sixteen systems participated in the CoNLL-2003 shared task competition. For the English test, the best $F_{\beta=1} = 88.76 \pm 0.7$ and for the German test 72.41 ± 1.3.

2.4.3 Big Data and NER

Coupled expectation maximization (CoEM) algorithm for NER task entails a graph structure of 2 million vertices and 20 million edges (Jones, 2005). When implemented on a Hadoop cluster with 95 cores, the algorithm took over 7.5 h.

GraphLab is a graph-based, parallel programming, distributed computation framework written in C++ (Low et al., 2013). It targets sparse iterative graph algorithms. Though originally developed for machine learning tasks, several implemented libraries of algorithms area available for tasks such as clustering, collaborative filtering, computer vision, topic modeling, graphical models, and graph analytics.

The CoEM algorithm when implemented using GraphLab framework with 16 cores executed in less than 30 min. When executed on cloud-hosted GraphLab with 32 Amazon EC2 machines, the algorithm executed in 80 s. This example illustrates the role of high performance computing and selection of suitable computational framework in solving large NLP problems.

2.5 Parsing

Parsing is the process of deriving a syntactic structure for a sequence of words in the language (Chomsky, 2002). The structure depicts syntactic relationships that hold between the words in phrases. A *parse tree* is one such representation.

Parsing is a fundamental preprocessing step for higher level NLP tasks such as NER, IE, and MT.

Current formalisms for syntactic structures include context-free grammars (CFGs), categorical grammars, head-driven phrase structure grammars, lexical functional grammars, minimalist syntax, and tree adjoining grammars. We focus on CFGs and its variants: probabilistic CFGs (PCFG), and lexicalized probabilistic CFGs (LPCFG).

2.5.1 CFGs and PCFGs

A CFG G is 4-tuple: $(\sum, N, R, S)$, where $\sum$ is a set of terminal symbols; N is a set of nonterminal symbols; R is a set of rules of the form $X \rightarrow Y_1 Y_2 \cdots Y_n$, for $n \geq 0, X \in N, Y_i \in (\sum \cup N)$; and $S \in N$ is a special nonterminal called the *start symbol*. For example, $\sum$ = {unique words in the language} and VP $\rightarrow$ Vt NP is a rule. The rule says that a verb phrase (VP) can be replaced by a transitive verb (Vt) followed by a noun phrase (NP).

In a CFG, we may have more than one rule that have the same list of nonterminals on the left hand side, but differ in right hand side. For example, (VP $\rightarrow$ Vt NP) and (VP $\rightarrow$ VP NP) are two such rules. A VP can be replaced by the right hand side of the first rule or the second. In probabilistic CGFs, a value is associated with each rule. For example, for the two rules above, probability values might be 0.6 and 0.4. These values help to break ties, should they arise. A rule with higher probability is preferred over the rules with lower probabilities.

The probability of a parse tree t which uses rules $\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n$, denoted $p(t)$, is defined as:

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \rightarrow \beta_i) \tag{8}$$

where $q(\alpha_i \rightarrow \beta_i)$ is the probability of the rule $\alpha_i \rightarrow \beta_i$. Each instance of a rule is included in the computation.

A CFG is ambiguous if more than one parse tree can be constructed for a sentence. Ambiguity is problematic since each parse tree denotes a different interpretation of the sentence. Unlike the case of programming languages, CFGs for natural languages are typically ambiguous.

Consider a sentence s and let $\mathcal{T}(s)$ denote all possible parse trees of s. The number of parse trees grows exponentially with the length of the sentence. CKY, a dynamic programming algorithm, is used to generate all possible parse trees. The probability assigned by a PCFG for each $t \in \mathcal{T}(s)$ can be used to rank the parse trees. The most likely parse tree for s is:

$$\underset{t \in \mathcal{T}(s)}{\operatorname{argmax}}\, p(t)$$

A PCFG is often generated from an annotated corpus. Penn WSJ Treebank is one of the earliest ones in this category. It features 50,000 sentences along with

their manually generated parse trees. The underlying PCFG is all the rules seen in the corpus. The probability of a rule, $\alpha \rightarrow \beta$ is computed using maximum likelihood estimates:

$$q_{\text{ml}}(\alpha \rightarrow \beta) = \frac{\text{count}(\alpha \rightarrow \beta)}{\text{count}(\alpha)}$$

where $\text{count}(\alpha \rightarrow \beta)$ is the number of times the rule $\alpha \rightarrow \beta$ occurs, and $\text{count}(\alpha)$ is the number of times α occurs in some rule.

2.5.2 Lexicalized Probabilistic CFGs

Major weaknesses of PCFGs include lack of sensitivity to lexical information, and structural frequencies. LPCFG overcome these limitations. LPCFGs require that the grammar be specified in Chomsky Normal Form (CNF). Any CFG can be rewritten in CNF. In CNF, the left hand side of every rule is restricted to only one nonterminal. Assuming that the language does not contain the empty string ϵ, the right hand side may contain either two nonterminals or one terminal.

LPCFGs add annotations to the left hand side of each rule. This annotation is the lexical information of the right hand side carried over to the head (i.e., the left hand side) of the rule. This is a core idea in natural language syntax and is specified by a set of rules. The lexical information carried over to the root of a parse tree through this process is called the *headword* of the parse tree.

A parameter such as $q(S \rightarrow NP\ VP)$ in PCFG turns into, for example, $q(S(dried) \rightarrow NP(river)\ VP(dried))$ in LPCFG. The words *river* and *dried* are lexical information. There are a large number of parameters to estimate. Each parameter is decomposed into a product of two parameters, and parameters are computed and smoothed using training data.

2.5.3 Evaluating Parsing Algorithms

Performance of parsing algorithms is evaluated using *precision* and *recall* measures on test data. Parse trees are represented as a collection of *constituents* to facilitate evaluation. A constituent refers to a subtree in the parse tree and corresponds to a sequence of words in the sentence being parsed. The label assigned to a constituent is the label of the root of the subtree. A constituent will also have two other pieces of information: the *start* and *end* word numbers of the subsequence of the sentence represented by the constituent.

For each sentence in the test data, a manually parsed tree is available. The latter is called the *gold standard*. The constituents in the parse tree generated by the algorithm are compared with the constituents in the gold standard. Based on the degree of conformance of constituents, precision and recall measures are calculated. Parsers based on LPCFGs perform significantly better than their PCFG counterparts.

2.5.4 Big Data and Parsing

Deep parsing of natural language text holds key to harnessing unstructured Big Data. For example, dependency tree (DT) is a representation produced via deep parsing. A DT is an acyclic graph and depicts dependencies between lexical entities (words or morphemes). When two words are connected by a dependency relation, one of the words is the *head* and the other is the *dependent*. A *dependency link* is an arrow pointing from the head to the dependent. Usually, the dependent is the modifier, and the head plays a larger role in determining the behavior of the pair in the text. There is growing body of work on creating new tree-banks for training dependency parsers for different languages.

3 NLP APPLICATIONS

Shown in Table 2 are some NLP applications, and a subset of them are discussed below.

3.1 Machine Translation

The goal of machine translation (MT) is to translate speech or text from one language to another. Lexical and syntactic ambiguities are the two primary challenges in MT. A word in the *source language*, depending on its POS tag or word sense, will translate to different words in the *target language*. Another challenge comes from differing word orders in the source and target languages. For example, English word order is subject—verb—object, whereas in Japanese it is subject—object—verb.

3.1.1 Statistical Models

Most of the current approaches to MT employ statistical models. They view MT as a supervised learning problem and use a training set of translation examples from a parallel corpus. For example, IBM's early work in this area used a parallel corpus of French–English translation of Canadian parliament proceedings. The parallel corpus comprised 1.7 million sentences of length 30 words or less.

The *noisy channel model* for MT is based on statistical machine learning. Assume that s and t are sentences in the source and target languages. The goal is to generate t given s. The model has two components: $p(t)$—the language model, and $p(s\,|\,t)$—the translation model. The reverse translation model is defined as:

$$p(t\,|\,s) = \frac{p(t,s)}{p(s)} = \frac{p(t)\ p(s\,|\,t)}{\sum_t p(t)\ p(s\,|\,t)}$$

and the best target sentence is that t computed as:

$$\underset{t}{\text{argmax}}\, p(t\,|\,s) = \underset{t}{\text{argmax}}\, p(t)p(s\,|\,t)$$

TABLE 2 Representative Natural Language Processing Applications

NLP Application Name	Application Description
Machine Translation	Automatic translation of speech or text from one language to another.
Information Extraction	Extract structured information from unstructured data so that queries on the unstructured data can be precisely answered.
Topic Modeling	Discover main themes that are present in unstructured document collections and organize collections according to these themes.
Text Summarization	Provide a concise summary of one or more text documents.
Document Clustering	Non-overlapping partitioning of a set of documents into a number of categories.
Document Classification	Determine the class to which a given document belongs to. Usually, the set of classes are predefined.
Question Answering Systems	Provide concise and relevant information to user queries by extracting it from structured and unstructured sources.
Dialog Systems	Coherently converse with humans using multimodal communication—text, speech, gesture, graphics, and haptics.
Natural Language User Interfaces	Enable communicating with computing and other electro-mechanical devices in natural language.
Email Spam Detection	Differentiate legitimate email from unsolicited bulk email.
Recommender Systems	Information filtering systems that produce a list of recommendations through content and collaborative filtering.
Educational Applications	A broad range of applications that provide services such as personalized eLearning, automated question generation and assessment.

The language model can be a trigram model and parallel corpus is not needed to estimate its parameters. The translation model parameters are estimated by employing the parallel corpus sentence pairs (s, t) using the notion of *alignments*. When alignments data is not available in the parallel corpus, the Expectation Maximization (EM) algorithm is used to automatically compute it. This approach to MT is also called the *IBM models* due to the pioneering work of IBM with the noisy channel model.

Phrase-based translation approaches make direct use of alignments of the noisy channel model. Automatically learning a phrase-based lexicon from

parallel corpus (aka *bitext*) is central to this model. Phrase-based translation models assess the quality of a translation based on a score derived from the language, phrase, and distortion models. The last model imposes a penalty for not translating the phrases in the source sentence in sequential order.

Manual evaluation of MT systems is extensive but expensive. In Papineni et al. (2002), a method for automatically evaluating MT systems is proposed. The method is fast, inexpensive, and language independent. It also correlates highly with human evaluation.

3.1.2 Big Data and MT

Given the astronomical growth in the volume of text data coupled with global commerce and geopolitics, there is a greater need for multilingual, real-time MT than ever before. Human translators are not only expensive, but also cannot keep up with exploding data volumes. SDL, a company specializing in customer engagement, provides a cloud-based MT system called BeGlobal. The latter enables real-time translation of social media communications, Web content and other text-based data. BeGlobal's capability includes over 80 language translation combinations. Its monthly translation volume exceeds billions of words from structured and unstructured sources.

3.2 Information Extraction

The goal of information extraction (IE) is to map unstructured text into a structured database so that queries can be precisely answered using the latter. IE enables new ways of querying, organizing and analyzing unstructured data by leveraging the well-defined semantics of structured databases (Doan et al., 2006, 2009). For example, IE enables posing precise queries about job openings by extracting structured information from newspaper classified ads.

3.2.1 Approaches

Approaches to IE differ in (a) the type of structure extracted (e.g., entities, relationships, attributes), (b) type of unstructured sources (e.g., short strings or documents, templatized or open-ended), (c) type of training data available (e.g., labeled unstructured data, linguistic tags), (d) methods used for extraction (rule based or statistical), and (e) type of output extracted (e.g., structured database or annotated unstructured text).

The early IE systems were rule based and the rules are manually coded by domain experts. Since this is a tedious activity, algorithms for automatically learning the rules from labeled examples of entities in unstructured text were introduced. Rules-driven IE systems do not perform well with noisy unstructured data sources.

Limitations of rules-driven IE systems led to the emergence of two classes of statistical methods: generative models based on HMMs, and conditional models

based on maximum entropy. Currently, global conditional models (aka CRFs) have gained prominence. Also, techniques from grammar construction were used to address the needs of some IE applications that require elaborate analysis of document structure.

Given the diversity of IE applications, both rule based and statistical methods continue to enjoy popular use. Hybrid models which combine rule based and statistical methods are also increasingly used.

3.2.2 Performance

The IE problem has been extensively studied for about three decades. Numerous IE systems are deployed both in industrial and academic settings. Accuracy continues to be a prime concern (Sarawagi, 2008). State-of-the-art extraction models achieve over 90% accuracy for entity extraction. However, for relationship extraction, accuracy hovers around 75% even in restricted domains such as news articles. Research on more advanced extractions such as long entities, soft attributes of entities, and higher order structures is needed to advance IE to the next level.

3.2.3 Big Data and IE

CiteSeer$^{\mathcal{X}}$ is a digital library which features over 3.5 million scholarly documents and receives between 2 and 4 million requests per day (Williams et al., 2014). Scholarly documents are gathered from the Web by focused crawling and they through a series of automatic processing steps including IE, data integration and entity linking. CiteSeer$^{\mathcal{X}}$ exemplifies issues and solutions to large scale IE, clustering, and name disambiguation.

3.3 Topic Modeling

PubMed is a free search engine for querying primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics (NCBI, 2015). It stores over 24 million citations for biomedical literature from MEDLINE, life science journals, and online books. Likewise, ACL Anthology is a digital archive of over 34,000 research papers in computational linguistics and NLP (ACL, 2015). Even with advanced search tools it is often difficult to find and discover what one is looking for. New computational tools are needed to help organize, search, visualize, and understand these unstructured document repositories. *Topics models* are tools just for this purpose.

Topic models are algorithms for discovering main themes that are present in a document collection and to organize the collection according to these themes (Blei, 2012). Topic models have also been used to analyze other types of data such as streaming collections, social networks, genetic data, and images.

3.4 Probabilistic Topic Modeling

Topic modeling algorithms are statistical methods that analyze the words in a set of documents to discover themes, relationships between themes, and how the themes have evolved over time. *Probabilistic topic modeling* refers to a suite of algorithms from the machine learning domain whose goal is to discover and annotate large document collections with thematic information. These algorithms do not require any prior annotations to the documents and the themes are discovered by analyzing the documents.

Latent Dirichlet allocation (LDA) is the simplest topic model (Blei et al., 2003). One assumption that LDA makes is *bag of words*—the order of words in a document does not matter. Some recent topic models relax this assumption by assuming that the topics generate words conditioned on the previous word. Another assumption that topic models make is that the order of documents in the collection does not matter. However, topics change over time and dynamic topic models relax this assumption. A third assumption that the number of topics is known and fixed is relaxed by Bayesian parametric topic models. Finally, other extensions of LDA relax additional assumptions.

3.5 Big Data and Topic Modeling

Some of the largest LDA models reported in literature have up to 10^3 topics. They are too small in scale to be of use in large-scale applications such as online advertising systems. Wang et al. (2014) report an LDA system which learns from Big Data and generates topics in the range 10^5. This LDA has been used in industrial search engines and online advertising systems that serve hundreds of millions of users.

3.6 Text Summarization

Automatic text summarization is the process of producing a summary of one or more text documents. The summary should retain the most important points of the original text document. A good text summarizer should also take into account variables such as length, writing style and syntax of the original document.

3.6.1 Approaches

Current approaches to automatic summarization fall into two broad categories: *extraction* and *abstraction*. Extractive methods work by selecting a subset of existing words, phrases, or sentences in the original text to form the summary. In contrast, abstractive methods first build an internal semantic representation and then use natural language generation techniques to create a summary. Such a summary might contain words not explicitly present in the original document.

Key challenges in text summarization include topic identification, interpretation, summary generation, and evaluation of the generated summary. Most practical text summarization systems are based on some form of extractive summarization. Abstraction based summarization is inherently more difficult and is an active area of research.

All extraction based summarizers, irrespective of the differences in approaches, perform the following three relatively independent tasks (Nenkova and McKeown, 2011, 2012): (a) capturing key aspects of text and storing it using an intermediate representation, (b) scoring sentences in the text based on that representation, (c) and composing a summary by selecting several sentences.

Open Text Summarizer (OTS) is an open source tool for summarizing texts (Rotem, 2014). It supports over 25 languages, and language-specific information is specified using XML rule files. OTS is available as a library for integration with NLP applications. It is included with many Linux distributions and can be used as a command line tool.

3.6.2 Big Data and Text Summarization

Every day over 150 billion messages are being sent from about 3.5 billion email accounts worldwide. It is quite easy to overlook important email messages as we strive hard to keep up with ever increasing surge. Both one-line summaries and short summaries for long emails will help to deal with this problem. This is not a trivial problem given the diversity in the language skills of email authors and linguistic genre of email contents. Here again Big Data comes to the rescue by providing training data for data-driven machine learning algorithms.

3.7 Document Clustering and Classification

Clustering, in the general sense, is the nonoverlapping partitioning of a set of objects into classes. Text can be clustered at various levels of granularity by considering cluster objects as documents, paragraphs, sentences, or phrases. Clustering algorithms use both supervised and unsupervised learning methods. Document clustering has several applications including collection browsing, corpus summarization, and document classification.

Hierarchical organization of documents into coherent classes facilitates exploratory browsing of documents in a collection. Corpus summarization is provided in the form of cluster digests or word clusters. The latter provides insights into the overall content of the document collection. Clustering at the sentence level is useful for the document summarization task.

3.7.1 Clustering Algorithms

Clustering algorithm categories include agglomerative clustering, partitioning, and parametric based ones such as the EM algorithm (Agarwal and Mittal, 2014).

These algorithms operate on features and differ in terms of the types of features extracted, transformations applied to the features for dimensionality reduction, and the way the features are used in the algorithms.

Dealing with noisy documents is a major issue for document clustering (Agarwal et al., 2007). Social media data has created the need for clustering for noisy and streaming data. Also, with the ubiquity of multimedia data, text clustering need to be applied in the context of heterogeneous data.

3.7.2 Classification Algorithms

Clustering and classification are closely related problems. Clustering usually involves unsupervised learning, whereas classification is implemented using supervised learning methods. In classification, typically there are a predefined set of classes and the task is to determine the class to which a new instance belongs to. The classifier is trained using a set of labeled examples. The label indicates the class to which a given instance in the training set belongs to.

Commonly used text classification methods include decision trees, rule-based classifiers, SVM, and Bayesian classifiers. Text classification has important applications such as email classification and spam filtering; news filtering, organization, and personalized recommendations; document organization and retrieval; and sentiment analysis. Effectiveness of these data-driven machine learning approaches depend on the amount of training data.

3.7.3 Big Data and Clustering

Clustering has important applications for Big Data. For example, creating a keyword taxonomy for valuable English keywords, clustering millions of documents in collections such as the ACLWeb, and clustering of Web pages.

3.8 QA and Dialog Systems

The goal of automated QA systems is to deliver concise information that contains answers to user questions. Many organizations provide QA Web applications which feature large repositories of valuable knowledge to help customers to troubleshoot, for example, malfunctioning mechanical and electronic devices. QA systems provide personalized answers and are significantly advanced compared to static *Frequently Asked Question* lists.

Recently there has been a shift towards treating QA as a community-driven knowledge creation process and creating an end product that provides enduring value to a broad audience. Stack Overflow (Anderson et al., 2012), Yahoo! Answers, and Baidu Zhidao are such examples. Many QA sites employ voting and reputation mechanisms to help users identify the trustworthiness and accuracy of the answers.

3.8.1 Dialog Systems

The goal of a dialog system (aka conversational agent) is to coherently converse with a human. Dialog systems may employ text, speech, graphics, haptics, gestures and other modes for communication on both the input and output channels.

Components of a dialog system include (a) input conversion (using automatic speech, gesture, and handwriting recognizers) to plain text, (b) analyzing the text for identifying proper names, tagging parts of speech, and performing syntactic and semantic parsing, (c) analyzing the semantic information to construct a response, (d) constructing the response (using natural language, and gesture generators, and a layout engine), and (e) rendering the output (using text-to-speech engine, talking head, robot, or avatar).

3.8.2 Big Data and QA

QA systems will play an ever more increasing role in delivering factual answers to business questions by synthesizing answering using structured, semi-structured, and unstructured data from proprietary databases, big data platforms, and social media. This capability is in infancy stage and its potential to harness big data is promising.

3.9 Natural Language User Interfaces

As the name implies, Natural Language User Interfaces (NLUI) are a means to communicate with computer systems using natural languages. They are preferred for their interaction speed and ease of use. However, they pose several challenges for practical use.

Interest in NLUI dates back to 1960s. In the 1980s, NLUI focus shifted to database systems. Database applications continue to drive the NLUI research. In Li et al. (2005), a generic NLUI for XML databases is developed. Given an arbitrary English language sentence as query input, it is translated into an XQuery expression which can be evaluated against an XML database. This work is extended in Li et al. (2007) for developing a domain-adaptive NLUI. The same work is further extended to relational database querying (Li and Jagadish, 2014).

An application framework to enable third-party developers to add spoken NLUI to standalone mobile applications is developed in Han et al. (2013). An integrated NLUI and GUI for mobile devices is described in Hodjat et al. (2006).

3.9.1 Role of Big Data

Systems such as Apple Siri, IBM Watson, Google Now, and Microsoft Cortana provide spoken language interfaces. Though these interfaces are not perfect, they

allow the user to specify commands in a free format. This is possible due to semantic analysis of user commands, rather than syntactic pattern matching at a superficial level. Big Data plays a significant role in training these systems given the diversity of end users.

3.10 Software Tools and Frameworks

A range of high quality open source tools and frameworks are available for NLP research and applications. Some tools focus on specific tasks such as tokenization, POS tagging, and parsing. Others are available in the form of algorithmic libraries such as *Lucene* for full-featured text search (The Apache Software Foundation, 2014), *OTS* for text summarization (Rotem, 2014), *tm* for text mining applications within R (Feinerer et al., 2008), *WEKA* for machine learning (The University of Waikato, 2014), and *WordNet*—a lexical database of English (Princeton University, 2015).

End-to-end infrastructure and frameworks for developing complete NLP applications include *GATE* (Cunningham et al., 2002; The University of Sheffield, 2015), *Open NLP* (The Apache Software Foundation, 2012), *NLTK* (Bird et al., 2009; Perkins, 2010), *MALLET* (University of Massachusetts at Amherst, 2014), RapidMiner (Hofmann and Klinkenberg, 2013), *Stanford CoreNLP* (Stanford NLP Group, 2014), and *TectoMT* (Popel and Žabokrtský, 2010). Apache Unstructured Information Management Architecture is another framework (OASIS Technical Committee, 2015) suitable for large scale data. It is implemented in Java and is also an OASIS standard. It enables NLP applications development as a pipeline of components. Each component implements interfaces defined by the framework. Functionality needed for composing workflows using the components and managing the resulting workflows is provided by the framework. Scaling is achieved by replicating processing pipelines over a cluster of networked computer nodes.

The distinction between libraries and frameworks is rather fluid as tools are evolving rapidly. Developing NLP applications using some of the above resources is illustrated in Bird et al. (2009), Perkins (2010), Ingersoll et al. (2012), Mihalcea and Radev (2011), and Russell (2011).

4 DATA SOURCES FOR NLP RESEARCH

As seen earlier, many machine learning based approaches to NLP critically depend on data for training and evaluation. Datasets come in different forms and sizes. They range from raw text, speech corpora, to text and speech with manual or automatic annotations, words and their relationships (morphological, semantic, translational, evolutionary), grammars, and parameters of statistical models (e.g., grammar weights).

4.1 Brown and New York Times Corpora

Brown corpus is one of the earliest datasets which contains over one million English words. Reuters-21578 Text Categorization Collection dataset encompasses documents which appeared on the Reuters newswire in 1987 (Lewis, 2015). A New York Times corpus includes 1.8 million articles published between January 1987 and June 2007. Over 650,000 of these articles include a summary written by NYT library scientists.

4.2 Google Corpora

In 2006, Google released a *trillion-word corpus* with frequency counts for all sequences up to five words long. It is a million times larger than the Brown corpus (Brants and Franz, 2012; Norvig, 2009). However, since the data came from unfiltered Web pages, it has various kinds of errors including grammatical and spelling as well as incomplete sentences. Furthermore, the data is not annotated carefully with hand-corrected POS tags. Given its sheer size, this corpus could be used for building statistical models for language modeling, MT, and speech recognition. The Google 5M LID dataset contains 5 million utterances collected from different Google Services for training and testing systems for automatic language identification (Gonzalez-Dominguez et al., 2015).

In 2013, Google released a large dataset of *syntactic ngrams* derived from English Google Books (Goldberg and Orwant, 2013). Syntactic ngrams are structures in which the contexts of words are based on their positions in a syntactic parse tree, and not their sequential order in the sentence. Therefore, words that are far apart in the sentence can be close to each other syntactically. The dataset contains over 10 billion unique items covering a wide range of syntactic structures, and includes a temporal dimension as well (Google, 2014b). A large benchmark dataset containing *one billion words* for evaluating language modeling techniques is described in Chelba et al. (2013).

4.3 Linguistic Data Consortium

The Linguistic Data Consortium (LDC) is an open consortium of universities, libraries, corporations, and government research laboratories (Linguistic Data Consortium, 2015). LDC creates and distributes a wide array of language resources to promote and enable NLP research and applications. In October 2014, LDC released 8500 h of the United Nations Proceedings Speech dataset in six languages—Arabic, Chinese, English, French, Russian, and Spanish.

4.4 British National and Europarl Parallel Corpora

The British National Corpus (BNC) is a million-word dataset distributed in XML format. It consists of 90% written and 10% orthographically transcribed spoken

text (British National Corpus, 2014; Leech et al., 1994). The data is annotated with POS tags and lemmatized.

The Europarl parallel corpus is extracted from the proceedings of the European Parliament (Koehn, 2015). It includes versions in 21 European languages. It also provides alignment data for training and testing statistical MT systems.

4.5 Other Corpora

The International Conference on Language Resources and Evaluation is organized by the European Language Resources Association (ELRA) biennially. This conference publishes papers about new datasets and metrics. Evaluations and Language resources Distribution Agency is ELRA's operational body which produces language resources to promote NLP research and applications.

The World Atlas of Language Structures (WALS) is a large database of structural—phonological, grammatical, lexical—properties of world languages. This data is gathered by language experts from descriptive materials such as reference grammars (Dryer and Haspelmath, 2013). Project Gutenberg is another source which offers over 46,000 free eBooks (Project Gutenberg, 2015).

An approach to large-scale speaker recognition using a dataset collected from the Google Tech Talk channel on YouTube is discussed in Schmidt et al. (2014). The dataset contains 1111 videos with 998 distinct speakers. Each video contains at least 30 min of speech.

Amazon Mechanical Turk provides an on-demand, inexpensive, scalable workforce for tasks that require human intelligence. This service can be used to annotate NLP test data or correct automatically annotated data.

5 BIG DATA DRIVEN NLP RESEARCH AND APPLICATIONS

As discussed in the previous sections, current NLP research and applications critically depend on large datasets for both training and evaluation. The availability of massive training data also simplifies models in some cases. For example, simple *n*-gram models or linear classifiers based on millions of specific features perform better than elaborate models that try to discover general rules (Halevy et al., 2009).

The key to benefiting from Web-scale data is to use the available large-scale data and derive annotations. For example, useful semantic relationships can be automatically learned from the statistics of search queries and the corresponding results (Riezler et al., 2008) or the accumulated evidence of Web-based text patterns and formatted tables (Talukdar et al., 2008).

5.1 Memoization

Based on the experiences with statistical methods in speech recognition and MT, *memoization* is a good practice if large training datasets are available (Halevy

et al., 2009). Earlier approaches relied on elaborate rules for the relationships between syntactic and semantic patterns in the source and target languages. In contrast, current statistical MT models employ large memorized phrase tables that give candidate mappings between specific source- and target-language phrases.

5.2 Leveraging Big Data

We briefly describe current approaches to NLP which leverage large datasets to significantly improve performance. Many NLP applications such as automatic speech recognition, MT, spelling correction depend on the quality of language models. The amount of training data is one of the factors that determines the performance of language models.

One billion words of training data (Chelba et al., 2013; Google, 2014a) and syntactic ngrams data (Goldberg and Orwant, 2013; Google, 2014b) can be used to improve the accuracy of core NLP tasks such as syntactic language modeling and syntactic parsing. This is significant given that, until now most successful approaches to language modeling and parsing depended on sequential data (e.g., *n*-grams) as large-scale datasets of syntactic counts were not readily available. The WALS (Dryer and Haspelmath, 2013) and the Europarl parallel corpus (Koehn, 2015) data can be used for developing multilingual NLP applications.

A unified neural network architecture and learning algorithms which can perform various NLP tasks such as POS tagging, chunking, NER, and semantic role labeling is proposed in Collobert et al. (2011). The system learns internal representations on the basis of large amounts of mostly unlabeled training data. The resulting system requires minimal computational resources and performs well.

The United Nations Proceedings Speech dataset is a valuable resource for speech recognition and language identification (Linguistic Data Consortium, 2015). In Jyothi et al. (2012), large-scale language models were developed and integrated with a large vocabulary continuous speech recognition system. The language models were trained on 87,000 h of speech (equivalent to 350 million words).

The problem of automatically identifying the language in a given utterance using deep neural networks (DNN) is investigated in Gonzalez-Dominguez et al. (2015) using the Google 5M LID dataset. This study reports 76% performance improvement relative to i-vector approach. This improvement is attributed to DNN and the Google 5M LID dataset.

5.3 Streaming Data

Language models that work well in streaming data environments are essential in NLP applications such as filtering and classifying news articles and social media data. Sequential language modeling based on *n*-grams data from streams

is studied in Yogatama et al. (2014). They use a probabilistic language model that captures temporal dynamics and conditions on arbitrary nonlinguistic context features.

5.4 Sparseness Problems

The use of Latent Words Language (LWL) model in reducing the sparseness problems of traditional *n*-gram language models is investigated in Deschacht et al. (2012). This study shows that the LWL model significantly outperforms interpolated Kneser–Ney smoothing and class-based language models on three different corpora.

5.5 Noisy Data

Noisy unstructured text is generated from on-line chats, SMS, email, newsgroups, blogs, and automatically transcribed text from speech (Agarwal et al., 2007). Larger datasets are effective in dealing with noisy text. A survey on types of text noise and techniques to handle noise is described in Subramaniam et al. (2009).

5.6 Selecting and Combining Features

Finding the right representations for words is critical when domain-specific labeled data for the task is limited. In Huang et al. (2014), novel techniques are proposed for extracting features from *n*-gram, Hidden Markov, and Partial Lattice Markov Random Field models. These features are used for tasks such as POS tagging and IE. Results of this study indicate that features derived from statistical language models in combination with more traditional features outperform approaches that use only traditional representations. Furthermore, graphical model representations outperform *n*-gram models, especially on sparse and polysemous words.

5.7 Computational Advertising

Computational advertising (CA) is an emerging discipline, which is at the intersection of large scale search, text analysis, information retrieval, statistical modeling, machine learning, classification, optimization, and microeconomics (Broder et al., 2010). A central issue in CA is retrieving relevant ads using a textual representation of the user context. Proposed solutions to this problem use the Web as a repository of query-specific knowledge and Web search results retrieved by the query as a form of relevance feedback and query expansion.

5.8 Multimedia Data

As multimedia data is growing at a phenomenal rate, the need for IE from images and video is greater than ever. A survey of approaches for this task are described in Jung et al. (2004). In piggy-back text retrieval, text surrogates based indexing of multimedia data is performed automatically Rüger (2009). This work describes techniques and common approaches to enabling multimedia search engines using the text surrogates.

5.9 IBM Watson

IBM Deep QA system, code named Watson, is a perfect example of Big Data-enabled solutions to NLP problems. Watson understands natural language, generates hypotheses based on evidence, and learns through user interactions and new experiences. Watson's defining moment was when it defeated world Jeopardy champions in February, 2011. IBM markets Watson as a commercial product with packaged solutions for vertical market segments such as healthcare, finance, retail, and public sector.

6 TRENDS AND FUTURE RESEARCH DIRECTIONS

Availability of massive scale training data for NLP research is ushering in a paradigm of experimentation with hybrid models and ensemble approaches. For example, Brown clusters in conjunction with log-linear models significantly enhance performance in tasks such as NER and parsing.

The current research in NLP is driven more by statistical machine learning approaches than linguistic theory (Johnson, 2009). Traditionally, grammar based approaches to parsing produced representations which closely reflected the meaning in the parsed text by capturing predicate-argument structure (aka semantic role labeling) and quantifier scope. It is possible to achieve the same with parsers based on statistical approaches as demonstrated in Palmer et al. (2005). Statistical approaches to NLP tasks that use large training data validate the view that simpler models often outperform more complex models.

Never-Ending Language Learner (NELL) project exemplifies how Big Data enables NLP research (Read the Web Project, 2015). The goal of this project is to construct a relational database that reflects the Web content through IE. NELL has been running continuously for over 4 years and acquired a collection of 70 million interconnected beliefs. It uses learned phrases, morphological features, and Web page structures to extract beliefs. NELL is also learning to reason over its extracted knowledge and automatically extend its ontology.

6.1 Even More NLP Big Data

Large-scale distributed Web 2.0 applications such as blog sites and other social media are making it easier to collect large training datasets quickly. According to Google, in 2013 there were 1.5 million Android activations per day. This phenomenal rise in the use of smart phones and other hand-held devices will contribute to even more data for NLP research. An approach to opinion mining using YouTube comments data is described in Severyn et al. (2014). Classifiers for predicting the opinion polarity and the type of comment are developed using tree kernel technology. The latter automatically extracts and learns features with better generalization power.

6.2 Better Models with More Data

Maximum Entropy (MaxEnt) language models are linear models which are typically regularized using the L1 or L2 terms in the likelihood objective. This obviates the need for smoothed *n*-gram language models. In Biadsy et al. (2014), the effect of adding backoff features and its variants to MaxEnt models is investigated. This approach results in better language models with lower perplexity even with training data in the order of hundreds of billions of words and hundreds of millions of features.

6.3 Spatial Knowledge

The task of text-to-3D scene generation is addressed in Chang et al. (2014). This study demonstrates that spatial knowledge is useful for interpreting natural language. Users provide input in natural language text from which explicit constraints on the objects that should appear in the scene are extracted. These constraints are augmented with learned spatial knowledge to infer missing objects and likely layouts for the objects in the scene.

6.4 NLP Applications

Turning to practical NLP applications, functional features of Siri, Google Now, and Cortana showcase the state-of-the-art in natural language interfaces. Siri is a virtual assistant available for Apple's iOS devices. It features voice-controlled natural language interface to help users perform personal tasks. Siri is an embodiment of several decades of research of the NLP community. It uses sequential inference and contextual awareness to understand and respond to user voice commands. Siri is also being integrated into Apple's HomeKit framework so it can be used to control home automation tasks such as controlling the operation of garage doors, lights, home appliances, and thermostats. Siri understands and speaks over 15 languages including various dialects of English

and Spanish. Google Now and Cortana are similar applications for Android and Windows Phone devices.

Google conducted a mobile voice study in October, 2014 to understand search habits of American smart phone users across different age groups. According to this study, 55% of the teenagers use hands-free search every day, and 76% of the respondents think voice search is great for multitasking. Voice search is used for tasks such as seeking driving directions, dictating text messages, making phone calls, and checking weather. This trend will increase in the future as voice-based interfaces become mainstream. Furthermore, availability of APIs for Siri and Google Now will accelerate the development of such apps.

7 CONCLUSIONS

In a traditional software development approach, domain rules are manually derived and coded into an application. Enhancements to the application typically warrant manually crafting additional rules. In contrast, data-driven approaches do not require changes to the application. The application simply needs to be retrained with data that reflects the current and new requirements. A closely related issue is the problem of *domain adaptation*. For example, parsers do not perform well if they are applied on text whose domain is different from that of the text used for training. The problem is resolved by retraining the parser with the text from the relevant domain (Foster et al., 2011). Retraining with the right data offers the key to application portability as well as enhancement without source code changes for algorithmically ill-posed problems.

There is a parallel to what is happening now in NLP area to what has happened in the field of Information Retrieval in the early 1990s. Using computers to search for relevant information was first introduced in 1945 by Vannevar Bush in an article titled *As We May Think*. This is the genesis of Information Retrieval and this area has attracted several researchers since then. However, it remained a specialized area of computer science research until it was brought to the fore with the emergence of the World Wide Web in early 1990s. Similar situation exists with NLP research and applications today.

The field of Computational Linguistics originated in 1950s with focus on MT. Since then the field has made significant advances in various areas of NLP. Data-driven approaches to NLP will further accelerate progress in this area. Most machine learning algorithms attempt to capture low frequency events which contribute data to most relevant analysis. This is where more data plays a critical role.

Natural language analysis is essentially a stochastic process. Therefore, it is easier to illustrate this process through examples than with rules. Big Data provides unprecedented opportunities to advance the field in ways that was not possible hitherto. Research into scalability, multilingual interfaces,

personalization, portability, and provenance can greatly benefit from the availability of extremely large datasets. Big Data-enabled NLP research has the potential to bring NLP applications to the masses and fundamentally transform the way users interact with computing devices. Conversely, NLP approaches that are based on deep semantic analyses will help realize substantial value from Big Data.

REFERENCES

ACL, 2015. ACL anthology: a digital archive of research papers in computational linguistics. http://www.aclweb.org/anthology/.

Agarwal, B., Mittal, N., 2014. Text classification using machine learning methods—a survey. In: Babu, B.V., Nagar, A., Deep, K., Pant, M., Bansal, J.C., Ray, K., Gupta, U. (Eds.), Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012, Advances in Intelligent Systems and Computing, vol. 236. Springer India, pp. 701–709.

Agarwal, S., Godbole, S., Punjani, D., Roy, S., 2007. How much noise is too much: a study in automatic text classification. In: Seventh IEEE International Conference on Data Mining, 2007 (ICDM 2007), pp. 3–12.

Anderson, A., Huttenlocher, D., Kleinberg, J., Leskovec, J., 2012. Discovering value from community activity on focused question answering sites: a case study of stack overflow. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, pp. 850–858.

Biadsy, F., Hall, K., Moreno, P., Roark, B., 2014. Backoff inspired features for maximum entropy language models. In: Proceedings of Interspeech.

Bird, S., Klein, E., Loper, E., 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media, Sebastopol, California.

Blei, D.M., 2012. Probabilistic topic models. Commun. ACM 55 (4), 77–84.

Blei, D., Ng, A., Jordan, M., 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.

Brants, T., Franz, A., 2012. Web 1T 5-gram version 1. http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2006T13, Linguistic Data Consortium.

Brill, E., 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput. Linguist. 21 (4), 543–565.

British National Corpus, 2014. http://www.natcorp.ox.ac.uk/.

Broder, A., Gabrilovich, E., Josifovski, V., 2010. Information retrieval challenges in computational advertising. In: Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, NY, pp. 908–908.

Chang, A.X., Savva, M., Manning, C.D., 2014. Learning spatial knowledge for text to 3D scene generation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., 2013. One billion word benchmark for measuring progress in statistical language modeling. CoRR. abs/1312.3005. URL http://arxiv.org/abs/1312.3005.

Chomsky, N., 2002. Syntactic Structures, second ed. De Gruyter Mouton, Boston, Massachusetts.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P., 2011. Natural language processing (almost) from scratch. J. Mach. Learn. Res. 12, 2493–2537.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., 2002. GATE: an architecture for development of robust HLT applications. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 168–175.

Deschacht, K., De Belder, J., Moens, M.-F., 2012. The Latent words language model. Comput. Speech Lang. 26 (5), 384–409.

Dhar, V., 2013. Data science and prediction. Commun. ACM 56 (12), 64–73.

Doan, A., Ramakrishnan, R., Vaithyanathan, S., 2006. Managing information extraction: state of the art and research directions. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06. ACM, Chicago, IL, pp. 799–800.

Doan, A., Naughton, J.F., Ramakrishnan, R., Baid, A., Chai, X., Chen, F., Chen, T., Chu, E., DeRose, P., Gao, B., Gokhale, C., Huang, J., Shen, W., Vuong, B.-Q., 2009. Information extraction challenges in managing unstructured data. SIGMOD Rec. 37 (4), 14–20.

Dryer, M.S., Haspelmath, M. (Eds.) 2013. WALS Online. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Feinerer, I., Hornik, K., Meyer, D., 2008. Text mining infrastructure in R. J. Stat. Softw. 25 (5), 1–54. http://www.jstatsoft.org/v25/i05.

Foster, J., Çetinoglu, Ö., Wagner, J., Le Roux, J., Hogan, S., Nivre, J., Hogan, D., Van Genabith, J., 2011. #hardtoparse: POS tagging and parsing the Twitterverse. In: AAAI 2011 Workshop On Analyzing Microtext, United States, pp. 20–25. https://hal.archives-ouvertes.fr/hal-00702445.

Francis, N., Kucera, H., 1964. The brown corpus. http://www.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html.

Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., Smith, N.A., 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers, vol. 2. Association for Computational Linguistics, pp. 42–47.

Goldberg, Y., Orwant, J., 2013. A dataset of syntactic-Ngrams over time from a very large corpus of english books. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity, Atlanta, Georgia, USA. ACL, pp. 241–247.

Gonzalez-Dominguez, J., Lopez-Moreno, I., Moreno, P.J., Gonzalez-Rodriguez, J., 2015. Frame by frame language identification in short utterances using deep neural networks. Neural Netw. 64, 49–58.

Google, 2014a. 1-Billion-word language modeling benchmark. https://code.google.com/p/1-billion-word-language-modeling-benchmark/.

Google, 2014b. Syntactic N-grams. http://storage.googleapis.com/books/syntactic-ngrams/index.html.

Halevy, A., Norvig, P., Pereira, F., 2009. The unreasonable effectiveness of data. IEEE Intell. Syst. 24 (2), 8–12.

Han, S., Philipose, M., Ju, Y.-C., 2013. NLify: lightweight spoken natural language interfaces via exhaustive paraphrasing. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland. ACM, New York, NY, pp. 429–438.

Hey, T., Tansley, S., Toll, K. (Eds.) 2009. The Fourth Paradigm: Data-Intensive Scientific Discovery, first ed. Microsoft Press, Redmond, Washington.

Hodjat, B., Hodjat, S., Treadgold, N., Jonsson, I.-M., 2006. CRUSE: a context reactive natural language mobile interface. In: Proceedings of the 2nd Annual International Workshop on Wireless Internet, Boston, Massachusetts, USA. ACM, New York, NY.

Hofmann, M., Klinkenberg, R., 2013. RapidMiner: Data Mining Use Cases and Business Analytics Applications. Chapman & Hall/CRC, Boca Raton, Florida. ISBN 1482205491, 9781482205497.

Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., Yates, A., 2014. Learning representations for weakly supervised natural language processing tasks. Comput. Linguist. 40 (1), 85–120.

Indurkhya, N., Damerau, F.J., 2010. Handbook of Natural Language Processing. Chapman & Hall/CRC, Boca Raton, Florida.

Ingersoll, G.S., Morton, T.S., Farris, A.L., 2012. Taming Text: How to Find, Organize, and Manipulate It. Manning Publications Co., Greenwich, Connecticut.

Johnson, M., 2009. How the statistical revolution changes (computational) linguistics. In: Proceedings of the EACL 2009 Workshop on the Interaction Between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous? Athens, Greece. ACL, Stroudsburg, PA, pp. 3–11.

Jones, R., 2005. Learning to extract entities from labeled and unlabeled text. http://www.cs.cmu.edu/rosie/thesis.html.

Jung, K., Kim, K.I., Jain, A.K., 2004. Text information extraction in images and video: a survey. Pattern Recogn. 37 (5), 977–997.

Jurafsky, D., Martin, J.H., 2009. Speech and Language Processing, second ed. Prentice Hall, Upper Saddle River, New Jersey.

Jyothi, P., Johnson, L., Chelba, C., Strope, B., 2012. Large-scale discriminative language model reranking for voice-search. In: Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT, Montreal, Canada, pp. 41–49.

Keretna, S., Lim, C.P., Creighton, D., 2014. A hybrid model for named entity recognition using unstructured medical text. In: 9th International Conference on System of Systems Engineering (SOSE), 2014, pp. 85–90.

Klein, D., 2005. The unsupervised learning of natural language structure. Ph.D. thesis, Stanford University.

Koehn, P., 2015. Europarl: a parallel corpus for statistical machine translation. http://www.statmt.org/europarl/.

Leech, G., Garside, R., Bryant, M., 1994. CLAWS4: the tagging of the british national corpus. In: Proceedings of the 15th Conference on Computational Linguistics—Volume 1, Kyoto, Japan. ACL, Stroudsburg, PA, pp. 622–628.

Lewis, D.D., 2015. Reuters-21578 text categorization collection data set. https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection.

Li, F., Jagadish, H.V., 2014. NaLIR: an interactive natural language interface for querying relational databases. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, Snowbird, Utah, USA. ACM, New York, NY, pp. 709–712.

Li, Y., Yang, H., Jagadish, H.V., 2005. NaLIX: an interactive natural language interface for querying XML. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland. ACM, New York, NY, pp. 900–902.

Li, Y., Chaudhuri, I., Yang, H., Singh, S., Jagadish, H.V., 2007. DaNaLIX: a domain-adaptive natural language interface for querying XML. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China. ACM, New York, NY, pp. 1165–1168.

Linguistic Data Consortium, 2015. Language resources. https://www.ldc.upenn.edu/language-resources.

Low, Y., Bickson, D., Gonzalez, J., Guestrin, C., Kyrola, A., Hellerstein, J.M., 2013. GraphLab: a new framework for parallel machine learning. http://www.select.cs.cmu.edu/publications/paperdir/uai2010-low-gonzalez-kyrola-bickson-guestrin-hellerstein.pdf.

Manning, C., Schutze, H., 1999. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, Massachusetts.

Marcus, M.P., Marcinkiewicz, M.A., Santorini, B., Cambridge, MA, USA, 1993. Building a large annotated corpus of english: the penn treebank. Comput. Linguist. 19 (2), 313–330.

Mihalcea, R., Radev, D., 2011. Graph-based Natural Language Processing and Information Retrieval. Cambridge University Press, New York, NY.

Nadeau, D., Sekine, S., 2007. A survey of named entity recognition and classification. Lingvist. Investig. 30 (1), 3–26.

Navigli, R., 2009. Word sense disambiguation: a survey. ACM Comput. Surv. 41 (2), 10:1–10:69.

NCBI, 2015. PubMed, national center for biotechnology information. http://www.ncbi.nlm.nih.gov/pubmed.

Nenkova, A., McKeown, K., 2011. Automatic summarization. Found. Trends Inform. Retr. 5 (2-3), 103–233.

Nenkova, A., McKeown, K., 2012. A survey of text summarization techniques. In: Aggarwal, C., Zhai, C. (Eds.), Mining Text Data. Springer, Heidelberg, pp. 43–76.

Norvig, P., 2009. Natural language corpus data. In: Segaran, T., Hammerbacher, J. (Eds.), Beautiful Data: The Stories Behind Elegant Data Solutions. O'Reilly Media, pp. 219–242.

OASIS Technical Committee, 2015. Apache UIMA project. http://uima.apache.org/.

Palmer, D.D., 1997. A trainable rule-based algorithm for word segmentation. In: Proceedings of the 35th Annual Meeting of the ACL, Madrid, Spain. Association for Computational Linguistics, Stroudsburg, PA, pp. 321–328.

Palmer, M., Gildea, D., Kingsbury, P., 2005. The proposition bank: an annotated corpus of semantic roles. Comput. Linguist. 31 (1), 71–106.

Papineni, K., Roukos, S., Ward, T., Zhu, W.J., 2002. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on ACL, Philadelphia, Pennsylvania, USA, ACL '02. ACL, Stroudsburg, PA, pp. 311–318.

Perkins, J., 2010. Python Text Processing with NLTK 2.0 Cookbook. Packt Publishing, Birmingham, United Kingdom.

Popel, M., Žabokrtský, Z., 2010. TectoMT: modular NLP framework. In: Proceedings of the 7th International Conference on Advances in Natural Language Processing, IceTAL'10. Springer-Verlag, Berlin, pp. 293–304.

Princeton University, 2015. WordNet: a large lexical database for English. http://wordnet.princeton.edu/wordnet/.

Project Gutenberg, 2015. https://www.gutenberg.org/.

Ratnaparkhi, A., 1996. A maximum entropy model for part-of-speech tagging. In: Brill, E., Church, K. (Eds.), Proceedings of the Empirical Methods in Natural Language Processing, Philadelphia, PA, USA, pp. 133–142.

Read the Web Project, 2015. NELL: never-ending language learning. http://rtw.ml.cmu.edu/rtw/.

Riezler, S., Liu, Y., Vasserman, A., 2008. Translating queries into snippets for improved query expansion. In: Proceedings of the 22nd International Conference on Computational Linguistics—Volume 1, Manchester, United Kingdom, COLING '08. ACL, Stroudsburg, PA, pp. 737–744.

Rotem, N., 2014. Open text summarizer. http://libots.sourceforge.net/.

Rüger, S., 2009. Multimedia information retrieval. Synth. Lect. Inf. Concepts Retr. Serv. 1 (1), 1–171.

Russell, M.A., 2011. Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites. O'Reilly Media, Sebastopol, California.

Sarawagi, S., 2008. Information extraction. Found. Trends Databases 1 (3), 261–377.

Schmidt, L., Sharifi, M., Moreno, I.L., 2014. Large-scale speaker identification. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, pp. 1650–1654.

Severyn, A., Moschitti, A., Uryupina, O., Plank, B., Filippova, K., 2014. Opinion mining on YouTube. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Baltimore, MD, pp. 1252–1261.

Sproat, R., Shih, C., Gale, W., Chang, N., 1994. A stochastic finite-state word-segmentation algorithm for Chinese. In: Proceedings of the 32nd Annual Meeting of ACL, Las Cruces, New Mexico. Association for Computational Linguistics, Stroudsburg, PA, pp. 66–73.

Stanford NLP Group, 2014. Stanford NLP tools. http://nlp.stanford.edu/software/index.shtml.

Subramaniam, L.V., Roy, S., Faruquie, T.A., Negi, S., 2009. A survey of types of text noise and techniques to handle noisy text. In: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data, AND '09. ACM, New York, NY, pp. 115–122.

Sutton, C., McCallum, A., 2012. An introduction to conditional random fields. Found. Trends Mach. Learn. 4 (4), 267–373. ISSN 1935-8237.

Talukdar, P.P., Jacob, M., Mehmood, M.S., Crammer, K., Ives, Z.G., Pereira, F., Guha, S., 2008. Learning to create data-integrating queries. Proc. VLDB Endow. 1 (1), 785–796.

The Apache Software Foundation, 2012. openNLP. http://opennlp.apache.org/.

The Apache Software Foundation, 2014. Apache lucene core. http://lucene.apache.org/core/.

The University of Sheffield, 2015. GATE. http://gate.ac.uk/.

The University of Waikato, 2014. Weka 3: data mining software in java. http://www.cs.waikato.ac.nz/ml/weka/.

University of Massachusetts at Amherst, 2014. Machine learning for language toolkit. http://mallet.cs.umass.edu/.

Wang, K., Thrasher, C., Hsu, B.-J.P., 2011. Web scale NLP: a case study on URL word breaking. In: Proceedings of the 20th International Conference on World Wide Web, WWW '11. ACM, New York, NY, pp. 357–366.

Wang, Y., Zhao, X., Sun, Z., Yan, H., Wang, L., Jin, Z., Wang, L., Gao, Y., Law, C., Zeng, J., 2014. Peacock: learning long-tail topic features for industrial applications. ACM Trans. Intell. Syst. Technol. 9 (4), 39.1–39.23.

Wigner, E., 1960. The unreasonable effectiveness of mathematics in the natural sciences. Commun. Pure Appl. Math. 13 (1), 1–14.

Williams, K., Wu, J., Choudhury, S., Khabsa, M., Giles, C., 2014. Scholarly big data information extraction and integration in the CiteSeer$^{\chi}$ digital library. In: 2014 IEEE 30th International Conference on Data Engineering Workshops (ICDEW), pp. 68–73.

Yogatama, D., Wang, C., Routledge, B., Smith, N., Xing, E., 2014. Dynamic language models for streaming text. Trans. ACL 2, 181–192.

YouTube, 2015. Viewership statistics. http://www.youtube.com/yt/press/statistics.html.

Zhang, Y., Clark, S., 2011. Syntactic processing using the generalized perceptron and beam search. Comput. Linguist. 37 (1), 105–151.

Chapter 10

Analyzing Big Spatial and Big Spatiotemporal Data: A Case Study of Methods and Applications

Varun Chandola[*,1], **Ranga Raju Vatsavai**[†,1], **Devashish Kumar**[‡,1], **Auroop Ganguly**[‡,1]

[*] *Computer Science and Engineering, State University of New York at Buffalo, Buffalo, New York, USA*

[†] *Computer Science, North Carolina State University, Raleigh, North Carolina, USA*

[‡] *Civil and Environmental Engineering, Northeastern University, Boston, Massachusetts, USA*

[1] *Corresponding authors: e-mail: chandola@buffalo.edu; vatsavairr@ornl.gov; kumar.d@husky.neu.edu; a.ganguly@neu.edu*

ABSTRACT

Spatial and spatiotemporal data mining is the process of discovering interesting and previously unknown, but potentially useful patterns from the data collected over time and space. However, explosive growth in the spatial and spatiotemporal data, and the emergence of social media and location sensing technologies, emphasizes the need for developing new and computationally efficient methods tailored for analyzing big data. In this chapter, we study approaches to handle big spatial and spatiotemporal data by closely looking at the computational and I/O requirements of several analysis algorithms for such data. We also study applications of such methods in domains where data is encountered at a massive scale.

Keywords: spatial, Spatiotemporal data mining, Big spatial data, Biomass monitoring, Climate analysis, Object recognition

1 INTRODUCTION

We are living in the era of "Big Data." Spatial and spatiotemporal data, whether captured through cameras, remote sensors (e.g., remote sensing imagery, atmospheric radiation measurement (ARM) data), or produced using large-scale simulations (e.g., climate data) has always been "Big." Table 1 shows some of

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00010-1

TABLE 1 Earth Science Grid Integrated Data Archive

	CMIP5	ARM	DACC
Sponsor	SciDAC	DOE/BER	NASA
Description	40+ climate simulations	Atmospheric processes and cloud dynamics	Biogeochemical dynamics, FLUXNET
Archive size	~ 6 PB	~ 200 TB	~ 1 TB
Year started	2010	1991	1993

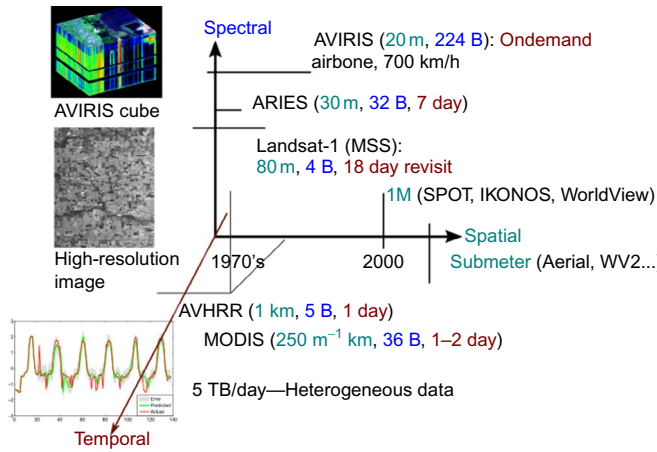


FIGURE 1 Advances in remote sensing data products (1970s through present).

the climate and earth systems data stored at the Earth System Grid[1] (ESG) portal. On the observation side, the Sloan Digital Sky Survey (SDSS)[2] archival center contains a catalog of about 300 million objects and spectral data for more than 1 million of these objects has already been archived (Adelman-McCarthy et al., 2008). Remote sensing imagery data archived at the NASA EOSDIS exceeds 3 PB. NASA generates about 5 TB of data per day. Figure 1 shows progression of remote sensing instruments along three important sensor characteristics: spatial, spectral, and temporal resolutions. Though these improvements are leading to increase in volume, velocity, and variety of remote sensing data products and making it hard to manage and process, they are also enabling new applications. For example, improvements in temporal resolution allows monitoring biomass on a daily basis. Improvements in spatial resolution allows fine-grained classification (settlement types), damage assessments, and critical infrastructure (e.g., nuclear proliferation) monitoring.

1. https://www.earthsystemgrid.org/.
2. The Sloan Digital Sky Survey, http://www.sdss.org/.

The rate at which spatiotemporal data is being generated clearly exceeds our ability to organize and analyze them to extract patterns critical for understanding dynamically changing world. Therefore, we need focused research on developing efficient management and analytical infrastructure for big spatial data. What has been observed in the past is that analysis of big spatial and spatiotemporal data poses a unique set of challenges arising from the scale as well as the need to model the dependencies in space and time. In this chapter, we have selected four widely used methods that handle such data. These are *Spatial Autoregressive (SAR) Models*, *Markov Random Field (MRF) Classifiers*, *Gaussian Process Learning Models*, and *Mixture Models*. We study each of these methods in the context of scalability and discuss possible approaches to handle the big data challenges. While the four methods form only a small subset of the wide array of existing analysis methods, we believe that the discussion will guide readers to analyze other methods in a similar fashion. In addition, we also provide a detailed study of three applications of spatial and spatiotemporal methods in areas where the data is typically at a large scale. These application areas are: *Biomass Monitoring*, *Complex Object Recognition*, and *Climate Change Studies*, which form a representative sample of a wide variety of applications such as air traffic management (Ayhan et al., 2012), weather analysis (Jitkajornwanich et al., 2012), etc.

The rest of this chapter is organized as follows. Section 2 discuss some of the key algorithms and the computational and I/O challenges posed by big data. Section 3 discuss a set of applications that deal with big spatial and spatiotemporal data.

2 ALGORITHMS

Increasing spatial and temporal resolution requires that the data mining algorithms should take into account the spatial and temporal autocorrelation. Explicit modeling of spatial dependencies increase computational complexity. We now briefly look at the following widely used data mining primitives that explicitly model spatial dependencies: SAR model (Anselin, 1988; LeSage, 1997a,b; LeSage and Pace, 2001; Ma, 2004; Pace and Barry, 1997a,b), MRF model (Anguelov et al., 2005; Boykov et al., 1999; Chou et al., 1993; Geman and Geman, 1984; Jhung and Swain, 1996; Li, 2001; Solberg et al., 1996; Warrender and Augusteijn, 1999), Gaussian Processes (Rasmussen and Williams, 2006), and Mixture Models (McLachlan and Basford, 1988). More details about these techniques can be found in Vatsavai (2011) and references therein.

2.1 SAR Model

We now show how spatial dependencies are modeled in the framework of regression analysis. In spatial regression, the spatial dependencies of the error

term, or the dependent variable, are directly modeled in the regression equation (Anselin, 1988). If the dependent values y_i' are related to each other, i.e., $y_i = f(y_j)\ i \neq j$, then the regression equation can be modified as:

$$\mathbf{y} = \rho W\mathbf{y} + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{1}$$

Here, W is the neighborhood relationship contiguity matrix and ρ is a parameter that reflects the strength of spatial dependencies between the elements of the dependent variable. After the correction term $\rho W\mathbf{y}$ is introduced, the components of the residual error vector ϵ are then assumed to be generated from independent and identical standard normal distributions.

2.1.1 Big Data Challenges

SAR analysis requires dealing with the contiguity matrix W in (1), which grows quadratically with the size of the image, i.e., if the image has p rows and q columns, then the size of the matrix W will be $pq \times pq$. This makes SAR modeling a challenge when dealing with large images. In particular, the first step in exact SAR modeling is the computation of eigenvalues of W. Given that typical eigenvalue decomposition is $O(p^3q^3)$, scaling exact SAR models to large-scale images is infeasible. SAR model also requires $O(p^2q^2)$ memory to store the contiguity matrix W.

2.1.2 Handling Big Data Challenges

The estimates of ρ and $\boldsymbol{\beta}$ can be derived using maximum likelihood (ML) theory or Bayesian statistics. Bayesian approach using sampling-based Markov Chain Monte Carlo methods can be found in LeSage (1997a). As mentioned earlier, without any optimization, likelihood-based estimation would require $O(n^3)$ operations. Recently, Pace and Barry (1997a,b), LeSage and Pace (2001), and Celik et al. (2007) have proposed several efficient techniques to solve SAR. Many of these techniques have been studied and compared in Kazar et al. (2004).

2.2 MRF Classifiers

A set of random variables whose interdependency relationship is represented by a undirected graph (i.e., a symmetric neighborhood matrix) is called a Markov Random Field (Li, 2001). The Markov property specifies that a variable depends only on the neighbors and is independent of all other variables. The location prediction problem can be modeled in this framework by assuming that the class label, $f_L(s_i)$, of different locations, s_i, constitute an MRF. In other words, random variable $f_L(s_i)$ is independent of $f_L(s_j)$ if $W(s_i, s_j) = 0$. The Bayesian rule can be

used to predict $f_L(s_i)$ from feature value vector X and neighborhood class label vector L_M as follows:

$$Pr(l(s_i)|X, L\backslash l(s_i)) = \frac{Pr(X(s_i)|l(s_i), L\backslash l(s_i))Pr(l(s_i)|L\backslash l(s_i))}{Pr(X(s_i))}.$$

The solution procedure can estimate $Pr(l(s_i)|L\backslash l(s_i))$ from the training data by examining the ratios of the frequencies of class labels to the total number of locations in the spatial framework. $Pr(X(s_i)|l(s_i), L\backslash l(s_i))$ can be estimated using kernel functions from the observed values in the training dataset. For reliable estimates, even larger training datasets are needed relative to those needed for the Bayesian classifiers without spatial context, since we are estimating a more complex distribution. This is accomplished by taking contiguous regions (windows) instead of sample points. An assumption on $Pr(X(s_i)|l(s_i), L\backslash l(s_i))$ may be useful if large enough training data set is not available. A common assumption is the uniformity of influence from all neighbors of a location. Another common assumption is the independence between X and L_N, hypothesizing that all interaction between neighbors is captured via the interaction in the class label variable. Many domains also use specific parametric probability distribution forms, leading to simpler solution procedures. In addition, it is frequently easier to work with the Gibbs distribution specialized by the locally defined MRF through the Hammersley–Clifford theorem (Besag, 1974).

2.2.1 Handling Big Data Challenges

Solution procedures for the MRF Bayesian classifier include stochastic relaxation (Geman and Geman, 1984), iterated conditional modes (Besag, 1986), dynamic programming (Derin and Elliott, 1987), highest confidence first (Chou et al., 1993), and Graph cut (Boykov et al., 1999). We have explored the graph cut method in the past, more details can be found in Shekhar et al. (2002).

2.3 Gaussian Process Learning

MRFs described in the previous section are widely used to model spatial homogeneity. However, modeling spatial heterogeneity is also important in classification. Statistical modeling of spatial variation has been well known as spatial statistics or geostatistics.

The process of finding the optimal linear predictor is called kriging, named after a South African mining engineer D. G. Krige (Cressie, 1993). In the machine learning community, the same model is known as the Gaussian process regression model. When the underlying stochastic process is a Gaussian random process, the linear predictor obtained by kriging is optimal in least-square sense.

More specifically, the Gaussian process regression model corresponds to a simple or ordinary kriging model.

The conventional maximum-likelihood classifier typically models the class-conditional distribution, $p(\mathbf{x}|y)$, as a multivariate Gaussian distribution:

$$p(\mathbf{x}|y = y_i) \sim N(\boldsymbol{\mu_i}, \Sigma_i), \tag{2}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_d)^{\mathrm{T}}$ is a d-dimensional vector representing spectral bands of a pixel in a hyperspectral image, and $y \in \{y_1, y_2, \ldots, y_c\}$ is the LULC class label. The parameters for multivariate Gaussians, $\boldsymbol{\theta_i} = (\boldsymbol{\mu_i}, \Sigma_i)$, are obtained by the maximum-likelihood estimation (MLE), and assumed to be constant over all possible locations. As discussed earlier, this assumption do not hold in general. Spatial heterogeneity can be modeled via nonparametric Gaussian process model. In this model, the class-conditional distribution of the i-th class is modeled as a function of spatial coordinate $\mathbf{s}$:

$$p(\mathbf{x}(\mathbf{s})|y_i) \sim N(\boldsymbol{\mu_i}(\mathbf{s}),\ \Sigma_i), \tag{3}$$

where $\boldsymbol{\mu_i}(\mathbf{s}^*) = (\mu_{i1}(\mathbf{s}^*), \mu_{i2}(\mathbf{s}^*), \ldots, \mu_{id}(\mathbf{s}^*))$. We omit i that indicates the i-th class in the following equations to simplify the notation. Each spectral band of $\mathbf{x}$ is modeled as a random process indexed by a spatial coordinate $\mathbf{s} = (s_1, s_2)$, then the j-th band of $\mathbf{x}$, x_j, can be written as

$$x_j(\mathbf{s}) = f_j(\mathbf{s}) + \epsilon_j, \tag{4}$$

where $f_j(\mathbf{s})$ is a Gaussian random process and ϵ_j is an additive white Gaussian noise:

$$\epsilon \sim \mathcal{N}(0, \sigma^2_{\epsilon_j}).$$

Given $f_j(\mathbf{s})$, then the class conditional distribution of x_j is

$$p(x_j(\mathbf{s})|f_j(\mathbf{s})) = \mathcal{N}(f_j(\mathbf{s}),\ \sigma^2_{\epsilon_j}).$$

We assume a (zero-mean) Gaussian process for $f_j(\mathbf{s})$:

$$f_j(\mathbf{s}) \sim \mathcal{GP}(0,\ K_j(\mathbf{s}_l, \mathbf{s}_m)),$$

where $K_j(\mathbf{s}_l, \mathbf{s}_m)$ is a spatial covariance function between locations $\mathbf{s}_l$ and $\mathbf{s}_m$. The zero-mean prior assumption correspond to the simple kriging model in spatial statistics (Cressie, 1993). In practice, we can approximately satisfy the zero-mean assumption by normalizing given feature values. Characteristics of a Gaussian random process is solely defined by a covariance function. More details on GP-based classification can be found in Jun et al. (2009). In addition to the classification, we have adopted GP for biomass change detection over large

areas (Chandola and Vatsavai, 2010). For change detection, we used a covariance function known as *Exponential Periodic (ep)*:

$$k(t_1, t_2) = \sigma_f^2 \exp\left(-\frac{\Delta t^2}{2l^2\omega^2}\right) \exp\left(-\frac{(1-\cos\frac{2\pi\Delta t}{\omega})}{a}\right) \tag{5}$$

where ω is the length of a single cycle of the periodic time series. This covariance function effectively models the periodic time series.

$$K = \begin{pmatrix} k_0 & k_1 & k_2 & \dots & k_{t-1} \\ k_1 & k_0 & k_1 & \dots & k_{t-2} \\ k_2 & k_1 & k_0 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{t-1} & k_{t-2} & \dots & \dots & k_0 \end{pmatrix} \tag{6}$$

2.3.1 Big Data Challenges

It is well known that GP do not scale well for large datasets. First, the covariance function (Eq. 6) requires $O(n^2)$ memory. The solution consists of estimating the following terms: $\mathbf{y}^\top K^{-1}\mathbf{y}$, $k^\top K^{-1}\mathbf{y}$, $k^\top K^{-1}k$. Straight forward solutions are computationally expensive, $O(t^3)$.

2.3.2 Handling Big Data Challenges

However, noting that the covariance function (6) leads to a symmetric Toeplitz and positive semi-definitive matrix, the memory and computational requirements reduces to $O(n)$ and $O(n^2)$ respectively. More details can be found in Chandola and Vatsavai (2010). Even after employing computationally efficient algorithms, change detection is a challenging task. For example, in the case of biomass monitoring using coarse spatial resolution (250 m) MODIS data, one has to process 23,040,000 time series for one (tile) image. One has to process 326 such tiles (that is, 7,511,040,000 individual time series) in a day before new images arrive.

2.4 Mixture Models

Mixture models are widely used in clustering and semi-supervised learning. In this section, we present a Gaussian Mixture Model (GMM) based clustering algorithm. GMM based clustering consists of two subproblems. First, we have to estimate the model parameters. Second, we need to estimate the number of components in the GMM.

2.4.1 Estimating the GMM Parameters

First we solve the model parameter estimation problem by assuming that the training dataset D_j is generated by a finite GMM consisting of M components. If the labels for each of these components were known, then problem simply reduces to usual parameter estimation problem and we could have used MLE. We now describe a parameter estimation technique that is based on the well-known expectation maximization (EM) algorithm. Let us assume that each sample x_j comes from a super-population D, which is a mixture of a finite number (M) of clusters, $D_1, \ldots, D_M$, in some proportions $\alpha_1, \ldots, \alpha_M$, respectively, where $\sum_{i=1}^{M} \alpha_i = 1$ and $\alpha_i \geq 0 (i = 1, \ldots, M)$. Now we can model the data $D = \{x_i\}_{i=1}^{n}$ as being generated independently from the following mixture density.

$$p(x_i|\Theta) = \sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j) \tag{7}$$

$$L(\Theta) = \sum_{i=1}^{n} \ln \left[\sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j) \right]. \tag{8}$$

Here, $p_j(x_i|\theta_j)$ is the pdf corresponding to the mixture j and parameterized by θ_j, and $\Theta = (\alpha_1, \ldots, \alpha_M, \theta_1, \ldots, \theta_M)$ denotes all unknown parameters associated with the M-component mixture density. The *log-likelihood* function for this mixture density is given in (8). In general, Eq. (8) is difficult to optimize because it contains the ln of a sum term. However, this equation greatly simplifies in the presence of unobserved (or incomplete) samples. We now simply proceed to the EM algorithm, interested reader can find detailed derivation of parameters for GMM in Bilmes (1997). The EM algorithm at the first step maximizes the expectation of the *log-likelihood* function, using the current estimate of the parameters and conditioned upon the observed samples. In the second step of the EM algorithm, called maximization, the new estimates of the parameters are computed. The EM algorithm iterates over these two steps until the convergence is reached. For multivariate normal distribution, the expectation $E[.]$, which is denoted by p_{ij}, is the probability that Gaussian mixture j generated the data point i, and is given by:

$$p_{ij} = \frac{\left|\hat{\Sigma}_j\right|^{-1/2} \mathrm{e}^{\left\{-\frac{1}{2}(x_i - \hat{\mu}_j)^t \hat{\Sigma}_j^{-1}(x_i - \hat{\mu}_j)\right\}}}{\sum_{l=1}^{M} \left|\hat{\Sigma}_l\right|^{-1/2} \mathrm{e}^{\left\{-\frac{1}{2}(x_i - \hat{\mu}_l)^t \hat{\Sigma}_l^{-1}(x_i - \hat{\mu}_l)\right\}}} \tag{9}$$

The new estimates (at the k^{th} iteration) of parameters in terms of the old parameters at the M-step are given by the following equations:

$$\hat{\alpha}_j^k = \frac{1}{n}\sum_{i=1}^{n} p_{ij} \tag{10}$$

$$\hat{\mu}_j^k = \frac{\sum_{i=1}^{n} x_i p_{ij}}{\sum_{i=1}^{n} p_{ij}} \tag{11}$$

$$\hat{\Sigma}_j^k = \frac{\sum_{i=1}^{n} p_{ij}(x_i - \hat{\mu}_j^k)(x_i - \hat{\mu}_j^k)^t}{\sum_{i=1}^{n} p_{ij}} \tag{12}$$

2.4.2 Clustering

Once the GMM is fitted to the training data, we can use the model to predict labels for each cluster. The assignment of label is carried out using the ML procedure. The discriminant function $g(.)$ given by ML principle is as following:

$$g_i(x) = -\ln|\Sigma_i| - (x-\mu_i)^t|\Sigma_i|^{-1}(x-\mu_i) \tag{13}$$

For each pixel (feature vector), we assign a cluster label i, if $g_i(x)$ is maximum over all cluster labels.

2.4.3 Big Data Challenges

The computational complexity of GMM model fitting depends on the number of iterations and time to compute expectation (E) and maximization (M) steps. Let us assume that size of training dataset size is N, and the number of components is M, and the dimensionality is d. Then the cost of E- and M-steps are $O(NMD + NM)$ and $O(2NMD)$, respectively, at each iteration. On the other hand spatial extension (Vatsavai et al., 2007) incurs additional cost of iterative conditional mode (ICM) step.

2.4.4 Handling Big Data Challenges

We developed an efficient solution for GMM based spatial semi-supervised learning in Vatsavai et al. (2007). We have also parallelized GMM clustering algorithm on GPUs. Initial results on GTX285 with 240 CUDA cores and 1 GB memory shows excellent scalability of 160× on learning part. Learning part is typically computationally expensive and less I/O intensive, as we have to deal with small training data which is typically 3–5% of the total data. However, for clustering (Eq. 13), that is assigning label to each pixel in the image, the performance is suffered by I/O as we have to deal with 95–97% of the total data. The main reason being that the computational requirements of clustering are modest (Eq. 13), as compared to the learning, but the number of samples to process are huge. We need efficient I/O schemes to scale-up clustering for large datasets.

3 APPLICATIONS

In this section, we present a diverse but representative set of applications dealing with big spatial data.

3.1 Biomass Monitoring

Monitoring biomass over large geographic regions for identifying changes is an important task in many applications. With recent emphasis on biofuel development for reducing dependency on fossil fuels and reducing carbon emissions from energy production and consumption, the landscape of many countries is going to change dramatically in coming years. Already there are several preliminary reports that address both economic and environmental impacts of growing energy crops. In the United States, continuous corn production is becoming a dominant cropping pattern as more and more soybean and wheat rotations are replaced by continuous corn production. It is also expected that more and more pasture lands will be converted to Switchgrass in the coming years, which may positively impact climate change because of its superior carbon uptake properties. These changes are not limited to the United States alone. Developing countries like India, the rural areas are facing increasing demand for energy. It is expected that energy crops like Jatropha curcas are going to be widely planted in Asian countries. Recent FAO report (Brittaine and Lutaladio, 2010) indicates a threefold increase in the area planted to jatropha from 4.72 in 2010 to 12.8 million ha by 2015.

Monitoring biomass over a large geographic region requires high temporal resolution satellite imagery. The launch of NASA's Terra satellite in December of 1999, with the MODIS instrument aboard, introduced a new opportunity for continuous monitoring of biomass over large geographic regions. MODIS data sets represent a new and improved capability for terrestrial satellite remote sensing aimed at meeting the needs of global change research (Justice et al., 1998). MODIS land products are generally available within weeks or even days of acquisition and distributed through the EROS Data Center and are currently available free of charge. The availability of multi-temporal MODIS imagery has made it possible to study plant phenology, quantitatively describe NPP patterns in time and space, and monitor and map natural resources at regional and global scales. MODIS allows users to identify vegetation changes over time across a region and estimate quantitative biophysical parameters, which can be incorporated into global climate models. Even though several cumulative vegetation indices can be found in the literature, MODIS NDVI temporal profiles are widely used in studying plant phenology.

Since data at global scale is difficult to handle, MODIS data is organized into tiles of $10^\circ \times 10^\circ$ (4800×4800 pixels). Though there are 460 daily MODIS tile products available, we need to process 326 products, which contain land pixels. At daily temporal resolution, MODIS time series contains about 3600 data points

(at each pixel location). Often the computational complexity of change detection algorithms is very high, for example, GP learning presented in the Section 2.3 is $O(n^3)$ and $O(n^2)$, where n is the number of data points in each time series. In addition, we need to process about 7,511,040,000 time series in a day, where each time series contains 3600 data points, before new set of MODIS data products arrive. In addition to finding changes, we also need to characterize changes using high-resolution satellite image products, which put tremendous constraints on computational resources. More details about a scalable biomass monitoring system can be found in Vatsavai (2009) and Chandola and Vatsavai (2011a,b).

3.2 Complex Object Recognition

Most of the pattern recognition and machine learning algorithms are per-pixel based (or single instance). These methods worked well for thematic classification of moderate and high-resolution (5 m and above) images. Very high-resolution (VHR) images (submeter) are offering new opportunities beyond thematic mapping, they allow recognition of structures in the images. For example, consider the problem of settlement mapping (Graesser et al., 2012). The high rate of urbanization, political conflicts and ensuing internal displacement of population, and increased poverty in the twentieth century has resulted in rapid increase of informal settlements. These unplanned, unauthorized, and/or unstructured homes, known as informal settlements, shantytowns, barrios, or slums, pose several challenges to the nations as these settlements are often located in most hazardous regions and lack basic services.

Though several World Bank and United Nations sponsored studies stress the importance of poverty maps in designing better policies and interventions, mapping slums of the world is a daunting and challenging task. VHR images provides the ability to distinguish informal settlements from formal settlements. However, per-pixel based methods do not work well for VHR images (submeter). The main problem being that the pixel size (less than meter) is too small as compared to the object size (10 s of meters) and contains too little contextual information to accurately distinguish between given set of pixels. As shown in Fig. 2, often do not provide sufficient discrimination power between classes. One way to alleviate this problem is to consider a bigger window or patch consisting a group of adjacent pixels, which offers better spatial context than a single pixel. Unfortunately, this makes all well known per-pixel based classification schemes ineffective. Multi-instance learning approaches might be useful in moving from pixel-based or object-based structure recognition in VHR images, but computational complexity is too high to be practically applied for global settlement mapping.

Now, let us consider the problem of identifying complex facilities (e.g., nuclear facilities, thermal power plants; Vatsavai et al., 2010) in VHR images. As can be seen from Fig. 3, thematic classification is designed to learn and predict

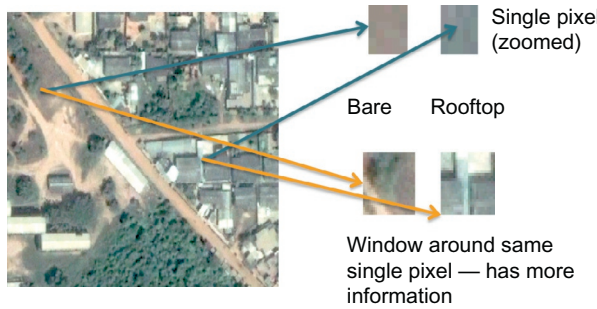


FIGURE 2 Problems with pixel-based pattern recognition methods.

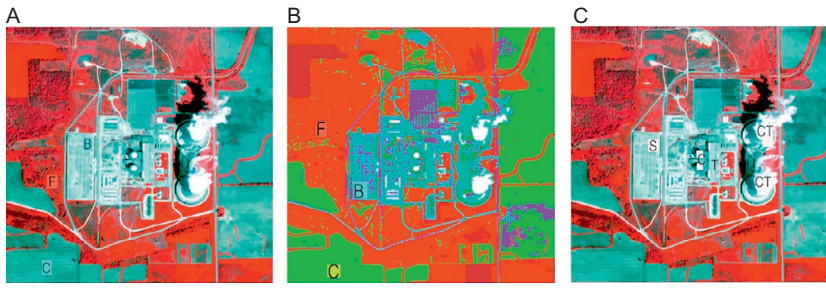


FIGURE 3 Thematic versus semantic classes. (a) FCC image with thematic class labels (B, buildings; C, crop; F, forest). (b) Thematic classified image (B, buildings; C, crop; F, forest). (c) FCC image with semantic labels (S, switch yard; C, containment building; T, turbine generator; CT, cooling tower).

thematic classes such as forest (F), crops (C), buildings (B), etc., at pixel level. However, such thematic labels are not enough to capture the fact that the given image contains a nuclear power plant. What is missing is the fact that the objects, such as switch yard (S), containment building (C), turbine building (T), and cooling towers (CTs) have distinguishing shapes, sizes, and spatial relationships (arrangements or configurations) as shown in Fig. 3c. These semantics are not captured in the traditional pixel- and object-based classification schemes. In addition, traditional image analysis approaches mainly exploit low-level image features (such as, color and texture and, to some extent, size and shape) and are oblivious to higher level descriptors and important spatial (topological) relationships without which we cannot accurately discover these complex objects or higher level semantic concepts. Figure 4 shows four different images (baseball and football fields, two residential neighborhoods) where they share common objects, for example, grass and soil across baseball and football fields, and two (economically) different neighborhoods where in one neighborhood buildings are colocated with cars (parked on the road) while in the other builds are collocated with swimming pools. Both pixel- and object-based methods often

FIGURE 4 Example images sharing similar objects (e.g., grass, buildings, roads, cars, water) but entirely different global labels. (a) Baseball field. (b) Football field. (c) Building + car. (d) Building + swimming pool.

fails to capture these complex relationships. Future research requires models that explicitly learn complex spatial relationships among the objects to accurately predict semantic classes and scale to big VHR image collections.

3.3 Climate Change Studies

The big data challenge is rising to be among the major concerns in furthering our understanding of climate science and translating the science to information relevant for impacts and policy. Here we show, through specific examples drawn primarily from our prior work, how the complexity, size, and heterogeneity of the climate data and corresponding solutions from the data and computational sciences have the potential to transform the field.

The Intergovernmental Panel on Climate Change, in their Nobel Prize winning Fourth Assessment Report (AR4) in 2007, primarily used simulations from the Coupled Model Intercomparison Project Phase 3 (CMIP3), in addition

to observations, to support two major hypotheses: First, global warming is real, and second, human emissions of greenhouse gases are to blame. The CMIP3 multi-model archives, which started at tens of terabytes, have grown to a few hundred terabytes since. The latest generation, or CMIP5 models have finer spatial resolutions (number of longitudes and latitudes) compared to that of CMIP3 (Taylor et al., 2012). The CMIP5 model archives have already reached the petabyte scale and are expected to grow to beyond a couple of petabytes. The size of remotely sensed data is beginning to catch up as well.

Some of the CMIP5 models have incorporated additional physics, carbon cycle feedback, dynamic land vegetation, atmospheric chemistry, and biogeochemistry in order to generate credible future climate projections at smaller spatial scales (Taylor et al., 2012). However, an intercomparison study of CMIP3 and CMIP5 models at regional and seasonal scales reported little changes in the mean climatology and variability of historical skills and consensus among models (Kumar et al., 2014a). Figure 5 illustrates difference in projection maps of seasonal mean precipitation at the end of this century (2070–2099 vs. 1970–1999) from an ensemble of 11 model pairs from both generations. It can be observed that the spatial patterns of dry and wet biases from both generations of models have not change significantly. Similar studies on extreme precipitation (Kumar et al., 2014b) and wind extremes (Kumar et al., 2014a) have not shown improvements in the performance of CMIP5 models (Kumar et al., 2014b).

According to a recent article (Overpeck et al., 2011) in Science magazine, climate-related data is expected to reach about 350 petabytes by the 2030s, with

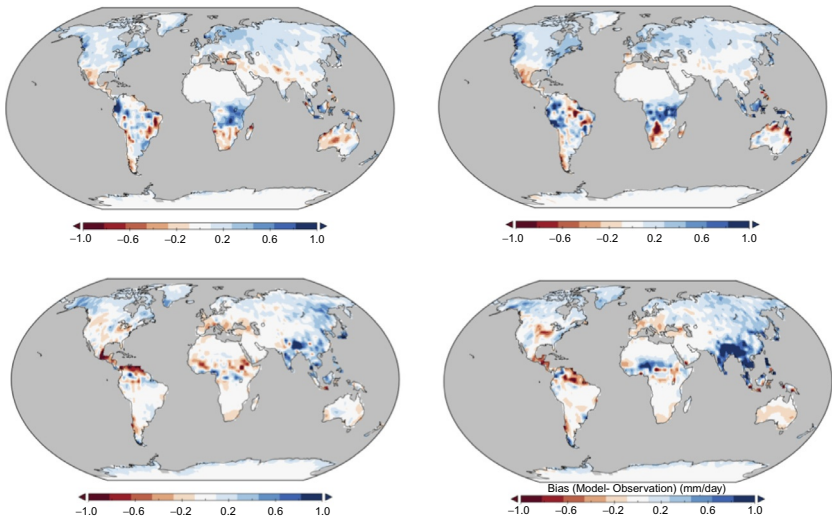


FIGURE 5 Difference in projection maps for multi-model ensemble seasonal precipitation mean [December–January–February (top panel) and June–July–August (bottom panel)]. Eleven pairs of CMIP3 (left panel) and CMIP5 (right panel) models were used to generate the spatial maps.

about half of that contributed by models, a bit less than half by remote sensing data, and the rest from *in situ* sensors. The focus of climate science, meanwhile, has been shifting more towards the regional or even local scales, with a view to consequence assessments and informing adaptation decisions. This has in turn motivated regional climate modeling runs as well as statistical downscaling, which could result orders of magnitude increase in data size. In addition, stakeholders from multiple sectors and policy makers are specifying to perform climate change impact assessment at regional-to-local scales over a planning horizon of 0–30 years (decadal time scales). Uncertainties in future climate projections arise primarily from three sources: model response uncertainty (due to the lack of understanding of physics), scenario uncertainty (inadequate knowledge about how much greenhouse gas will be emitted in the future), and climate natural variability (sensitivity to initial conditions). Uncertainties resulting from climate intrinsic variability dominate at smaller spatial scales and over the 30-year-time horizon. Figure 6 illustrates how spatial patterns of changes in mean annual precipitation in 2030s (2020–2040) with respect to historical climatology (1986–2005) differ just because of different initial conditions. The results were generated from only one climate model, National Climate Atmospheric Research Community Climate System Model (CCSM4) but for four initial conditions.

Even though these are outputs from the same model, we observe a complete change in sign for the mean annual precipitation (dry vs. wet patterns) over several regions: parts of the United States, Australia, Amazonia, Southern Africa, Southeast Asia, and India. The existing climate change impact assessment

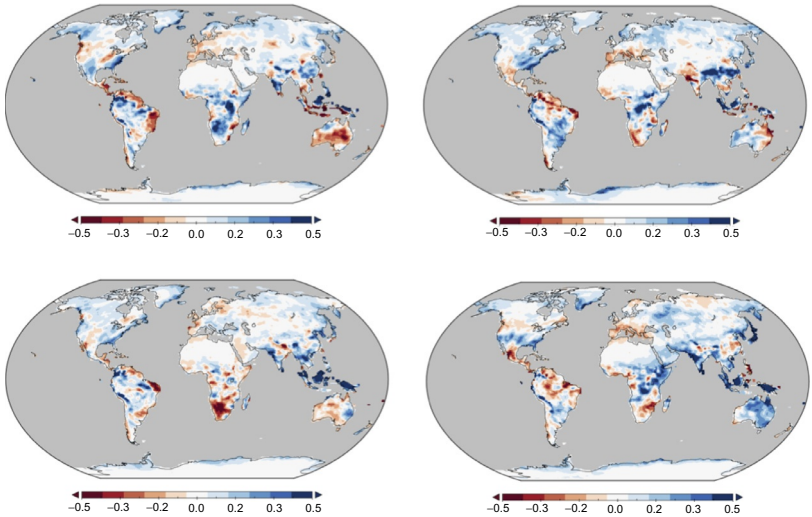


FIGURE 6 Spatial patterns of changes in annual mean precipitation for 2030s (2020–2040) with respect to historical climatology (1986–2005) for four initial conditions from CCSM4 model.

studies employ outputs from one or an ensemble of climate models but only from one initial condition. Presently we lack a coherent framework to perform climate change impact assessment at regional scales over 0–30 years that considers climate model output from all models, all initial conditions, and all greenhouse gas emissions scenarios.

In addition to the sheer size, the complexity of climate data contributes to computational and data science challenges (Ganguly and Steinhaeuser, 2008). While data mining or even machine learning methods and principles were originally developed for independent and identically distributed data, the areas of time series, spatial and spatiotemporal data mining partly emerged as a response to the growing challenge of finding interesting and novel patterns from correlated but massive data.

Time series analysis and forecasting in statistics typically starts with an examination of the dependence structure, for example, through the autocorrelation function, while signal processing starts with the power spectrum which is a representation of the dependence structure in frequency space and hence a Fourier transform of the autocorrelation function. Spatial and spatiotemporal statistics rely on the fundamental notion that values closer to each other in time or space are more closely related to each other, with due considerations for seasonality and low frequency components. In fact, the first law of geography states that all things being equal things closer to each other are more closely related.

The fundamental principles of time series, spatial and spatiotemporal data mining often rely on the same basic principles, for example, auto- or cross-correlation, in space and/or time. Thus, regression and weighted regression techniques need to be adapted through geographical weights or spatial and temporal constraints, leading to numerical and computational challenges. For climate data and processes, however, the situation grows worse. Some the key challenges are as follows:

1. The dynamical climate system exhibits interconnections and dependence structures that may often only be observed as teleconnections and captured through nonlinear correlation measures (Khan et al., 2006) or graphical dependence structures (Kawale et al., 2011), including complex networks (Steinhaeuser et al., 2012) or probabilistic graphs (Chatterjee et al., 2012).
2. Climate processes operate at multiple scales, from the intraseasonal scales of the Madden–Julian Oscillation to the interannual El Nino Southern Oscillation all the way to the Atlantic Multidecadal Oscillation, just as a few examples. The processes can vary across not just time but also spatial scales, which enhance the challenge. From a computational and data perspective, the multiscale behavior of the climate system makes the combinatorics for dependence and prediction structures significantly more complex.
3. Climate processes are nonlinear dynamical, and may even exhibit chaos or extreme sensitivity to initial conditions, while the variability in the data may be non-Gaussian, even showing noise.

4. Climate projections and attributions are typically desired for the longer term (decadal to century scales), which implies uncertainties owing to socioeconomic changes, technological policies, and mitigation regulations.
5. The problems of interest in climate are typically changes at regional scales or for the extremes (Das et al., 2012; Kao and Ganguly, 2011), which makes the data and computational challenges harder.
6. Uncertainty quantification becomes a major challenge (Ganguly et al., 2009) owing to non-stationarity, nonlinear dynamics and long projection lead times with high precision.
7. The consequence of climate change are felt on critical infrastructures and key resources, which implies the need to bring together climate analysis results together with impact-relevant data, for example, through geographical information systems. As climate related hazards and impacts on natural and human systems become more common, the detection of change under non-stationarity and nonlinearity becomes a major concern. Thus, civil and water resources engineers, who work with intensity–duration–frequency curves for designing hydraulic infrastructures or for water resources decisions, would need to consider the changes on the very basis of their design or planning decisions (Kao and Ganguly, 2011).

Computational data sciences can help the science of climate change and impacts in multiple ways. Climate science can benefit from pattern discovery process assessment, analysis and uncertainty characterization, as well as enhanced predictions, particularly at regional scales and for extremes. The translation to impacts, adaptation and vulnerability can especially benefit from precise projections with comprehensive uncertainty characterizations.

4 CONCLUSIONS

Big spatiotemporal data, though opening up new applications, poses several unique challenges. New approaches are required to overcome both computational and I/O challenges, and new models that explicitly model spatial and temporal constraints efficiently. Further research is required in the area of compression and sampling. Especially there is a great need for integrating spatial data mining workflows with modern computing infrastructure like cloud computing, *in situ* (Klasky et al., 2011), data spaces (Docan et al., 2010), and the like.

REFERENCES

Adelman-McCarthy, J.K., et al., 2008. The sixth data release of the sloan digital sky survey. Astrophys. J. Suppl. Ser. 175 (2), 297.

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., Ng, A., 2005. Discriminative learning of Markov random fields for segmentation of 3d scan data. In: CVPR '05:

Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)–Volume 2. IEEE Computer Society, Washington, DC, pp. 169–176.

Anselin, L., 1988. Spatial Econometrics: Methods and Models. Kluwer, Dordrecht, The Netherlands.

Ayhan, S., Pesce, J., Comitz, P., Gerberick, G., Bliesner, S., 2012. Predictive analytics with surveillance big data. In: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, BigSpatial '12, pp. 81–90.

Besag, J., 1974. Spatial interaction and the statistical analysis of lattice systems. J. R. Stat. Soc. 36, 192–236.

Besag, J., 1986. On the statistical analysis of dirty pictures. J. R. Stat. Soc. (48), 259–302.

Bilmes, J., 1997. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report, University of Berkeley, ICSI-TR-97-021, 1997.

Boykov, Y., Veksler, O., Zabih, R., 1999. Fast approximate energy minimization via graph cuts. Int. Conf. Comput. Vis. 23 (11), 1222–1239.

Brittaine, R., Lutaladio, N., 2010. Jatropha: a samllholder bioenergy crop. The potential for pro-poor development. Integr. Crop Manag. 8, 1–114.

Celik, M., Kazar, B., Shekhar, S., Boley, D., Lilja, D., 2007. Northstar: a parameter estimation method for the spatial autoregression model. AHPCRC Technical Report No: 2005-001.

Chandola, V., Vatsavai, R.R., 2010. Scalable time series change detection for biomass monitoring using Gaussian process. In: NASA Conference on Intelligent Data Understanding (CIDU), pp. 69–82.

Chandola, V., Vatsavai, R.R., 2011a. A Gaussian process based online change detection algorithm for monitoring periodic time series. In: Proceedings of the 2011 SIAM International Conference on Data Mining, pp. 95–106.

Chandola, V., Vatsavai, R.R., 2011b. A scalable Gaussian process analysis algorithm for biomass monitoring. Stat. Anal. Data Min. 4 (4), 430–445.

Chatterjee, S., Steinhaeuser, K., Banerjee, A., Chatterjee, S., Ganguly, A.R., 2012. Sparse group lasso: consistency and climate applications. SDM, 4 (4), pp. 47–58.

Chou, P., Cooper, P., Swain, M.J., Brown, C., Wixson, L., 1993. Probabilistic network inference for cooperative high and low levell vision. In: In Markov Random Field, Theory and Applicaitons. Academic Press, New York, NY.

Cressie, N., 1993. Statistics for Spatial Data (Revised Edition). Wiley, New York, NY.

Das, D., Kodra, E., Obradovic, Z., Ganguly, A.R., 2012. Mining extremes: severe rainfall and climate change. In: ECAI, pp. 899–900.

Derin, H., Elliott, H., 1987. Modeling and segmentation of noisy and textured images using Gibbs random fields. IEEE Trans. Pattern Anal. Mach. Intell. (9), 39–55.

Docan, C., Parashar, M., Klasky, S., 2010. DataSpaces: an interaction and coordination framework for coupled simulation workflows. In: HPDC, pp. 25–36.

Ganguly, A.R., Steinhaeuser, K., 2008. Data mining for climate change and impacts. In: ICDM Workshops, pp. 385–394.

Ganguly, A.R., Steinhaeuser, K., Erickson, D.J., Branstetter, M., Parish, E.S., Singh, N., Drake, J.B., Buja, L., 2009. Higher trends but larger uncertainty and geographic variability in 21st century temperature and heat waves. Proc. Nat. Acad. Sci. 106 (37), 15555–15559.

Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. 6 (6), 721–741.

Graesser, J., Cheriyadat, A., Vatsavai, R.R., Chandola, V., Long, J., Bright, E., 2012. Image based characterization of formal and informal neighborhoods in an urban landscape. IEEE J. Sel. Top. Appl. Earth Observ. Remote Sens. 5 (4), 1164–1176.

Jhung, Y., Swain, P.H., 1996. Bayesian contextual classification based on modified M-estimates and Markov random fields. IEEE Trans. Pattern Anal. Mach. Intell. 34 (1), 67–75.

Jitkajornwanich, K., Elmasri, R., McEnery, J., Li, C., 2012. Extracting storm-centric characteristics from raw rainfall data for storm analysis and mining. In: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, BigSpatial '12, pp. 91–99.

Jun, G., Vatsavai, R.R., Ghosh, J., 2009. Spatially adaptive classification and active learning of multispectral data with Gaussian processes. In: ICDM Workshops: Spatial and Spatiotemporal Data Mining (SSTDM), pp. 597–603, URL http://doi.ieeecomputersociety.org/10.1109/ICDMW.2009.107.

Justice, C.O., Vermote, E., Townshend, J.R., Defries, R., Roy, D.P., Hall, D.K., Salomonson, V.V., Privette, J.L., Riggs, G., Strahler, A., Lucht, W., Myneni, R.B., Knyazikhin, Y., Running, S.W., Steve W. Nemani, S.W., Wan, Z., Huete, A.R., van Leeuwen, W., Wolfe, R.E., Giglio, L., Muller, J.P., Lewis, P., Barnsley, M.J., 1998. The moderate resolution imagin spectrradiometer (MODIS): land remote sensing for global chang research. IEEE Trans. Geosci. Remote Sens. 36, 1228–1249.

Kao, S.-C., Ganguly, A.R., 2011. Intensity, duration, and frequency of precipitation extremes under 21st-century warming scenarios. J. Geophys. Res. 116 (D16119), 139–150.

Kawale, J., Liess, S., Kumar, A., Steinbach, M., Ganguly, A., Samatova, N., Semazzi, F., Snyder, P., Kumar, V., 2011. Data-guided discovery of climate dipoles in observations and models. In: NASA Conference on Intelligent Data Understanding (CIDU), pp. 1–15.

Kazar, B., Shekhar, S., Lilja, D., Vatsavai, R., Pace, R., 2004. Comparing exact and approximate spatial auto-regression model solutions for spatial data analysis. In: Third International Conference on Geographic Information Science (GIScience2004). LNCS, Springer.

Khan, S., Ganguly, A., Bandyopadhyay, S., Saigal, S., Erickson, D., Protopopescu, V., Ostrouchov, G., 2006. Non-linear statistics reveals stronger ties between ENSO and the tropical hydrological cycle. Geophys. Res. Lett. 33 (L24402), 6.

Klasky, S., et al., 2011. In situ data processing for extreme-scale computing. In: SicDAC, pp. 16.

Kumar, D., Kodra, E., Ganguly, A., 2014a. Regional and seasonal intercomparison of CMIP3 and CMIP5 climate model ensembles for temperature and precipitation. Clim. Dyn. 43 (9-10), 2491–2518.

Kumar, D., Mishra, V., Ganguly, A., 2014b. Evaluating wind extremes in CMIP5 climate models. Clim. Dyn. 45, 1–13.

LeSage, J., 1997a. Bayesian estimation of spatial autoregressive models. Int. Reg. Sci. Rev. (20), 113–129.

LeSage, J., 1997b. Regression analysis of spatial data. J. Reg. Anal. Policy 27 (2), 83–94.

LeSage, J.P., Pace, R., 2001. Spatial dependence in data mining. In: Geographic Data Mining and Knowledge Discovery. Springer, New York.

Li, S.Z., 2001. Markov Random Field Modeling in Image Analysis. Springer-Verlag New York, Inc., Secaucus, NJ. ISBN 4-431-70309-8.

Ma, C., 2004. Spatial autoregression and related spatio-temporal models. J. Multivar. Anal. 88 (1), 152–162.

McLachlan, G.J., Basford, K.E., 1988. Mixture Models: Inference and Applications to Clustering. Marcel Dekker, New York.

Overpeck, J.T., Meehl, G.A., Bony, S., Easterling, D.R., 2011. Climate data challenges in the 21st century. Science 331 (6018), 700–702. URL http://www.sciencemag.org/content/331/6018/700.full.pdf.

Pace, R., Barry, R., 1997a. Quick computation of regressions with a spatially autoregressive dependent variable. Geogr. Anal. 29 (3), 232–247.

Pace, R., Barry, R., 1997b. Sparse spatial autoregressions. Stat. Prob. Lett. (33), 291–297.

Rasmussen, C., Williams, C., 2006. Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning. MIT Press. ISBN 9780262182539.

Shekhar, S., Schrater, P., Vatsavai, R., Wu, W., Chawla, S., 2002. Spatial contextual classification and prediction models for mining geospatial data. IEEE Trans. Multimedia 4 (2), 174–188.

Solberg, A.H., Taxt, T., Jain, A.K., 1996. A Markov random field model for classification of multisource satellite imagery. IEEE Trans. Geosci. Remote Sens. 34 (1), 100–113.

Steinhaeuser, K., Ganguly, A., Chawla, N., 2012. Multivariate and multiscale dependence in the global climate system revealed through complex networks. Clim. Dyn. 39, 889–895. ISSN 0930-7575.

Taylor, K.E., Stouffer, R.J., Meehl, G.A., 2012. An overview of CMIP5 and the experiment design. Bull. Am. Meteorol. Soc. 93, 485–498.

Vatsavai, R.R., 2009. BioMon: a Google Earth based continuous biomass monitoring system. In: 17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems (ACM-GIS). ACM, New York, NY, pp. 536–537.

Vatsavai, R.R., 2011. STPMiner: a highperformance spatiotemporal pattern mining toolbox. In: Proceedings of the 2nd International Workshop on Petascal Data Analytics: Challenges and Opportunities, Seattle, Washington, USA, PDAC '11. ACM, New York, NY, pp. 29–34.

Vatsavai, R.R., Shekhar, S., Burk, T.E., 2007. An efficient spatial semi-supervised learning algorithm. Int. J. Parallel Emergent Distrib. Syst. 22 (6), 427–437. URL http://dx.doi.org/10.1080/17445760701207546.

Vatsavai, R.R., Cheriyadat, A., Gleason, S.S., 2010. Unsupervised semantic labeling framework for identification of complex facilities in high-resolution remote sensing images. In: ICDM Workshops, pp. 273–280.

Warrender, C.E., Augusteijn, M.F., 1999. Fusion of image classifications using Bayesian techniques with Markov rand fields. Int. J. Remote Sens. 20 (10), 1987–2002.

Chapter 11

Experimental Computational Simulation Environments for Big Data Analytic in Social Sciences

Michal Galas[*,1]
[*] *Department of Computer Science, University College London, London, United Kingdom*
[1] *Corresponding author: e-mail: m.galas@cs.ucl.ac.uk*

ABSTRACT
Experimental computational simulation environments (e.g., Galas, 2014) are increasingly being developed by major financial institutions to model their analytic algorithms; this includes evaluation of algorithm stability, estimation of its optimal parameters, and the expected risk and performance profiles. Such environments rely on big data analytics (e.g., McAfee and Brynjolfsson, 2012), as part of their software infrastructure, to enable large-scale testing, optimization, and monitoring of algorithms running in virtual or real mode.

At UCL, we believe that such environments will have a profound impact on the way research is conducted in social sciences. Consequently, for the last few years, we have been working on our DRACUS system, a state of-the-art computational simulation environment, believed to be the first available for academic research in Computational Finance, specifically Financial Economics. As part of the DRACUS project, we work to support simulations in algorithmic trading, systemic risk, and sentiment analysis.

Keywords: Computational simulation environments, Financial software systems, Big data analytics

1 INTRODUCTION

Although we have learned how to store and search through large volumes of information, such knowledge does not solve the inherent analytic problem itself. Scientists need environments that will allow them to efficiently perform large-scale experiments and analytical tasks, rather than simply store and search through data. This is particularly apparent in social sciences, especially

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00011-3

Econophysics (e.g., Mantegna and Stanley, 1999) where economic processes are modeled and described with respect to their performance on large volumes of data.

To effectively conduct research in finance and economics, one needs suitable models of trading agents (traders, banks, households, etc.) and market exchanges (to facilitate the exchange of assets). Trading agents can be modeled on various levels of abstractions: from individual traders (or their strategies) to financial institutions, banks, or economies (of entire nations). These models are typically represented from the order-book perspective, to track the money-flow process.

Furthermore, due to the possibility of tracking interactions between multiple active elements, the multiagent simulation framework (e.g., Luke et al., 2004) is considered particularly suitable for socioeconomic-financial simulations. Multiagency can be generally represented in a twofold manner, as either a set of completely independent agents coexisting within a given environment (with a set of behavioral policies triggering actions and reactions from the environment) or a set of behavioral policies triggered externally by the environment to simulate multiagency.

When put together, the ideas introduced above create a rich sociofinancial-economic framework for academic experiments. In the following sections, we will provide more details of such framework, including (a) the key concepts in big data analytics (BDA), (b) the typical templates for socioeconomic-financial simulations, (c) the software infrastructure for social sciences, (d) the market simulators for financial economics, and (e) the DRACUS platform, an example of big data environment for analytics and simulation in social sciences.

2 BIG DATA ANALYTICS

Most social scientists lack the computer skills required to handle large amounts of data. Consequently, the experimental computational simulation environments are designed to embed the low-level big data functionalities to enable the researchers to focus on the core analytic tasks. The following section outlines such low-level big data components.

BDA was created to handle data volume, velocity, and variety and is concerned with analysis of large databanks of information. The concept of volume is the most straightforward and is related to the fact that real-life analytic model needs to be able to handle large volumes of data. The concept of velocity is related to the rate at which new data arrive and need to be analyzed by the model. Finally, the concept of variety is related to the various forms of data analyzed by the model, e.g., unstructured data, time-series data, text, audio/video, or geolocation data. The Hadoop Technology Stack (Fig. 1) is a particular implementation of the BDA concept and enables reliable, scalable, distributed computing. This technology stack is built out of the following set of components.

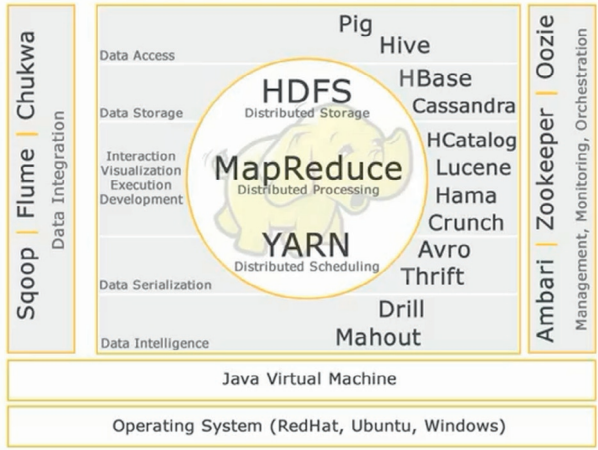


FIGURE 1 Hadoop Technology Stack (http://hadoop.apache.org/).

Core components: there are three core components of Hadoop, namely: (a) the HDFS (Hadoop Distributed File System) providing distributed storage, (b) MapRecude for distributed data retrieval/processing, and (c) YARN/MR2 (Yet Another Resource Negotiator) for distributed scheduling. The HDFS is capable of dividing data into blocks and distributes such blocks over the cluster. It is also capable of data replication of blocks for fault tolerance. The MapRecude is a two-step process (involving a mapper and a reducer), where the mapper is responsible for finding the required data in the cluster and the reducer processes the data.

Cluster management[1]: Zookeeper is a distributed service coordinator that enables synchronization of services in the cluster. It also provides Hadoop with a central management point. Oozie is a workflow library that enables Hadoop to automate workflow between different elements of the technology stack. Ambari enables provision of services to the cluster; e.g., installation, monitoring, and services control across all the nodes in the cluster.

Data access[2]: the Hive technology enables SQL-like queries rather than Java programming of MapReduce. Pig is another element of Hadoop stack, which is a high-level data flow language to pull data out of the cluster. Both Hive and Ping run MapReduce in the underlying layer and provide Hadoop with flexible ways to retrieve data. Hadoop is also naturally scalable due to the underlying cluster structure.

1. http://zookeeper.apache.org/; https://oozie.apache.org/; http://ambari.apache.org/.
2. http://hive.apache.org/; http://pig.apache.org/.

Data storage[3]: HBase is based on Google's big table, while Cassandra is based on Amazons dynamo data storage. Both are column-oriented NOSQL databases wrapped around the HDFS technology.

Data utilities[4]: HCatalog (now part of Hive) is a metadata table and storage management system that enables creation of shared schemes for consistent view of data. Lucene is designed for full text pattern searching (wildcards, phrases, etc.). Hama is designed for box synchronous processing. Crunch is designed for running MapReduce pipelines (map, shuffle, reduce combine).

Data serialization[5]: Avro and Thrift are designed for data exchange and storage where Avro is a more general framework, while Thrift is language agnostic and can enable serialization and deserialization of data for different languages, e.g., Java/C++.

Data intelligence[6]: Mahout is a general machine learning library for data analytics, while Drill is designed for interactive analysis on nested data.

Data integration[7]: Sqoop is designed to integrate Hadoop with relational systems (e.g., for archiving or post analysis), and Flume and Chukwa are frameworks for real-time log processing, enabling forwarding of data to Hadoop and analysis of logs in real time.

To summarize, BDA helps to explain large volumes of different types of data at a significant processing rate. BDA enables to process previously ignored information that, if correctly analyzed and interpreted, may be business critical. It helps to find insight in new emerging types of data, or deeper insight in already existing data sets. Finally, BDA can be considered a platform for data mining/information retrieval and, when extended, can be used to form an experimental computational simulation environment for, e.g., sociofinancial-economic problems.

3 SOCIOFINANCIAL-ECONOMIC SIMULATIONS

The concept of BDA can be taken further not only to serve the purpose of finding interesting statistical relations between data but also to support observations of behavior of analytical models (while they interact with physical processes unraveling in streamed or stored data). One can identify a set of simulation templates suitable for modeling with use of socioeconomic-financial data. Such simulation templates (e.g., Galas, 2014) can be described as follows:

Free-Run: This is a process of simulation on-the-fly, with results being generated on the basis of newly provided input parameters, as soon as they

3. http://hbase.apache.org/; http://cassandra.apache.org.
4. http://lucene.apache.org/core/; https://hama.apache.org/; http://crunch.apache.org/.
5. http://avro.apache.org/; http://thrift.apache.org/.
6. https://mahout.apache.org/; http://incubator.apache.org/drill/.
7. http://sqoop.apache.org/; http://flume.apache.org/; https://chukwa.apache.org/.

appear. A key property of the free-run process is the fact that the investigated model cannot affect the input data and consequently the environment in which it resides (the sandbox effect, a lack of the feedback loop to the environment). This property is the main differentiator between the free-run and the actual execution of the model.

Back-Test: This is a process of simulation on the basis of historical records of event occurrences. The goal of the process is to observe a potential behavior of a model in the past, to eliminate any unexpected and disadvantageous behaviors, and to maximize the impact of advantageous behaviors in the future. Furthermore, the back-test processes are used for model tuning and for estimation of model behaviors.

Stress-Test: While the back-test can be considered a walk over the historic "expected data circumstances," the stress-test is a process of simulation on the basis of "abnormal (highly unexpected) data circumstances." The goal of such process is to estimate and improve the model's susceptibility to such data circumstances.

Forward (Monte-Carlo)-Test: This is a process of simulation on the basis of a synthetic data generated for multiple, alternative, scenarios most commonly with use of a form of a linear drift and an element enforcing stochastic process. Often the scenarios are generated with parameters for the drift and stochastic process estimated from the historic data to enforce continuity between the history and the potential future scenario. This approach is used to estimate the most probable statistical distribution of future scenarios and consequently the most probable statistical distribution of model behaviors. This approach can also be used to bind the model within the most advantageous thresholds of future scenarios.

Sensitivity-Test: This is a process of model simulation on the basis of a synthetic data of different time and values granularity that allows estimation of models' sensitivity characteristics: susceptibility to the minimal changes in such data granularities. This approach is used to evaluate the most optimal range of the data granularity for the model, as well as the boundaries within which the model exhibits the expected behavior.

Optimization: This is a process of running multiple disconnected simulations with use of the same model but on the basis of different model parameters, with a goal to maximize or minimize an outcome of a simulation measured by the utility function of the model. The key feature of optimization is a possibility of repetitiveness of an achieved outcome. Given that the free-run, the back-test, the stress-test, the forward-test, and the sensitivity-test are all forms of simulation, they can be used as an underlying process for optimization (with an exception that different goals of an outcome for each type of simulation are expected).

Multiagent simulation: All the mentioned types of simulations and optimization can be performed on a single-agent model or a multiagent model. Typically, a single-agent model implies that the model itself is the only active element in the simulation that reacts to the stimuli generated by the simulation process. In

a multiagent simulation, the model is an aggregate of the active elements being active and reactive to the simulation process and the rest of the active elements in the model.

4 SOFTWARE INFRASTRUCTURE FOR SOCIAL SCIENCES

The simulation classes identified in previous section may, apart from the Hadoop Technology Stack, require a broader software infrastructure support (e.g., for information streaming, processing, simulation controlling, monitoring, or user interaction). Such infrastructure is described below.

Effective large-scale analytics and simulation require sufficient software infrastructure to support the intended statistical tasks. A variety of related work concerning financial data and environments were identified in both academia and the industry. They can be grouped into (a) Financial Data Streaming and Market Information Services, (b) Complex Event Processing Engines, (c) Analytical Tools and Business Intelligence, (d) Experimental and Simulation Environments, and (e) Trading Platforms for Manual and Algorithmic/Automated Traders. These are reviewed below.

4.1 Financial Data Streaming and Market Information Services

A number of papers survey the data sharing services, such as the market information services, or the messaging systems, and the large-scale live data streaming of financial data (e.g., Ainsworth, 2009; Andrade et al., 2009). The major academic platforms for financial and economic data are the Wharton Research Data Services,[8] which provides web-based, terminal-based, and programmatic access to its resources for data mining.

The major commercial platforms for financial data are the Bloomberg[9] and the Thomson Reuters[10] platforms. Both are capable of streaming large quantities of data in either live or historical mode, and both architectures provide big data storage on the server-side and buffering capabilities on the client-side. Apart from the data streaming, both platforms provide basic data analytics on the client-side and charting functionalities for data visualization.

4.2 Complex Event Processing Engines

Complex event processing (CEP) engines are capable of processing events organized in an SQL-style type of queries, constructed on the basis of table-like buffers created and maintained in near-real time. CEP technologies can

8. https://wrds-web.wharton.upenn.edu/wrds.
9. http://www.bloomberg.com.
10. http://www.thomsonreuters.com.

be considered general analytical tools best utilized for the purpose of the on-the-fly monitoring and analytical calculations (e.g., Chandramouli et al., 2010; Luckham, 2008).

Prominent commercial CEP engines include StreamBase[11] and StreamInsight.[12] Both systems offer SQL-like languages to query events on-the-fly. Open source systems also exist with Esper[13] being well regarded.

4.3 Analytical Tools and Business Intelligence

Analytical tools are the mainstay of science and engineering, spanning statistical libraries, data mining frameworks and machine learning toolkits (e.g., Gangadharan and Swami, 2004; Khan and Quadri, 2012).

Well-known general-purpose analytical toolkits are Matlab, R, Mathematica, SPSS, and SAS. Their design supports access to historical and live sources, and quantitative data manipulations on the processed data. Matlab, R, and Mathematica can be considered lower-level platforms, providing rich environments consisting of libraries of application-specific analytical functions where users can code in the platform-specific languages. R and Mathematica focus mainly on historical data analytics, although limited event analytics is also possible. SPSS and SAS can be considered higher-level analytical platforms with a strong support for machine learning and data mining functionalities.

Well-known analytical libraries include the Numeric Algorithms Group[14] which provides a variety of numerical libraries, compilers, and visualization tools for engineering, scientific, and research applications. Likewise, machine learning frameworks, usually available on an open-source basis, include Weka[15] and Encog.[16]

4.4 Experimental and Simulation Environments

Computing clouds, grids, and clusters are often utilized as experimental environments for running simulations, to model data and to calibrate models, especially for large-scale financial analytics (e.g., Bullock, 2011). Various commercial environments exist, with the most prominent being Google Cloud Services,[17] Amazon Web Services,[18] and Microsoft Azure Cloud.[19]

11. http://www.streambase.com.
12. http://msdn.microsoft.com/en-us/library/ee362541.aspx.
13. http://esper.codehaus.org.
14. http://www.nag.co.uk.
15. http://www.cs.waikato.ac.nz/ml/weka.
16. http://www.heatonresearch.com/encog.
17. http://www.google.com/enterprise/cloud/index.html.
18. http://aws.amazon.com/.
19. http://www.windowsazure.com/en-us.

All major universities provide experimental environments of these types. For example, UCL currently hosts three research computing facilities: the Legion Cluster, the Unity SMP, and the Condor Pool.[20]

4.5 Trading Platforms for Manual and Algorithmic Traders

Trading platforms are systems capable of supporting manual and algorithmic traders. Such systems typically support data provision, visualization and data analytics, and the forwarding of trades to exchanges. More sophisticated trading platforms also support risk management and posttrade analysis and provide an API for implementation of trading models. Major banks and hedge funds typically have proprietary trading platforms. There are also vendor platforms, such as MetaTrader, TradeStation, and X-Trader among others, and even open source systems, e.g., Marketcetera.

The majority of academic research on algorithmic trading (AT) is devoted to data analytics, algorithms for forecasting, pricing, and risk. Significantly less academic work investigates the trading platforms and the trade-support infrastructure (e.g., Galas et al., 2012).

4.6 Social Media Platforms for Sentiment Analysis

Social media platforms are analytic systems designed to aggregate, process, and analyze large quantities of publicly available social data including, i.e., RSS feeds and web blogs. These types of platforms are capable of supporting analysis of data, data modeling and monitoring as well as model simulation and evaluation. This includes social pattern recognition, data mining, and predictive analytics. Platforms and services for social analytics are being provided by Thompson Reuters, Google, and SAP among other vendors.

Academic research investigates numerous areas of social analytics. While the early research focused on the basic relationships between social features, e.g., brand recognition and its relationship with financial performance of companies (e.g., Yoon et al., 1993) later research focuses on sentiment detection and analysis (e.g., Vovsha and Passonneau, 2011), technologies that support sentiment analysis (e.g., Chua et al., 2009), and predictive influence that social data may have, for example, on trading (e.g., Vincent and Armstrong, 2010).

When considered individually, the systems and technologies summarized above are capable of providing partial, limited functionality for computational simulation environment. Their strength, however, may be really appreciated when combined together, as features of a large-scale experimental computational simulation environment for, e.g., social sciences.

20. http://www.ucl.ac.uk/isd/common/researchcomputing/services.

5 MARKET SIMULATORS FOR FINANCIAL ECONOMICS MODELING

When considering experimental computational simulation environments from the perspective of simulation logic, one element appears to be central to most of such simulations. The following section will discuss Market Simulators, the central point in financial economics.

Within computational simulation environments an electronic market simulator (EMS) (e.g., Galas, 2014) can be defined as a software system that provides a selection of functionalities of a physical exchange for the purpose of transaction simulation. The selection of functionalities depends on the purpose of simulation.

The users of such systems are interested in the simulation of individual transactions or groups of transactions, for the modeling of action flows of such systems, or for the modeling of behavior of the entire agent-trading ecosystems. They may also be interested in improving the software architectures of trading exchanges or in investigating the theoretical basis of the exchange theory (e.g., investigating game theory that may improve the process of the exchange of goods or improve the management of systemic risk). Such systems allowed the aspiring traders (e.g., students), to trade virtually and gain experience, without losing real funds. AT institutions are using the simulated trading exchanges to "fine-tune" their algorithms, before enabling them for real trading. Simulations of exchanges provide users the chance to participate in a market, receive live feeds, and monitor their profit or loss without the burden of bearing risks associated with the real market trading.

The remaining part of this section will focus on the design and implementation of EMS architecture and a software service capable of providing such functionality. It will introduce a simulation model that embodies the key features of the electronic market exchange.

5.1 Simulation Types and Applications

Given all the various scenarios and applications, in which the EMS can be utilized, we can differentiate four key models of simulators:

LIVE EMS for multiagent transaction simulation: The live, on-the-fly exchange simulator that enables matching and execution of the participant–issued order–instructions against the real or synthetic price quotes. The simulator typically respects a major part of the standard communication protocol, including the key types of the order instructions, the price quotes, and the execution reports. Such a simulator is characterized by utilization of the simplified version of the order validation and clearing functionalities, the lack of logging functionality, the lack of auction functionality, and the simplified version of accountancy and the P&L clearing modules. The order book and the

price marker functionalities are also simplified. The key focus in the design of this type of the EMS simulator is shifted to the exchange protocol.

EOD EMS for end-of-day multiagent transaction simulation: The end-of-day model of the exchange simulator is similar to the live model in that it is able to receive and send the on-the-fly events, indicating intent for transaction against the real or synthetic price quotes. The execution model, however, is significantly different in that the process of clearing, matching, and execution of all the orders, accepted in a given day, is scheduled to happen only at the end of the day. The only processing that happens live, in the on-the-fly mode, is the process of order-parameter validation. At the end of the execution process, the exchange generates the execution reports, indicating the state of transaction. One significant difference here is the fact that only the full-order fills are possible. Another significant difference is the fact that the orders are matched against the one-day candle representation of an average price, rather than against the price quotes (ticks).

While considering the case of the EOD EMS, a simplification in the execution protocol of that type of simulator is to enable the approximation of daily execution. This is particularly useful for portfolio managers who seldom trade intraday but are instead interested on a longer investment horizons. Furthermore, this simulator is expected to not contain the auction functionality at all.

HISTORY-TEST EMS for single-agent historic transaction simulation: The exchange simulator for back-tests differs significantly from the two previous models in that it is designed to run on the historical time-series of price quotes (ticks or higher granularity data, e.g., candlestick—open/high/low/close data pattern) and is typically designed for a single participant/agent only. During, e.g., the back-test process, the back-test manager retrieves (from a database) the time-series of price quotes for required securities and feeds it synchronously to both the participant's API and the exchange simulator. The participant's API contains an AT strategy that is submitted to the back-test. The back-test manager feeds individual quotes to both the functionalities and waits, after every quote, for notification in case the strategy logic is to issue any orders. In case orders are issued, the order instructions are entered to the exchange simulator for immediate execution (during the execution, it is assumed that all the steps in the exchange communication protocol are respected). After the matching process is finished and the strategy receives order execution confirmation in a form of an execution report, the back-test manager moves to the next feed cycle, providing subsequent quote to the strategy and the simulator. This process continues up until the point where the entire anticipated history of quotes is exhausted.

The key simplification in this type of market simulator is the fact that the feed cycles are executed on single threads and the threads are responsible for running all the calculations in the simulator and the API (each simulation cycle starts after the orders and the prices are provided for processing in the cycle). The accountancy functionality requires only one account for the current user and

no authentication is necessary. The utilization of any messaging functionality is also minimized and the strategy is coupled directly with the exchange simulator.

LIVE EMS for multiagent financial system simulation: This is an order vs. order simulator, which focuses on modeling and measuring responses of trading agents to changing market circumstances (rather than focusing on strictly following exchange protocol, as in case of the previously described simulators). Consequently, the order book and the matching engine of the simulator contain additional functionalities allowing introduction of market shocks (e.g., Subrahmanyam and Titman, 2012), regime changes (e.g., Ang and Timmermann, 2011), or influence of changes of trading parameters on an agent behavior. This enabled the multiagent exchange simulations to support modeling of large-scale group behavior of market participants. Since the number of actively trading agents can be as high as a few hundred thousands, the simulator needs to be able to handle large amounts of highly concurrent connections at the same time. The authentication functionality however, as well as the range or handled securities and order types, is significantly simplified. As latency is not considered a key issue, the P&L calculation and tracing functionalities can be extended to provide more sophisticated methods of evaluating participants.

HISTORY-TEST EMS for multiagent financial system simulation: this order vs. order type of simulator is designed to model the entire ecosystem of agents and exchanges over time. Rather than using one exchange simulator, the back-test system utilizes a few exchanges with different execution regimes, different execution parameters and with a possibility of introducing, e.g., endogenous and exogenous shocks (e.g., Johansen and Sornette, 2010) to the system. The entire system is submitted to either a back-test (over historic data sets of price quotes stored in the database) or a stress-test (over a multiset of synthetic time-series of price quotes), all to observe evolution of the system.

The accountancy functionality of the utilized simulators is significantly simplified to hold basic trading statistics. The clearing or authentication functionalities are also minimalistic. The order book and the matching engine functionalities support only the most basic order instructions. In most cases, the utilized simulators support only the end-of-day mode, since trading cycles may be simplified not to incur overcomplicated functionalities. On the other hand, such simulators can contain sophisticated stop functionalities or global risk and market stability measures to support evaluation of the state of the ecosystem.

6 STATISTICAL SIMULATIONS OF AT MODELS

With an appropriate software infrastructure (supporting (a) various types of simulation templates, and (b) access to large amounts of data and market simulator functionality), we can now consider examples of models that are to be submitted to the process of simulation. The following section is focused on the

AT, which the author considers a great case study for socioeconomic-financial modeling, due to its strong relations to statistics, machine learning, computer science, and finance.

In electronic financial markets, AT (e.g., Treleaven and Galas, 2013; Treleaven et al., 2013) is a field of computational finance that combines different analytical techniques, among others (from statistics, machine learning, physics, and economics) to create algorithms capable of taking, executing, and administering investment decisions with optimal levels of profit and risk. AT aims to automate one or more stages of the trading process, where the stages can be defined as (a) pretrade analysis (data analysis, state of the world analysis); (b) signal generation (decision-taking process, policy formation); (c) trade execution (execution of actions, policy execution); and (d) posttrade analysis (evaluation of results, utility analysis). Trade execution is subdivided into two broad categories: agency/broker execution, when a system optimizes the execution of a trade on behalf of a client, and principal/proprietary trading, where an institution is trading on its own account. Historically, AT draws from cybernetics and control systems by introducing one or more feedback loops into the process; typically between posttrade analysis and pretrade analysis. Two stages that are typically part of such feedback loops are money management (asset allocation) and risk management.

A typical AT process may execute as follows: (1) during the pretrade analysis stage, the trading model acquires and abstracts the information about the current state of the financial markets and retrieves the most indicative (with respect to a predefined utility function) set of features of analyzed data; (2) such indicators are then utilized in the signal generation step, where the features are filtered by a set of rules that define circumstances when a trading signal should occur; (3) every time new market information arrives, it is streamed through the pretrade analysis process and signal generation process; (4) in an instance the signal generation process indicates that tradable opportunities exist in the model, a signal is sent to the trade execution element responsible for generation of a set of instructions to trade particular securities defined in the strategy; (5) to be able to evaluate the utility of the executed transaction, the algorithm performs a posttrade analysis.

Apart from the described stages, the more advanced strategies also utilize the money management and the risk management elements, where (6) the money management element allocates a specific amount of assets to each transaction with an aim to maximize a utility of the model, (7) while the risk management is responsible for minimizing risk of a loss and can also be expressed in a form of a utility function.

The pretrade analysis: This aims to help trading strategies to take the most optimal action at a specific time, given all the available information. Such analysis forms a first step in the creation of trading decisions in a model. It is a process of analysis of input data properties that leads to identification and quantification of a set of data features on the basis of which strategies can take their trading decisions. Such data features can be used to predict the future

behavior of financial instruments (that the algorithmic system is intended to trade), can be used to evaluate the levels of exposure/risk associated with the financial instruments, and can be also used to calculate potential costs associated with trading the financial instruments.

During this phase, the strategy may identify exploitable relationships between various assets (rather than just features of one of the assets alone) on the basis of which it may generate profitable transactions. Identification of comovement patterns, in securities the strategy trades, can be considered one type of data relationship that can be exploited to generate profit in AT.

The pretrade analysis is intended to improve indicativeness of the input data in AT. Raw input data are assumed to seldom carry a desired degree of information to support the decision-taking process. To improve the quality of data, the pretrade analysis stage may be further divided into preprocessing, feature selection, feature extraction, and feature cross-validation. The preprocessing stage ensures that the raw data are denoised, detrended, normalized, etc., before being considered further. The feature selection stage ensures selection of a set of relevant statistical features of the preprocessed data. The feature extraction stage ensures calculation and retrieval of the selected statistical features. Finally, the feature cross-validation stage ensures that only the most indicative features in a set are being used. Given such a list of strong indicators, a trading algorithm may start to form patterns out of the selected data features. The formed patterns may then be utilized in signal generation and other decision-taking processes. Formation of trade-exploitable patterns is often based on recognition of co-relations between different features of the input data.

Data mining is a method of discovery of useful information in data and is particularly applicable to this building block of an AT strategy. Finding nonobvious patterns in data sets is comparable to identification of indicative features relevant to trading strategies. The processes used for feature selection and cross-validation are often similar in both fields.

The signal generation: stage is another building block of AT strategy. It is responsible for generation of signals that define the behavior of a trading model, on the basis of the most indicative features identified during the pretrade analysis process. Trading strategies can be defined in terms of tasks (actions) they are to perform. Such tasks are constrained by the given information, or more precisely by the sets of features of the given information identified as influential during the pretrade analysis process. When a given set of constraints is met, it defines a signal for the execution of particular task of the strategy process, e.g., signal for issuing orders or maintaining, closing, and rebalancing market positions. Furthermore, the stage may also be responsible for the generation of support signals for, e.g., risk management and money management.

There are two approaches to the process of the definition of behavior of models that drive execution of risk management, money management, and trade execution blocks of AT strategies, in the next time horizon. These are the static and dynamic approaches. In the static approach, the behavior of a model is defined once, typically in a form of a set of logic rules with thresholds,

where the values of the thresholds can be optimized when the model no longer passes the fitness test of its utility function. In the dynamic approach, the definition of the behavior of a strategy is defined in a continuous manner during a learning process.

From the perspective of the signal generation stage, Artificial Neural Networks and Support Vector Machines are able to model complex, nonlinear relationships in data and therefore are particularly suitable for mapping data inputs (in the form of the most indicative features identified during the pretrade analysis process) to data outputs (in the form of signals that define the trading, the risk and money management behavioral policy). Furthermore, Evolutionary Algorithms are particularly suitable to the process of optimization of parameters of static strategies. In case of a dynamic control system, the machine learning approaches, i.e., the Reinforcement Learning methodologies, are particularly applicable.

The trade execution: stage can be viewed as a process that defines optimal execution of order instructions and transactions. After the trading signal has been generated, the risk and transaction costs estimated and accepted, and the assets allocated for a specific investment horizon, the execution functionality issues a set of order instructions (of a specific type and with required parameters) and manages the execution of orders (until successful or unsuccessful completion of transactions). For best execution and to minimize the market impact, advanced trading platforms provide a smart order routing functionality that allows the division of large instructions into small, more optimal orders.

The posttrade analysis: stage is responsible for generation and evaluation of results of the trading activity, such as the final execution price, the account profit & loss (P&L), and the remaining margin. The results of such analysis are then fed back to the pretrade analysis process and can be used to improve the future actions of the strategy.

The money management: (asset allocation) stage is a decision process responsible for the division and assignment of available funds. The process can be applied to a portfolio of orders, a portfolio of trading models managing the orders, a portfolio of accounts managing funds of the trading models, a portfolio of institutions managing the accounts, etc. Consequently, the process is applicable to multiple levels of the AT abstraction.

The risk management: stage in some respects is similar to the asset allocation approaches, where multiple levels of risk are identified and the risk features are monitored to influence behavior of trading algorithms according to the generated risk signals.

7 DRACUS

At this stage, the reader should have a clear view of the elements of experimental computational simulation environment for financial economics, including an overview of the structure of a potential AT model that can be submitted to

simulation. The following section will describe UCL's DRACUS environment, an example implementation of the abstract idea of experimental computational simulation environment.

DRACUS (DistRibuted Analytics, Control, & Utilities System) is an analytic environment, supported by a Hadoop-based computer cluster, designed to run a set of simulation classes on social, financial, and economic data. The environment is utilized in UCL for experiments in AT and Risk. UCL is now extending DRACUS to support a range of single and multiagent simulations for financial-economic experiments (e.g., simulations of systemic risk of banking systems).

The remaining part of the section describes how a DRACUS simulation is configured with use of a graph of models. It then describes how the internal components of DRACUS support experiments with such configured simulation and how different components interact together to make large amounts of independent simulations possible.

The key elements of DRACUS ecosystem (Fig. 2) include (a) a Software Development Kit (SDK) that provides (b) a set of predefined simulation templates containing default models that users can customize, (c) a general graph-based definition of simulation which involves various models being connected to for a simulation, (d) a simulation executor that forms a test bed for each instance of simulation and plays a role of information traffic control within a simulation, (e) a job dispatcher for optimal utilization of cluster capabilities, (f) data handlers, (g) compression, buffering, and conveying mechanisms, and (h) distributed data storage and collocated job execution.

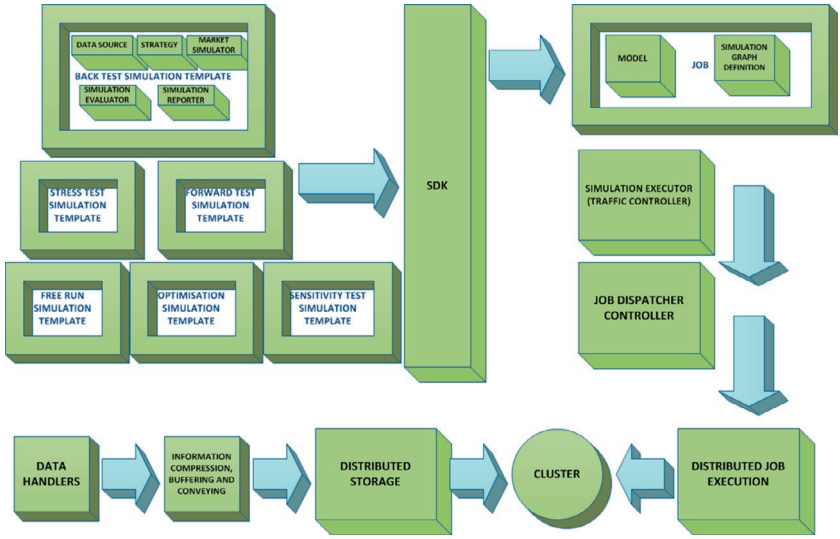


FIGURE 2 DRACUS ecosystem.

7.1 Model Implementation and Deployment

To effectively use the DRACUS environment, the users are provided with a library of abstract block models which they can customize. Particular instances of blocks can be then linked together according to user's needs. The environment provides two types of models. The first type enables the model to generate generic output in a form of time-series of information. The second type enables the model to both generate output time-series and consume input-time series. The DRACUS API defines a set of abstractions that enables the user to write custom logic of such models. When the logic is ready, the user can utilize the environment's SDK to deploy such model to the cluster. The model then waits to be organized in a simulation, together with other models. Each model committed to DRACUS has a unique ID that can be used in simulation definition.

7.2 Simulation Definition and Control

In order to start a simulation in the environment, the user needs to, firstly, define the elements of the simulation and their interconnections as part of the simulation. This may be perceived as a graph containing, but not restricted to a data source, the model, or models that are to be simulated, an evaluator, and a reporter. Such graph, when fully described and parametrized, can then be deployed to the environment via the SDK. After deployment, the user is able to control the state of the simulation by programmatically issuing commands to the environment. This enables to start the simulation, stop it, pause it, or restart it. The SDK also enables the user to check the progress of the simulation, or in the instance where the simulation finished, to retrieve the summary report.

The environment is highly customizable, enabling the user to incorporate (as part of the simulation definition graph) either the default versions of the above described models or custom versions. This may, for example, enable the user to have a custom version of the evaluator module, with its own simulation evaluation statistics, or a custom version of data source with a specific way to generate, e.g., synthetic data.

7.3 Simulation Executors

Every simulation in the environment can be described as an acyclic directional graph of nodes (where directional edges describe the flow of information from one node to another and nodes can be either roots, with only outflow edges, or normal nodes with both inflow and outflow edges). This representation provides a twofold benefit, it enables easy configuration for either single or multiagent simulation execution, and it is an extremely flexible and reconfigurable way of defining simulations. In order to execute such simulation in the cluster, the graph requires a "simulation bed" that will allow the information to be sent and received according to the definition of the graph, and even more importantly that

will enforce the flow of time only in one direction. Such simulation bed, called executor, can be considered an information traffic control during the course of the simulation.

The DRACUS environment currently supports two types of executors: a single-threaded executor and a multithreaded executor, with one thread per each node. To enable communication between the nodes, each leave is represented as a blocking queue, with the node threads reading and writing to/from it.

7.4 Job Distribution in the Cluster

All simulations committed by the user to SDK (to be executed in the DRACUS system) reach the job manager at the cluster entrance point, which queues simulations prior to dispatching them to a particular node on the cluster. The role of the job manager is to ensure optimal utilization of the cluster so that all the simulations can benefit from most of the available resources. This includes management of available processing power, memory and access to stored data. Given that most simulations are designed to run on long time-series of data, the manager benefits from MapReduce capabilities of the cluster to optimally locate jobs with data.

7.5 Connectivity Engine and Compressed Data Transition

An ability to work with large amounts of data often requires wide range of data sources which then needs to be aggregated. Apart from accessing new sources of data, the system also utilizes conveyor functionality that enables fast transition of data from its source to a 24-h buffer. The buffer is a central, persistent data point that can be then easily accessed by individual data readers of the environment that stores the data to its final destination. This is particularly important in the instance of node failures (during, e.g., writing to the aggregation storage space) in the cluster and aims at preventing data loss during individual cluster node downtimes. Furthermore, since large amounts of data are transferred every day, the conveyor functionality enables data compression and data clustering during transition, which further increases transfer speeds.

7.6 Data Aggregation in the Environment's Cluster

In order to perform analytic tasks in the environment, its users need large amounts of data in the system. For that purpose, the environment utilizes a set of connectors to stream live information from its source as it appears. Such streams are aggregated into larger chunks of data, compressed and stored at high-speed in a 24-h buffer facility hosted on one of the servers surrounding the environments cluster. Aggregation elements of the cluster are then able to retrieve the necessary information from the buffer and store it at their own speed

to a particular cluster node. DRACUS uses Hadoop Distributed File System (HDFS) as a main storage engine.

Stored data can be then accessed in simulations via data source abstractions. Each data source can be parametrized to retrieve a specific period of time of a selected information item and stream through the simulation until exhaustion of the data source. Thanks to the MapReduce capabilities, the executed simulations are optimally collocated with the necessary data.

8 SUMMARY

Social scientists and professionals need environments that will allow them to efficiently perform large-scale experiments and analytical tasks rather than simply store and search through large volumes of data. Consequently, experimental computational simulation environments that logically extend functionalities of BDA (to enable modeling and simulation of, e.g., financial economics phenomena) are being now extensively adapted. Such Environments are predominantly designed for testing, optimization, and monitoring of algorithms running in virtual or real mode and can complement academic efforts to organize publicly available data for research. One can identify a set of simulation templates suitable for modeling with socioeconomic-financial data in such Environments. Furthermore, effective large-scale analytics and simulation require sufficient software infrastructure to support the intended statistical tasks. When considering the experimental computational simulation environments from the perspective of simulation logic, the market simulators are central to such financial economics simulation. Finally, AT can be perceived as an interesting case study for socioeconomic-financial modeling.

REFERENCES

Ainsworth, M., 2009. Market data and business information two peas in a pod? Business Information Rev. 26 (2), 81–89.

Andrade, H., Gedik, B., Wu, K.L., Yu, P., 2009. Scale-up strategies for processing high-rate data streams in system S. In: ICDE'09. IEEE 25th International Conference on Data Engineering, pp. 1375–1378.

Ang, A., Timmermann, A., 2011. Regime changes and financial markets. Technical report. National Bureau of Economic Research.

Bullock, S., 2011. Prospects for large-scale financial systems simulation.

Chandramouli, B., Ali, M., Goldstein, J., Sezgin, B., Raman, B., 2010. Data stream management systems for computational finance. Computer 43 (12), 45–52.

Chua, C., Milosavljevic, M., Curran, J., 2009. A sentiment detection engine for internet stock message boards. In: Australasian Language Technology Association Workshop, p. 89.

Galas, M., 2014. Experimental Computational Simulation Environments for Algorithmic Trading. Ph.D. thesis, UCL (University College London).

Galas, M., Brown, D., Treleaven, P., 2012. A computational social science environment for financial/economic experiments. In: Proceedings of the Computational Social Science Society of the Americas.

Gangadharan, G., Swami, S., 2004. Business intelligence systems: design and implementation strategies. In: 26th International Conference on Information Technology Interfaces, pp. 139–144.

Johansen, A., Sornette, D., 2010. Shocks, crashes and bubbles in financial markets. Brussels Econ. Rev. (Cahiers economiques de Bruxelles) 53 (2), 201–253.

Khan, R., Quadri, S., 2012. Business intelligence: an integrated approach. Business Intell. J. 5, 64–70.

Luckham, D., 2008. The power of events: an introduction to complex event processing in distributed enterprise systems. Rule Representation, Interchange and Reasoning on the Web, p. 3.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., 2004. Mason: a new multi-agent simulation toolkit. In: Proceedings of the 2004 Swarmfest Workshop, vol. 8.

Mantegna, R.N., Stanley, H.E., 1999. An Introduction to Econophysics: Correlations and Complexity in Finance. Cambridge University Press.

McAfee, A., Brynjolfsson, E., 2012. Big data: the management revolution. Harvard Business Rev. 90, 60–68.

Subrahmanyam, A., Titman, S., 2012. Financial market shocks and the macroeconomy. Available at SSRN 2195184.

Treleaven, P., Galas, M., 2013. Algorithmic trading, flash crashes and it risk. In: Shojai, S., Feiger, G. (Eds.), Risk Management in Financial Institutions. Euromoney Books, pp. 83–123.

Treleaven, P., Galas, M., Lalchand, V., 2013. Algorithmic trading review. Commun. ACM 56 (11), 76–85.

Vincent, A., Armstrong, M., 2010. Predicting break-points in trading strategies with twitter. SSRN eLibrary.

Vovsha, A., Passonneau, O., 2011. Sentiment analysis of twitter data. ACL HLT 2011, 30.

Yoon, E., Guffey, H., Kijewski, V., 1993. The effects of information and company reputation on intentions to buy a business service. J. Business Res. 27 (3), 215–228.

Chapter 12

Terabyte-Scale Image Similarity Search

Diana Moise[*,‡,1], **Denis Shestakov**[†,‡]
[*] *Cray Inc., Switzerland*
[†] *Bright Computing, Netherlands*
[‡] *This work was carried out as part of the Post-Doctoral Researcher position at INRIA Rennes, France.*
[1] *Corresponding author: e-mail: dmoise@cray.com*

ABSTRACT
While the past decade has witnessed an unprecedented growth of data generated and collected all over the world, existing data management approaches lack the ability to address the challenges of Big Data. One of the most promising tools for Big Data processing is the MapReduce paradigm. Although it has its limitations, the MapReduce programming model has laid the foundations for answering some of the Big Data challenges. In this chapter, we focus on Hadoop, the open-source implementation of the MapReduce paradigm. Using as case study a Hadoop-based application, i.e., image similarity search, we present our experiences with the Hadoop framework when processing terabytes of data. The scale of the data and the application workload allowed us to test the limits of Hadoop and the efficiency of the tools it provides. We present a wide collection of experiments and the practical lessons we have drawn from our experience with the Hadoop environment. Our findings can be shared as best practices and recommendations to the Big Data researchers and practitioners.

Keywords: Big Data, Hadoop, MapReduce, Image search, Multimedia retrieval, Smart deployment, SIFT, HDFS, Hadoop deployment, Hadoop configuration, Hadoop performance, Map waves

1 INTRODUCTION

In the past several years, Big Data has become ubiquitous in our society. The scale at which data is generated has grown greater than ever before. According to the 2011 Digital Universe study of International Data Corporation (IDC), the

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00012-5

information produced in 2011 all over the world has gone beyond 1.8 Zettabytes (ZB), marking an exponential growth by a factor of 9 in just 5 years. Such data explosion has enormously raised the bar for both industry and research: existing systems that acquire, store, and process the data collections do not possess the hardware and software capacities to accommodate the data. In this context, "Big Data" is becoming a hot term used to characterize the recent explosion of data. Big Data describes the unprecedented growth of data generated and collected from all kinds of data sources. This growth can be in the volume of data or in the speed of data moving in and out data-management systems. It can also be the growth in the number of different data formats in terms of structured or unstructured data.

A massive amount of data produced all over the planet belongs to the multimedia type. Continuously growing multimedia collections come from social networks: in 2012, Facebook reported that 300 million images were uploaded every day, accumulating to 7 petabytes of photo content every month (Gigaom, 2012). These huge streams of images coming from all over the Web render multimedia content a significant instance of Big Data. Multimedia collections match all the points in the "5V's" list. Big Volume is clearly a feature. Usually, there are more than a few ways to represent multimedia content, so here Variety goes. Due to numerous scenarios and use cases, Big Variability, Velocity, and Value are important factors in multimedia processing. For instance, in a copyright detection scenario, obtaining the right result is as important as the time it takes to get it.

This increase in the amount of multimedia data over the past years is partially an effect of the quality of information that is contained in the data, quality of content that is captured through a high number of dimensions. *High dimensionality* of data represents a major challenge in information retrieval. Among applications that require efficient high-dimensional image retrieval, we count face recognition (with a direct application in the security industry), medical image retrieval, and video retrieval (where the video is viewed as a sequence of frames).

Current state-of-the-art technologies developed for Big Data analytics can be applied to analyze extremely large multimedia datasets now available on the Web. MapReduce (Dean and Ghemawat, 2008) is currently the most efficient and popular tool for processing and analyzing large amounts of information using commodity machines. Although it has its limitations, the MapReduce programming model has laid the foundations for higher-level data processing and is definitely a first step to answering many Big Data challenges. In this chapter, we focus on the Apache Hadoop project (http://hadoop.apache.org/), an open-source implementation of the MapReduce paradigm. By the means of a specific multimedia application, i.e., image similarity search, we present our experiences with Hadoop when processing terabytes of data. The scale of the data and the application workload have tested the limits of Hadoop and the efficiency of the tools it provides. Our findings could be useful for

researchers and practitioners working on such applications as regular image search, general image categorization, product image search, object recognition (e.g., detecting texts, logos, vehicles, people, structures, etc.) in image or video collections/streams, face recognition, medical image analysis, and satellite image processing.

Our first contribution is adapting the Hadoop environment to multimedia retrieval application workloads, where default Hadoop settings are no longer efficient. We list the parameters and values that delivered best performance for us. The second contribution is addressing the cluster heterogeneity problem by proposing a platform-aware configuration of the Hadoop environment. The third contribution is introducing the tools, provided by the Hadoop framework, useful for a large class of application workloads, in which a large-size auxiliary data structure is required for processing main dataset. Finally, we present a series of experiments conducted on a very large image dataset (the biggest among reported in the literature). In contrast to a number of Big Data reports describing experiments with either small or synthetic data, we report our findings and lessons obtained from a real-life use case, namely, finding images potentially violating copyrights in terabyte-sized collections.

We structured this chapter as follows. Next section briefly lists the challenges of Big Data analytics and presents current trends in this area, with a focus on the Hadoop framework. Section 3 gives an overview of MapReduce-based high-dimensional indexing scheme we designed and implemented for the Hadoop runtime environment. Section 4 presents the results of running the application in the default Hadoop environment. How we achieved further optimizations through various mechanisms is discussed in Section 5. Section 6 concludes.

2 BIG-DATA PROCESSING

In this section, we focus on Big Data challenges and the current trends in Big Data analytics.

2.1 Challenges

Scale. The need for efficiently managing rapidly increasing volumes of data has always been there, and it was addressed for the most part. Parallelism is the key to adjusting to the data size challenge. Whether it is achieved on a single machine through multiple computing units, or across many machines in a distributed fashion, large data sets are nowadays efficiently handled. However, in the Big Data context, scalability continues to be a challenge. Infrastructures such as clusters (notably HPC clusters), Grids, Clouds (Nimbus (Keahey and Freeman, 2008), Amazon EC2 (Jackson et al., 2010), OpenNebula, OpenStack (Wen et al., 2012)), and multicore servers with large memory represent a major step forward, but still have their limitations when accommodating Big Data.

Speed. In practice, the size of the data gives rise to real challenges, especially when considering the rate at which data needs to be ingested and analyzed. Performing actions such as data acquisition, storing, analyzing, and visualizing becomes a problematic issue that needs dedicated approaches. It is obvious that the volume of data is the most immediate challenge for conventional data-management and processing frameworks. A fundamental change is required in the architecture of systems capable of delivering high throughput.

Heterogeneity. Big Data is also about the growth in the number of different data formats in terms of structured or unstructured data. Usually, a preprocessing step is required to bring the data to a form that is understood by the processing framework. Efficient representation, access, and analysis of semistructured data require further work.

2.2 Trends

MapReduce (Dean and Ghemawat, 2008) is currently the most widely adopted tool used for Big Data analytics. There are several implementations of the MapReduce paradigm, e.g., Apache Hadoop (Bhandarkar, 2010), Twister (Ekanayake et al., 2010), and Disco (Mundkur et al., 2011), on top of which specialized higher-level frameworks were developed.

Computer vision-wise, the Mahout (Owen et al., 2011) project uses Hadoop to provide an extensive list of machine learning algorithms. In the area of content-based image retrieval, there are systems that can handle several million images (Jégou et al., 2012; Lejsek et al., 2009), billions of descriptors (Jegou et al., 2011; Lejsek et al., 2011), or address Web-scale problems (Batko et al., 2010; Douze et al., 2009). Lastly, the ImageTerrier project (Hare et al., 2012) provides a set of libraries for the Hadoop-based multimedia platforms.

The MapReduce programming model, its implementations such as Hadoop, and all the aforementioned higher-level frameworks belong to the *batch processing* approach for analyzing Big Data. However, in many contexts, Big Data is so *fast* that most batch frameworks are simply not suitable. Trying to overcome the shortcomings of batch processing with respect to ingesting data in real time and providing results in real time, current Big Data solutions include *stream processing* frameworks such as Storm (Storm, 2012), S4 (Neumeyer et al., 2010), and Spark Streaming (Zaharia et al., 2013). Recently, a generic architecture that supports processing of big and fast data has been proposed, under the term of *lambda architecture* (Architecture, 2013). The goal of the lambda architecture is to support workloads requiring low-latency I/O, while guaranteeing fault tolerance. There are already quite a few systems that implement the lambda architecture with the means of batch frameworks (Hadoop, Spark), stream frameworks (Storm, Spark Streaming), and query engines.

2.3 Hadoop

The MapReduce programming model has been implemented by the open-source community through the Hadoop project (Hadoop, 2007). Maintained by the Apache Foundation and supported by Yahoo!, the Hadoop project has rapidly gained popularity in the area of distributed data-intensive computing. The core of the Hadoop project consists of the MapReduce implementation and the Hadoop Distributed File System (HDFS).

The Hadoop MapReduce framework (Bhandarkar, 2010) was designed following Google's architectural model and has become the reference MapReduce implementation. The architecture is tailored in a master–slave manner, consisting of a single master *jobtracker* and multiple slave *tasktrackers*. The jobtracker's main role is to act as the task *scheduler* of the system, by assigning work to the tasktrackers. Each tasktracker consists of a number of available *slots* for running tasks. Every active map or reduce task takes up one slot; thus, a tasktracker usually executes several tasks simultaneously. When dispatching "map" tasks to tasktrackers, the jobtracker strives at keeping the computation as close to the data as possible. This technique is enabled by the data-layout information previously acquired by the jobtracker. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes closer to the data (belonging to the same network rack). The jobtracker first schedules "map" tasks, as the reducers must wait for the "map" execution to generate the intermediate data. Apart from data splitting and scheduling responsibilities, the jobtracker is also in charge of monitoring tasks and dealing with failures.

The HDFS (Shvachko et al., 2010) was built with the purpose of providing storage for *huge files* with *streaming data access patterns*, while running on clusters of *commodity hardware*. HDFS implements design concepts commonly used by distributed file systems: data is organized into files and directories; a file is split into fixed-size blocks that are distributed across the cluster nodes. The blocks are called *chunks* and are usually of 64 MB in size (this parameter specifying the chunk size is configurable). The architecture of HDFS consists of several *datanodes* and a single *namenode*. The nodes in a HDFS cluster that store the chunks are called *datanodes*. A centralized *namenode* is responsible for keeping the file metadata and the chunk location. HDFS handles failures through chunk-level replication (three replicas are kept by default). When distributing the replicas to the datanodes, HDFS employs a rack-aware policy: the first copy is always written locally, the second copy is stored on a datanode in the same rack as the first replica, and the third copy is shipped to a datanode belonging to a different rack (randomly chosen). The namenode decides and maintains the list of datanodes that store the replicas of each chunk.

3 APPLICATION WORKLOAD (DISTRIBUTED INDEXING + SEARCHING)

Our workload is based on the scalable, MapReduce-oriented, high-dimensional indexing scheme initially presented in Moise et al. (2013a). The indexing algorithm uses at its core a hierarchical unstructured quantization scheme, quite similar to the approach proposed by Nister and Stewenius (2006). In a nutshell, high-dimensional descriptors are clustered around randomly picked representative points that are hierarchically organized. The clustering process is done in two phases, detailed in Section 3.2. The first step is to choose how many clusters will be built and which are the centroids of each cluster (cluster *representatives*). A simple Java program chooses random points from the dataset to act as cluster representatives. Further, to be able to efficiently cluster points around these representatives, an *index tree* is built. This tree contains cluster representatives organized in a tree hierarchy (see Moise et al., 2013a for more details on how the tree is built). The second step is to use MapReduce and Hadoop to assign all the points in the data collection to the cluster representatives previously chosen. This is done by traversing the index tree and assigning each point to the closest (Euclidean-distance wise) cluster representative.

3.1 Dataset Preparation

Our datasets, the collections of high-dimensional descriptors, were converted to *Hadoop Sequence Files*. A SequenceFile is a special file format in Hadoop, utilized to efficiently store binary data and provide fast access to it. The benefits of using SequenceFiles are support for block compression (reducing data size on disk) and efficient implementation of *file seek* operation. In the case of our dataset, each raw descriptor becomes a sequence file record. A record in this file has a key (an image identifier) and a value (a high-dimensional descriptor). Hadoop scatters across many machines the blocks of the sequence file in its distributed file system, HDFS. Thus, the entire descriptor collection is stored as a set of HDFS sequence files; this set of sequence files also represents the input data to the applications described further.

3.2 Distributed Index Creation

To be able to index a huge collection of descriptors using MapReduce, we first perform a phase of building auxiliary data required by the distributed process. This preliminary phase, which we execute on a single machine and usually only once, consists of building the *index tree*, a hierarchically organized collection of cluster representatives. The tree is created by randomly picking from the descriptor collection C points, i.e., the representatives of the visual vocabulary dictionary eventually created. These representatives are organized in a hierarchy

of L levels for efficiency. Each point in the index tree will become a cluster representative; clusters will be created in the next phase, i.e., the MapReduce phase, when each descriptor in the dataset will be assigned to a single point in the index tree. The index tree is serialized to a file on disk and sent to all the nodes involved in the distributed index creation.

Creating an index from a very large collection of descriptors is a very costly process. Distributing it with MapReduce involves designing the map and reduce phase of the indexing application. Each map tasks receives at startup time (i) a block of input data (a chunk of a Hadoop sequence file) and (ii) the file containing the index tree. The mapper further assigns the descriptors in its data block to the correct representative discovered by traversing the tree. The resulting assignments are sent to reduce tasks that write to disks high-dimensional descriptors grouped by cluster identifier. This eventually creates one or more *indexed files* which contain clustered high-dimensional descriptors. The number of such files is defined by the number of reduce tasks.

The indexing application is highly *data-intensive*, both in the map phase when reading the descriptors to assign them to cluster representatives and in the reduce phase when writing the clusters to disk. The application also generates a substantial amount of output data, slightly larger than the input set. The map stage is *CPU-intensive* as well, by performing distance calculations for every input record. The shuffling step intensively utilizes the network, as the intermediate data is sent from mappers to reducers. Finally, distributed indexing is a *long-running* application on account of its data- and CPU-intensive nature.

3.3 Distributed Search

Using the MapReduce paradigm to search through indexed data enforces some constraints on which type of queries one can execute. Since MapReduce is a batch processing model, employing this paradigm to answer a single query is not efficient: MapReduce will read all the data from disk and match it against the query. To better adjust to the paradigm, we focus on answering a batch of multiple queries, which is a scenario representative for many applications in different domains. Among such applications are regular image search (e.g., image search systems like Google Images or TinEye), general image categorization, product image search, object recognition (e.g., detecting texts, logos, vehicles, people, structures, etc.) in large collections of images or videos, face recognition, medical image analysis, satellite image processing, etc.

Search phase is geared toward throughput as it processes very efficiently large batches of queries, typically 10^4–10^7 query descriptors. The search also requires a preliminary step, the creation of a *lookup table*, where all query descriptors of a batch are grouped according to their closest representative discovered from traversing the index tree. This lookup table is written to the local

disk of all the nodes that will perform search. The MapReduce implementation of the search process is detailed further. Each map task receives (i) a block of data from one of the previously created index files and (ii) the file containing the lookup table. The mapper processes only the descriptors in its assigned chunk of data, which are relevant for the queries. Distance calculations are computed for those descriptors and queries assigned to the same cluster identifier. *k*-nn results are eventually emitted by the mappers and then aggregated by reducers to create the final result for the query batch.

Our implementation of the distributed search (Shestakov et al., 2013) is *data-intensive* in the map phase where indexed files are read from disk. Unlike the indexing application, the distributed search is *short-running* since only a part of the indexed descriptors are compared to the queries. This also renders the search application *unbalanced*, as the workload of each mapper depends on whether the chunk they process includes clusters required by the queries. Another aspect that differs in this application as opposed to the index creation refers to the data shuffled by mappers and generated by reducers, which is insignificant in size, compared to the input data.

3.4 Dataset

In our experiments, we used the dataset provided by one of our partners in the Quaero project, http://quaero.org/. The dataset consists of around 30 billion SIFT descriptors (approximately 4 terabytes of data) extracted from 100 million images harvested on the Web. Besides using this entire collection, we also used a subset of it containing 20 million images, i.e., 7.8 billion descriptors (about 1 terabyte). To the best of our knowledge, these datasets are among the largest collections presented to the multimedia community. Details on evaluation of indexing quality and the quality results themselves can be found in Moise et al. (2013a).

3.5 Observations

Several comments are in order. First, the indexing and searching schemes briefly described above perform approximate *k*-nn searches, trading result quality against dramatic response time improvements. Second, the schemes use quite a lot of *auxiliary* data, both during the creation of the index (this is the index tree) and during the search phase (this is the lookup table). Particularly, the index tree has to be loaded by each *Map*-task when processing *each* block of (not yet indexed) descriptors at assignment time, and the lookup table is required by each *Map*-task to process *each* block of (clustered) descriptors at search time.

While our experiences pertain to these two applications, their characteristics are general enough so that our findings are easily applied to many other Big Data analytics problems (Moise et al., 2013b).

4 HADOOP IN PRACTICE

Although the Hadoop framework is now widely adopted, processing and analyzing terabyte-sized datasets require to reconsider many of its design, implementation, and execution aspects. The Hadoop framework has been shown to scale and efficiently process hundreds of terabytes of data on thousands of machines: e.g., in Yahoo!, Facebook, etc., but only a few works actually reported Hadoop at such large scales, data- and platform-wise. In fact, one can hardly find real-life recommendations on Hadoop parameter configuration or suggestions on Hadoop techniques to be used for handling such amounts of data. The experiments presented further were conducted based on real-life problems and on real datasets. In contrast to the typical Hadoop benchmarking, real applications raise complex problems and challenges.

We present here our experience with Big Data analytics using Hadoop in a grid environment. Based on the challenges and problems we were faced with, we recommend best practices (Hadoop cluster configuration tuning, tools) for dealing with them.

The application workload described in Section 3 was integrated into a framework for ease of use and automation. We developed this framework to be able to configure the compute environment, deploy a Hadoop cluster, transfer input data to HDFS from external storage, run indexing and searching jobs and optionally, retrieve the results, and analyze logs.

4.1 Experimental Platform

To reach the point where we could process terabytes of data on a Hadoop cluster instance, we have performed a large series of experiments on the Grid'5000 (Jégou et al., 2006) platform. The Grid'5000 project is a widely distributed infrastructure devoted to providing an experimental platform for the research community. The platform is spread over nine geographical sites located on French territory. We conducted our experiments on machines belonging to three clusters of the Rennes site of Grid'5000. The cluster configurations are shown in Table 1. A Hadoop instance is deployed on these nodes as follows: the namenode and jobtracker are each launched on a dedicated node, while the datanodes and

TABLE 1 Cluster Configurations

Cluster	#Nodes	#CPU@Freq	#Cores /CPU	RAM	Local Disk
Cl_1	64	2 Intel@2.50 GHz	4	32 GB	138 GB
Cl_2	25	2 Intel@2.93 GHz	4	24 GB	433 GB
Cl_3	40	2 AMD@1.70 GHz	12	48 GB	232 GB

tasktrackers are co-deployed on the remaining nodes; we also allocate a separate machine to act as Hadoop client for job submissions.

Although this automated process is necessary to facilitate experimentation on Grid'5000, it is not sufficient to overcome the challenges of running at large scale, platform, and data wise. While our experiences pertain to the Grid'5000, the challenges we have encountered are also relevant to grid environments in general. We typically faced the three following problems:

- *Heterogeneity of resources*. Resource heterogeneity in a grid becomes a significant limitation when setting up the Hadoop environment. Since Hadoop settings can only be specified globally and not in a per-node manner, the Hadoop deployment is configured according to the least equipped node, at the expense of wasting resources on more out-fitted nodes. To be specific, heterogeneity of resources across clusters on the grid influences the number of map/reduce slots per tasktracker and the amount of RAM memory allocated to each task; these parameters have to be set to the lowest available values, which degrades jobs' performance.
- *Node failures*. Node failures represent the daily norm in grid environments. Even though Hadoop is designed to cope with failures in a transparent manner, machine deaths can severely impact the whole deployment. On Grid'5000, we experienced from one to five node failures during a 60-h run, some failures requiring a complete redeployment to exclude the failed nodes. The worst failure outcome is losing the data on the machine. To avoid this, we replicated data twice or thrice. However, this is not always possible for very large datasets; factors such as storage and replication time add up to substantial costs.
- *Deployment overhead*. The deployment overhead for each experiment is substantial when running at large scales. Since the grid is a shared tool, it operates using reservations that allow users to employ resources for a certain time slot. After the reservation expires, all deployment data and setup are deleted. Consequently, creating the experimental environment, setting up the Hadoop cluster, and making the data available in HDFS have to be repeated for every experiment.

4.2 Experiments with Default Hadoop Settings

In a first round of experiments, we performed indexing with the default Hadoop configuration parameters. This gave us a baseline for assessing any improvements we could further obtain through various means.

What we refer to as the default Hadoop deployment is the basic configuration shipped with Hadoop, to which we applied some adjustments, based on the guidelines provided for dealing with Big Data. The first obvious adjustment was to set the map and reduce slots to accommodate the platform capacities: eight slots for mapping and two slots for reducing per node. We set these values

based on our clusters computing power and following the recommendations in Hadoop's tutorial. The second parameter we modified is the HDFS chunk size; the default value of 64 MB is not suitable for Big Data, as it entails larger pressure on the namenode (because of filesystem metadata) and a large number of map tasks be executed. After some experiments with varying chunk sizes, we discovered the right chunk size for our workload to be 512 MB. At the level of HDFS, we kept the default replication factor of 3 for the input data. The replication factor is a crucial parameter for performance, not only because it enables fault tolerance but also because it favors local execution of map tasks, minimizing the amount of data read remotely. The output data, however, was written only once, to avoid the substantial overhead of writing replicas remotely. Given the size of our dataset, this cost is considerably high.

With these three basic configurations, indexing 1 TB of data on 106 nodes took 95 min, while 4 TB on 100 nodes were indexed in 10 h in the same environment (Table 2). There are several reasons to explain the significant difference in runtime between indexing 1 and 4 TB. In addition to the obviously larger number of tasks to be executed, there is also a larger computational work done by each map task. The tree of representatives built for indexing 4 TB is a lot larger than the one covering 1 TB: 1.8 GB as opposed to 461 MB. The size of the index tree has two consequences. First, each mapper has to perform more distance calculations to assign each point to the closest cluster. Second, the index tree takes up more RAM in each map slot; thus, we had to reduce the map slots to 4 per node.

4.2.1 Analysis of Job Run

We focus further on understanding the indexing process in both default and tuned configurations, by looking into each of the map and implementation reduce phases. Table 3 shows some statistics on building the index on 1 TB of data on 106 nodes with both default and tuned Hadoop configurations. The number of map tasks to run is defined by the number of chunks in the input data. In the case of processing 1 TB of input data with a chunk size of 512 MB, Hadoop creates 2050 map tasks to be executed by the tasktrackers. Line 1 of Table 3 shows larger numbers of total map tasks that were actually run by Hadoop because these numbers also include the failed tasks. Lines 2 and 3 of the table show that

TABLE 2 Indexing Time

	Nodes	Default Hadoop	Tuned Hadoop
1 TB	50	202 min	174.7 min
	106	95 min	75 min
4 TB	100	10 h 1 min	8 h 27 min

TABLE 3 Map Statistics

	Default Hadoop	Tuned Hadoop
#Total map tasks	2456	2357
#Remote map tasks	431	212
#Failed map tasks	406	307
Duration of average map task	23 min	17 min

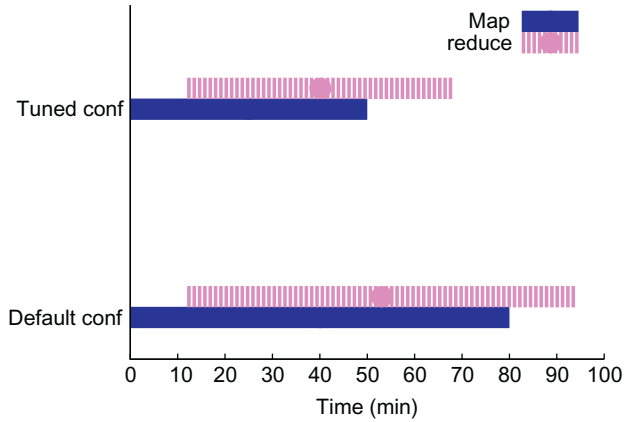


FIGURE 1 Map and reduce phase durations for indexing 1 TB.

tuning Hadoop's parameters reduces the number of maps that read data remotely and also the number of failed map tasks. The last line of Table 3 gives the average time of all successful map computations. By adjusting the map-side parameters, such as the size of the map output buffer, the sort factor, and compression of map output data, the tuned Hadoop framework executes map tasks faster than the default configuration.

In Fig. 1, we plot the overall runtime of the same 1 TB experiment, breaking it down into the execution time of the map and reduce phases. The impact of tuning Hadoop parameters can be observed on both map and reduce steps (shuffling is a part of the reduce phase).

4.2.2 Analysis of Map Waves

For the indexing process, the map phase is both data- and computationally intensive. Understanding the way Hadoop executes this phase is essential for optimizing the overall process. To index 1 TB of descriptors on 106 nodes, Hadoop creates 2050 map tasks and schedules them on the 848 map slots made available by the 106 tasktrackers (configured with 8 map slots each). The map

tasks are executed in *waves*: two complete waves of 848 maps and a final shorter wave of 354 tasks.

The number of waves to fully execute the map phase is adjustable by changing the number of map tasks, i.e., by changing the chunk size in HDFS. Discovering an optimal number of map waves is not, however, a significant goal. The mapper waves are rather means for assessing how well a Hadoop job is mapped to the platform. For example, a single incomplete wave would mean that the platform is being underutilized; on the other hand, having a large number of waves of short-running map tasks is also not recommended due to the overhead implied whenever Hadoop launches a JVM to execute a task.

The time progress of map phase for both default and tuned runs is shown in Fig. 2, where two plots, top for default run and bottom for tuned run, are mirrored along the *X*-axis (time in seconds). Thus, map phases for default and tuned runs can be directly compared. In both runs, the first wave of map tasks is launched in parallel within first 3 s. One can observe that performance of default and tuned runs during the first map wave is almost identical. However, during second and third waves, this pattern changed as duration of map tasks in default run significantly surpassed of those in tuned run. For convenience, we depict two lines following the average time of map task for each wave in both runs.

Since mappers run on nodes with different performance, there is some variance in map duration times which eventually leads to the degradation of mapper waves. Also, the spikes show that some map tasks are significantly slower than the rest of the wave. This is an effect of the default Hadoop settings that do not accommodate the large amount of data being processed. Some tasks have to spill their output to disk and wait for network to send the data to

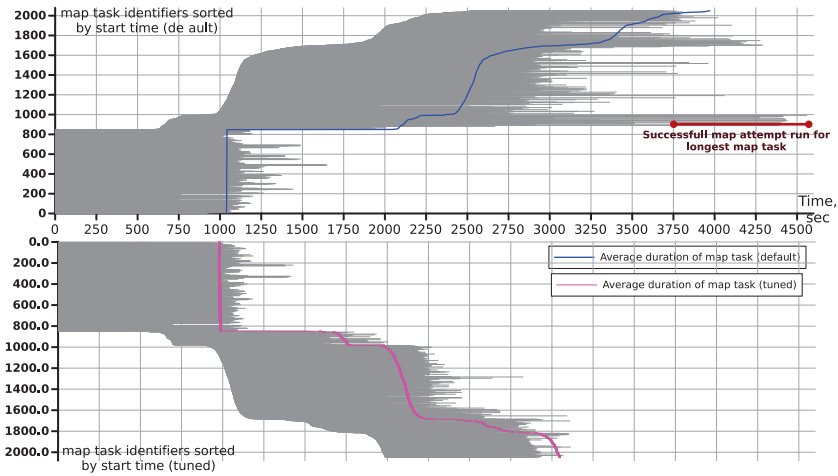


FIGURE 2 Time progress of map tasks for indexing 1 TB using default (top) and tuned (bottom) configurations.

the reducers. Adjusting low-level Hadoop parameters (increasing the size of memory buffers, compressing the map output, etc.) can prevent this behavior, as seen in the bottom plot of Fig. 2.

We also discovered that many of the spikes in Fig. 2 (top) describe tasks that failed and were reexecuted one or more times. The failure of the tasks is caused by using the default configuration that does not correspond to the scale of the experiment. Hadoop automatically reexecutes failed tasks, with each such task consisting of possibly several failed *attempts* and one successful final one. Many of such long-running tasks are seen in the top plot of Fig. 2 (second wave). To illustrate, we depict the last/successful attempt for the longest map task in Fig. 2 (top).

4.2.3 Underutilization in Reduce Phase

Lastly, we analyzed the performance of reduce tasks in indexing 1 TB (default run). With eight reducers configured per node, the maximum amount of reduce tasks is 848. Assuming the start of reduce phase at 0 s (then the indexing job started at −716 s), all 848 reducers were running within the next 26 s. At 3926 s (when the fastest reducer finished), the number of running reducers started to decrease from 848 to one reduce task at 4976 s. Figure 3 depicts the overall number of running reduce tasks at the end of reduce phase, from 3800 to 4976 s. As one can see, during the time frame 3926–4976 s, resources of nodes with

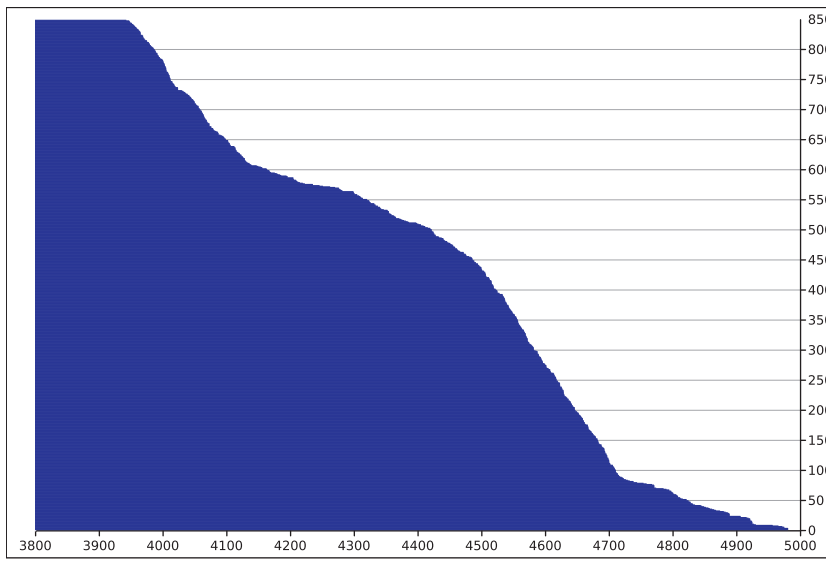


FIGURE 3 Number of running reduce tasks during indexing 1 TB (default run) at the end of reduce phase.

finished reduce tasks were underutilized. For example, the last 50 reduce tasks (finish time within 4809–4976 s) were run on just 7 nodes, leaving the rest 99 nodes completely idle.

5 LARGE-SCALE HADOOP

5.1 Adjusting to the Data Size

Understanding the application workload and accurately assessing the scale at which the application runs are key steps to achieving optimal performance. We present here a brief analysis of our workload, which helped us pinpoint the issues to further improve.

Indexing 4 TB of data with Hadoop involves storing 16 TB of data (input replicated three times plus output written once), shuffling 4 TB of intermediate data, and executing approximately 8000 map tasks. Platform-wise, we dispose of 100 nodes, i.e., 800 map slots (400 for 4 TB) and 200 reduce slots, organized in 3 clusters connected to the same network switch. Based on these observations, several optimizations are in order. Since a substantial amount of data is shuffled from the mappers to the reducers, compressing the map output can be beneficial, as it also reduces network usage. Rack-awareness is not useful in this case, since the nodes connect to a single switch; however, if it is the case, configuring rack-awareness according to the network topology is necessary as it allows Hadoop to optimally place tasks on nodes for execution.

Table 4 sums up these optimizations by listing the Hadoop parameters and the values we used in the indexing process. As mentioned above, enabling map output compression reduced the amount of shuffled data by 30% (from 1 to 740 GB). Other low-level parameters refer to the shuffling phase, when the data

TABLE 4 Hadoop Configuration Tuning

Parameter	Default Value	Tuned Value
#Map slots/tasktracker	2	8
#Reduce slots/tasktracker	2	8
Input data replication	3	3
Output data replication	3	1
Chunk size	64 MB	512 MB
JVM reuse	Off	On
Map output compression	Off	On
Reduce parallel copies	5	50
Sort factor	10	100
Sort buffer	100 MB	200 MB
Datanode max. receivers	256	4096
Namenode handlers	10	40

is sorted on the mapper's side, copied to the reducer and then merged and sorted on the reducer node. The last two lines of Table 4 show options configured on the HDFS nodes: the maximum number of chunk requests (read/write) each namenode can simultaneously handle, and the number of connections the namenode can serve at a time.

Table 2 shows a comparison of the indexing time using the default Hadoop configuration as well as the optimized one. On the same testbed and with the same slots for mapping and reducing, the customized Hadoop parameters deliver a much faster indexing time. The results in Table 2 show that setting the right parameters for the Hadoop framework can drastically improve Hadoop's performance. Tuning the parameters of the Hadoop framework is a rather complex task that requires good knowledge and understanding of both the workload and the framework itself. Given the performance gains obtained, tuning the parameters listed in Table 4 is worthwhile; nevertheless, the values delivering best performance are highly dependent on the workload, and thus, tweaking them in order to discover the best values is advisable.

5.2 Hadoop on Heterogeneous Clusters

Large-scale processing of Big Data is most commonly performed in a heterogeneous environment. Resource heterogeneity can impact the performance since most data-processing systems are designed for homogeneous platforms (Ahmad et al., 2012). This is also the case for the Hadoop framework: the configuration settings can be specified only globally, not in a per-cluster manner. The consequence is that the Hadoop deployment is configured according to the least equipped node, at the expense of wasting resources on more out-fitted nodes. Parameters such as the number of map/reduce slots per tasktracker and the amount of RAM memory allocated to each task have to be set to the lowest available values. The advantage of having a single configuration for the whole platform is that it is easy to manage; the considerable downside, however, is that the platform is underutilized.

To fully exploit the computing power of our three cluster environments, we developed a simple mechanism that configures tasktrackers parameters on each cluster based on the cluster's capabilities. This mechanism consists of three steps:

- deploy Hadoop on all the nodes using the global configuration settings, i.e., adjusting to the lowest available values;
- create cluster-specific configuration files; and
- restart tasktrackers with the new configuration files.

Specifically, considering the clusters description in Table 1, we deployed Hadoop with 8 map slots and 2 reduce slots per node and then restarted tasktracker processes on Cl_3 to use 20 map and 4 reduce slots. With this platform-aware Hadoop deployment, indexing 1 TB of data on 107 nodes

finished in 65.4 min, 30 min faster than the default Hadoop run and 10 min faster than the tuned Hadoop run (see Table 2).

Figure 4 shows the map tasks in time when indexing 1 TB with the default Hadoop deployment (Fig. 4a) and when configuring Hadoop per-cluster (Fig. 4b). The figures basically depict how the map phase progresses in time. There are about 2050 map tasks to be launched on the 100 node platform. With the default Hadoop run, we could set 8 map slots per tasktracker, giving us a total of 856 slots for mapping at a time. As Fig. 4a shows, the first wave on map tasks is launched in parallel. Since mappers run on nodes with different performance, there is some variance in map duration times which eventually leads to the degradation of mapper waves. Also, the spikes show that some map tasks are significantly slower than the rest of the wave. This is an effect of using the default Hadoop settings that do not accommodate the large amount of

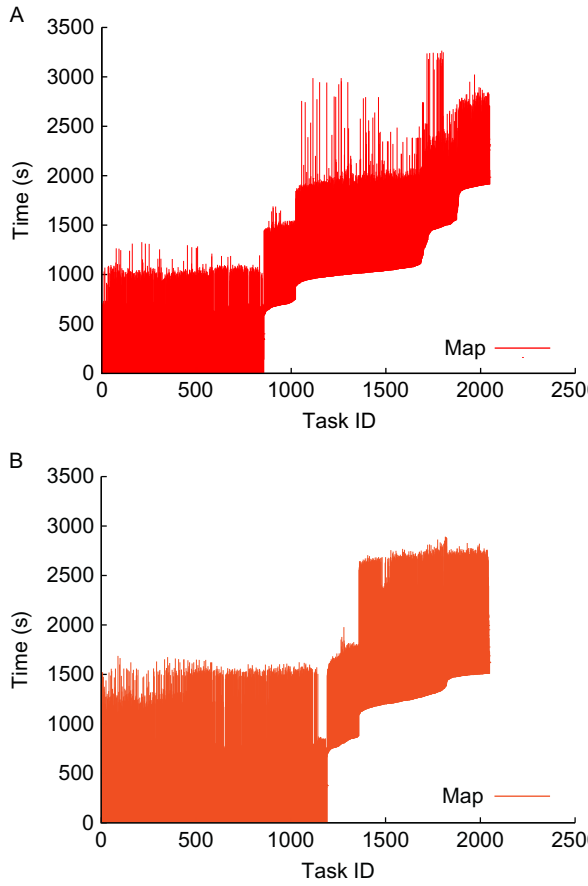


FIGURE 4 Time progress of map tasks when indexing 1 TB on 107 nodes. (a) Default Hadoop. (b) Platform-aware Hadoop.

data being processed. Some tasks have to spill their output to disk and wait for network to send the data to the reducers. Adjusting low-level Hadoop parameters (increasing the size of memory buffers, compressing the map output, etc.) can prevent this behavior, as can be seen in Fig. 4b. In addition, configuring each cluster according to its true capabilities provides 1192 slots for mapping in parallel, and finally, faster execution time.

5.3 Dealing with Large-Size Auxiliary Data

As discussed in Section 3, both indexing and searching applications use auxiliary information to process the input data. In the case of the distributed index creation, the mappers load the tree of cluster representatives and traverse it for every input point to assign it to the closest cluster. Whereas the search application is concerned, the auxiliary data consists of the batch of query descriptors to be processed. In both cases, the additional information can amount to significant sizes: the index tree corresponding to 1 TB of data is 461 MB in size, while for 4 TB of data, 1.8 GB of auxiliary data is required. We focus here on tools we found useful in managing auxiliary data.

5.3.1 Multithreaded Mappers

Loading the same piece of information in every mapper is a requirement commonly encountered in MapReduce applications. The way Hadoop addresses this issue is to allow users to adjust the RAM dedicated to each mapper so that it accommodates the size of the resource to load. However, by doing so, one has to take into account the machine's capacity, i.e., the total available memory: having more RAM per mapper means having less map slots per node, less map operations executed simultaneously, and finally, longer execution time. A way to avoid wasting CPU cores while still allocating more RAM is to make use of *multithreaded mappers*. Implemented in Hadoop, the multithreaded mapper spans a configurable number of threads, each thread executing map tasks as a normal mapper would. The threads of a mapper share the same RAM memory; thus, auxiliary data can be loaded only once and read simultaneously by multiple threads during the map phase. The map computations are executed in parallel, each thread working on different (key, value) pairs. By setting the right number of threads per mapper, the CPU capacity can be fully utilized. The downside of this mechanism is that multithreaded mappers use synchronization when both reading the input data and then writing the map output: the threads synchronize when reading each (key, value) pair from the input chunk of data allocated to the mapper; running the map function on that pair is done in parallel by all the threads; however, writing the output of each map operation to the same map output buffer is again synchronized.

The optimal configuration of map slots and number of threads per each slot strongly depends on the application workload. Obviously, the combination of

map slots × threads per slot has to fully utilize the machine's processing power. The synchronization steps in the multithreaded mapper implementation incur an inherent overhead when using this mechanism over the normal Hadoop mapper. Thus, the performance gain of using multithreaded mappers is only partially dependent on the workload and the configuration.

Our indexing application provides a suitable case for using multithreaded mappers. For this approach to be beneficial performance-wise, we have to discover the configuration that allows us to overcome the synchronization overhead. In the normal mapper case, the index tree is loaded by each map task, while in the multithreaded mapper, the tree is loaded once and then shared by all the threads. In both cases, there is an overhead that cannot be avoided: index tree loading versus synchronization. To discover the optimal number of map slots × threads per slot, we initially performed a small-scale experiment using 13 nodes belonging to the same cluster of the Rennes site. Disposing of eight cores per node, Hadoop tasktrackers were configured with various combinations of map slots and threads. In each setup, we performed indexing on 45 GB of data using the multithreaded mapper implementation; the index in this experiment was 460 MB in size. The results are displayed in Table 5.

In addition to these multithreaded mapper runs, we also ran the indexing with the default mapper implementation on the same dataset and on the same platform; in this case, the indexing took 40 min to complete. It is fair to assume that this is the optimal execution time in the given setup; furthermore, we can use it as baseline to compare against. The slowest run is the configuration in which eight threads are spawned within the same map slot; this is a clear effect of the reading/writing synchronization implemented in Hadoop's multithreaded mapper. As we increase the number of slots and reduce the number of threads per slot, the indexing time decreases. The best configuration that also achieves a running time comparable to the baseline is four map slots × two threads per slot.

The benefits of multithreaded mappers are revealed when processing 4 TB of data. With the default mappers, the size of the index tree (1.8 GB) compelled us to increase the RAM to 6 GB and limit the map slots per tasktracker to 4, underutilizing the node's CPU power. With the multithreaded mappers, we took

TABLE 5 Small-Scale Indexing with Varying Number of Map Threads

Mapper Slots	Threads per Mapper	Time (min)
1	8	87
2	4	68
4	2	41
8	1	42

full advantage of all the CPU cores, while sharing the memory between the threads of a mapper. Based on the small-scale experiment above, we configured the tasktrackers with four mapper slots, each mapper running two threads. This setting delivered the best execution time for the distributed indexing: 6 h and 8 min.

5.3.2 MapFiles

In some applications, the mappers require auxiliary data related only to the data chunks they process. In this case, we can avoid loading all the auxiliary data by loading only the necessary part. This partial loading of auxiliary information can be achieved by the means of *MapFiles*. Hadoop's MapFiles can be regarded as common map structures, sets of (key, value) associations that persistently store the data in binary format, as HDFS SequenceFiles. In addition to the *data file* that contains the (key, value) mappings in order, a MapFile also has an *index file* storing keys and for each key, its offset in the data file. The index file can be configured to store only a fraction of the key set.

MapFiles are an efficient tool to load auxiliary data and reduce RAM usage, since only the index file is loaded into memory, while the data file is read from disk whenever necessary. Accessing a MapFile, i.e., getting the value associated to a key, consists of (1) searching the key in the in-memory index and getting the offset of the key on disk and (2) seeking in the data file to the key's offset and reading the corresponding value.

We used MapFiles to store the batches of queries in the searching application. We performed searches on 1 TB of data in two scenarios: when the batch of queries is entirely loaded from disk into memory by each mapper, and when the batch is stored as a MapFile and only its index is kept in the main memory. Using a MapFile key interval of 1, we searched over 1 TB on 100 nodes, using three batches of different sizes. The search times shown in Table 6 prove that MapFiles are an efficient tool for managing auxiliary data. Apart from delivering comparable performance, MapFiles also reduce RAM usage. For instance, instead of loading approximately 500 MB of queries, each mapper keeps in

TABLE 6 Search Times over 1 TB on 100 Nodes

	Query Batch(#Images)	Time (s)
Complete loading	3000 12,000 25,000	382 521 755
MapFiles	3000 12,000 25,000	343 607 932

memory a 4.6 MB index of the keys. This can make a significant difference when the query batch is very large. If the query batch is of the order of gigabytes in size, loading it into memory by each mapper is inefficient and sometimes unfeasible. Even if the total available RAM memory is able to accommodate the size of the batch, this would impact the number of mappers running simultaneously. Although using MapFiles takes longer to process queries, it still allows all the mappers to run at the same time; thus, the overhead of reading from disk is eventually outweighed.

5.4 Observations

Among other tools that Hadoop provides to facilitate data distribution, we found particularly useful the *distributed cache*, which transparently transfers auxiliary data to the local disks of all the nodes.

Overall, Hadoop is helpful for achieving scalability. However, the size of the data puts a significant load on the framework. We observed that tasks often fail because of insufficient disk space, communication timeouts, etc. These failures can be avoided or at least minimized through parameter configuration. On the other hand, we encountered a few Hadoop errors that we were hard to localize and fix. At this point, the framework's fault tolerance mechanisms are crucial for completing the job, which Hadoop manages to achieve, on average.

6 CONCLUSION

We presented our experience with processing terabytes of multimedia data through the Hadoop MapReduce framework. We also described a wide collection of experiments and the practical lessons we have drawn from our experience with the Hadoop environment. We have shown the performance benefits of understanding the application workload and of tuning Hadoop configuration parameters accordingly. In addition, we addressed issues commonly found in Big Data processing: resource heterogeneity and management of large auxiliary data. The tools we employed are existing tools implemented in Hadoop, that, despite the lack of documentation, prove to be very efficient once one understands how and when to use them. Specifically, our experiments are proved the benefits of using multithreaded mappers and MapFiles in reducing memory usage while maintaining the same degree of parallelism.

Although the tools provided by the Hadoop framework to accommodate great data scales are critical for the performance of image search applications, there is still a room for improvement. For example, the multithreaded mappers synchronize when reading and writing a single key/value pair, negatively affecting the performance. One way to minimize the number of synchronization steps is to have each thread use a separate buffer for reading and writing. While this mechanism is easy to implement for writing, for reading it is less trivial, since

it requires modifications in the Hadoop's input data splitting process. Another interesting direction for future work is to implement our applications using other distributed frameworks and compare them to the Hadoop's performance. Since our experimental platform can accommodate our dataset into main memory, we plan to look into in-memory distributed frameworks, such as Main Memory Map Reduce (M3R) (Shinnar et al., 2012), a Hadoop variant that maintains the data in the cluster's main memory.

The implementation and the experiments presented in this chapter are practical applications conducted on real-life datasets. The lessons drawn from our work can be shared as best practices and recommendations in the unceasing task of rising to the Big Data challenge.

ACKNOWLEDGMENTS

This work was carried out in collaboration with Laurent Amsaleg and Gylfi Gudmundsson at the INRIA Rennes research centre. The authors would like to heartily thank them for their contributions to this work.

This work was partly achieved as part of the Quaero Project, funded by OSEO, French State agency for innovation (see http://www.quaero.org/). The experiments presented in this chapter were carried out using the Grid'5000/ALADDIN-G5K experimental testbed (see http://www.grid5000.fr/).

REFERENCES

Ahmad, F., Chakradhar, S.T., Raghunathan, A., Vijaykumar, T.N., 2012. Tarazu: optimizing MapReduce on heterogeneous clusters. SIGARCH Comput. Archit. News 40 (1), 61–74.

Architecture, L., 2013. Lambda architecture. http://lambda-architecture.net/.

Batko, M., Falchi, F., Lucchese, C., Novak, D., Perego, R., Rabitti, F., Sedmidubský, J., Zezula, P., 2010. Building a web-scale image similarity search system. Multimedia Tools Appl. 47 (3), 599–629.

Bhandarkar, M., 2010. Tutorial: MapReduce programming with Apache Hadoop. In: IEEE International Symposium on Parallel Distributed Processing IPDPS, p. 1.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51 (1), 107–113.

Douze, M., Jégou, H., Sandhawalia, H., Amsaleg, L., Schmid, C., 2009. Evaluation of GIST descriptors for web-scale image search. In: ACM Proceedings of the International Conference on Image and Video Retrieval CIVR, pp. 19:1–19:8.

Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G., 2010. Twister: a runtime for iterative MapReduce. In: ACM International Symposium on High Performance Distributed Computing HPDC, pp. 810–818.

Gigaom, 2012. Facebook has 220 billion of your photos to put on ice. http://gigaom.com/2012/10/17/facebook-has-220-billion-of-your-photos-to-put-on-ice/.

Hadoop, 2007. The Hadoop project. http://hadoop.apache.org/.

Hare, J.S., Samangooei, S., Dupplaw, D.P., Lewis, P.H., 2012. ImageTerrier: an extensible platform for scalable high-performance image retrieval. In: ACM Proceedings of the International Conference on Multimedia Retrieval ICMR, pp. 40:1–40:8.

Jackson, K., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H.J., Wright, N., 2010. Performance analysis of high performance computing applications on the Amazon Web Services Cloud. In: IEEE International Conference on Cloud Computing Technology and Science CloudCom, pp. 159–168.

Jégou, Y., Lantéri, S., Leduc, J., all, 2006. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. Int. J. HPC Appl. 20 (4), 481–494.

Jegou, H., Tavenard, R., Douze, M., Amsaleg, L., 2011. Searching in one billion vectors: re-rank with source coding. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP, pp. 861–864.

Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., Schmid, C., 2012. Aggregating local image descriptors into compact codes. IEEE Trans. Pattern Anal. Mach. Intell. 34 (9), 1704–1716.

Keahey, K., Freeman, T., 2008. Science Clouds: early experiences in Cloud computing for scientific applications. In: Proceedings of the 2008 Conference on Cloud Computing and Its Applications (CCA), Chicago, IL, USA.

Lejsek, H., Amundsson, F.H., Jónsson, B.T., Amsaleg, L., 2009. NV-Tree: an efficient disk-based index for approximate search in very large high-dimensional collections. IEEE Trans. Pattern Anal. Mach. Intell. 31, 869–883.

Lejsek, H., Jónsson, B.T., Amsaleg, L., 2011. NV-Tree: nearest neighbors at the billion scale. In: ACM International Conference on Multimedia Retrieval ICMR, pp. 54:1–54:8.

Moise, D., Shestakov, D., Gudmundsson, G., Amsaleg, L., 2013a. Indexing and searching 100M images with Map-Reduce. In: International Conference on Multimedia Retrieval ICMR, pp. 17–24.

Moise, D., Shestakov, D., Gudmundsson, G., Amsaleg, L., 2013b. Terabyte-scale image similarity search: experience and best practice. In: IEEE International Conference on Big Data, pp. 674–682.

Mundkur, P., Tuulos, V., Flatow, J., 2011. Disco: a computing platform for large-scale data analytics. In: Proceedings of the 10th ACM SIGPLAN Workshop on Erlang, pp. 84–89.

Neumeyer, L., Robbins, B., Nair, A., Kesari, A., 2010. S4: distributed stream computing platform. In: IEEE International Conference on Data Mining Workshops ICDMW, pp. 170–177.

Nister, D., Stewenius, H., 2006. Scalable recognition with a vocabulary tree. In: IEEE Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition CVPR, pp. 2161–2168.

Owen, S., Anil, R., Dunning, T., Friedman, E., 2011. Mahout in Action. Manning Publications. Shelter Island, NY.

Shestakov, D., Moise, D., Gudmundsson, G., Amsaleg, L., 2013. Scalable high-dimensional indexing with Hadoop. In: Content-Based Multimedia Indexing CBMI, pp. 207–212.

Shinnar, A., Cunningham, D., Saraswat, V., Herta, B., 2012. M3R: increased performance for in-memory Hadoop jobs. Proc. VLDB Endow. 5 (12), 1736–1747.

Shvachko, K., Kuang, H., Radia, S., Chansler, R., 2010. The Hadoop distributed file system. In: IEEE Proc. Mass Storage Systems and Technologies MSST, pp. 1–10.

Storm, 2012. The Storm project. http://storm.incubator.apache.org/.

Wen, X., Gu, G., Li, Q., Gao, Y., Zhang, X., 2012. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. In: International Conference on Fuzzy Systems and Knowledge Discovery FSKD, pp. 2457–2461.

Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I., 2013. Discretized streams: fault-tolerant streaming computation at scale. In: ACM Symposium on Operating Systems Principles SOSP, pp. 423–438.

Chapter 13

Measuring Inter-site Engagement in a Network of Sites

Janette Lehmann[*,1], **Mounia Lalmas**[†], **Ricardo Baeza-Yates**[‡]
[*] *Universitat Pompeu Fabra, Barcelona, Spain*
[†] *Yahoo Labs, London, United Kingdom*
[‡] *Yahoo Labs, Sunnyvale, California, USA*
[1] *Corresponding author: e-mail: lehmannj@acm.org*

ABSTRACT
User engagement is a key concept in the design of websites, motivated by the observation that successful websites are not just used, but are engaged with. Engagement metrics enable us to perform large-scale web analytic studies to understand how users engage with a website. Many large online providers (e.g., AOL, Google, Yahoo) offer a variety of websites, ranging from shopping to news. Standard engagement metrics are not able to assess engagement with more than one website, as they do not account for the user traffic between websites. We therefore propose a methodology for studying inter-site engagement by modeling websites (nodes) and user traffic (edges) between them as a network. Our methodology reduces the complexity of the data, and hence metrics can be efficiently employed to study user engagement within such networks. The value of our approach was demonstrated on 228 websites offered by Yahoo and a sample of 661M online sessions.

Keywords: User engagement, Network of sites, Metrics, Inter-site engagement, Large-scale web analytics

1 INTRODUCTION

User engagement refers to the quality of the user experience associated with the desire to use a website (O'Brien and Toms, 2008). In the online industry, engagement is measured through online behavior metrics aiming at assessing users' depth of interaction with a site. Widely used metrics include number of visits, click-through rates, time spent, and page views on a site. Since

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00013-7

these metrics are able to process large volumes of data in real-time, they are extensively used by the web analytics community and Internet market research companies such as comScore as proxy for online user engagement. In this chapter, we also study user engagement using online behavior metrics, which we refer to as *engagement metrics*, but with respect to a network of sites.

Many large online providers (e.g., Amazon, Google, Yahoo) offer a variety of sites, ranging from shopping to news. The aim of these large online providers is not only to engage users with each site, but with as many sites as possible. These providers spend increasing effort to direct users to various sites, for example by using hyperlinks to help users navigate to and explore other sites in their *network*; in other words, they want to increase the users traffic between sites. In this context, although the success of a site still largely depends on itself, it also depends on how and whether it is reached from other sites in the network. This leads to a strong relationship between site engagement and site traffic: each reinforces the other. We refer to this as the *network effect*.

When assessing the engagement of a site, accounting for user traffic is not new. For instance, search engines are major sources of referrals, as are social media sites. Knowing how users arrive at a site is used to optimize the site, e.g., by choosing better keywords (search engine optimization). Web analytic companies such as alexa.com produce statistics about the incoming and outgoing traffic of a site. However, the focus is the traffic to and from a site, and not the traffic between sites and its effect on user engagement.

In addition, engagement metrics cannot measure such online behavior, as they do not account for the traffic (interactions) between sites. They were designed to measure engagement at site level, and how to apply or adapt them to measure engagement in a provider network is not obvious. We therefore propose a methodology for studying *inter-site engagement*; user engagement within a network of sites. We model sites (nodes) and user traffic (edges) between them as a network, and employ inter-site metrics to measure the engagement with respect to a whole provider network of sites. Since it is unknown whether and how the traffic between sites influences user engagement, we also employ inter-site metrics at node (site) level to study the relationship between site and inter-site engagement. Some of the metrics are borrowed from the area of complex network analysis (Newman, 2003), for instance, density at network level, and page rank at node level. We then perform a large-scale study that shows how these metrics can be used and how they enhance our understanding of user engagement within a network of sites.

Our approach allows us to study user engagement at a large-scale while accounting for the relationships between sites. First, we reduce the complexity of the browsing data by transforming it into a network. Second, we can efficiently calculate inter-site metrics using a single processor machine, because the resulting network is usually small. In case of a large network, approaches performing large-scale complex network analysis can be used (e.g., Xue et al., 2010).

We start by covering previous work in Section 2. The inter-site metrics used in our work are introduced and evaluated in Sections 3 and 4, respectively. Section 5 analyses how returning traffic (users leaving the network but returning within the same session), the loyalty of users, and other aspects affect inter-site engagement. The network effect is investigated in Section 6. Section 7 analyses how hyperlinks influence inter-site engagement. The chapter ends with conclusions and thoughts for future work.

To the best of our knowledge, this is the first study looking at user engagement from a network perspective, and at possibilities to measure it.

2 RELATED WORK

Our work spans across several research areas. We discuss them, and position our work in their context.

2.1 Web Analytics and User Engagement

User engagement is the quality of the user experience associated with being captivated by an application, and so being motivated to use it (O'Brien and Toms, 2008). Users do not just use an application, they engage with it. The web analytics community and the online industry have studied user engagement through online behavior metrics that assess users' depth of engagement with a site. Several reports (e.g., Peterson and Carrabis, 2008) contain studies on existing online behavior metrics and their usage. Widely used metrics include click-through rates, number of page views, time spent on a site, how often users return to a site and number of users per month. Only online behavioral metrics are scalable to millions of users, and are commonly employed as a *proxy* for user engagement to websites. Two million users accessing a site daily is a strong indication of a high engagement with that site. Our work adds to online behavior metrics, which we refer to as engagement metrics, metrics accounting for user behavior within a provider network of sites.

2.2 Browsing Behavior

User traffic on the Web has been studied in a number of contexts, for example looking at the general browsing characteristics of users (Cockburn and McKenzie, 2001; Kumar and Tomkins, 2010; Meiss et al., 2010), how users visit websites (Bucklin and Sismeiro, 2003), the return rate to a website (Dupret and Lalmas, 2013; Lehmann et al., 2013), or how users discover and explore new sites (Beauvisage, 2009). From these and other studies, several user navigation models were developed (Chmiel et al., 2009; Meiss et al., 2010; Simkin and Roychowdhury, 2008), for example accounting for the usage of bookmarks, back buttons, teleportation, etc. These models, based on formalisms such as

branching processes, aimed to understand how users access sites and pages within them, and its effect on, for instance, link traffic and site popularity (Meiss et al., 2010; Simkin and Roychowdhury, 2008), and loyalty to a site (Beauvisage, 2009; Cockburn and McKenzie, 2001).

Several studies focused on the browsing behavior across sites and how to support such browsing behavior (Koidl et al., 2014). Jiang et al. (2012), Johnson et al. (2004), and Park and Fader (2004) analyzed cross-site navigation in the context of online shopping, and showed, for instance, that online shoppers do some research on products they wish to purchase (e.g., on social media sites) before actually purchasing them and that the more active shoppers tend to visit more shopping sites. The PEW Research Center (2010, 2012) studied the reading behavior across news sites. The studies found that 57% of users routinely obtain their news from between two to five news sites. Moreover, social media and search sites are playing an important role in the consumption of news. Dellarocas et al. (2013), De Maeyer (2010), and Roos (2012) even advocate that hyperlinks to other news sites can increase profits in a costless way, because doing so provides a more interactive, credible, transparent, and diverse news reading experience.

The main focus of our work is the development of metrics that are able to capture various aspects of cross-site navigation behavior, but between sites belonging to the same provider.

2.3 Network Analysis

In this work, we propose to incorporate user traffic into the study of user engagement by modeling sites and the traffic between them as a network. We use metrics from the area of complex network analysis (Newman, 2003) together with engagement metrics to study *inter-site engagement*. Additionally, we study the impact of the hyperlink structure in the provider network on inter-site engagement. Many types of complex networks have been studied (Newman, 2003); those closer to our work are user traffic networks (Chmiel et al., 2009; Meiss et al., 2008; Wu et al., 2011) investigating the traffic between web elements (pages, hosts, and sites), and hyperlink networks (Barabási et al., 2000) studying the topological (link) structure of the Web. Research on user traffic networks has looked at the structure of the networks (using metrics such as degree distribution) and evolution (Chmiel et al., 2009; Meiss et al., 2008). Some studies also considered engagement metrics (Chmiel et al., 2009).

There are many ranking algorithms that incorporate the browsing behavior within a network or its hyperlink structure to rank pages (Liu et al., 2008; Page et al., 1999), images (Trevisiol et al., 2012), news articles (Trevisiol et al., 2014), etc. A famous ranking algorithm is PageRank (Page et al., 1999), which ranks pages using the structure of the hyperlink network. However, Liu et al. (2008) showed that incorporating the time that users spend on a page is outperforming the solely link-based approaches. We extend existing research by developing a measure that ranks sites in a provider network according to how much users

engage to the network after visiting that site. This can be compared to the problem of influence maximization (Chen et al., 2009; Cheng et al., 2013; Kempe et al., 2003), i.e., the identification of the n most influential nodes in, for instance, social or epidemic networks. In our context, we identify nodes (sites) that maximize the engagement to the network of sites. The measure itself is an adaption of the downstream metric of Yom-Tov et al. (2013) to our network model.

3 DATA, NETWORKS, AND METRICS

Our study is based on anonymized browsing data (tuples of browser cookie, URL, referring URL, and timestamp) collected over a period of 12 months (August 2013 to July 2014) from a sample of users who gave their consent to provide browsing data through the toolbar of Yahoo. The user activities were split into sessions, where a session ends if more than 30 min have elapsed between two successive activities, a common way to identify the end of a session (Catledge and Pitkow, 1995). We extracted a sample of 661M sessions. Each session consists of page views on Yahoo and other sites (e.g., facebook.com, cnn.com). The browsing activity of a user on Yahoo during a session was used to create several provider (in our case Yahoo) networks as described in the following subsection.

To handle the large volume of browsing data, we used the Hadoop Map/Reduce framework[1] to preprocess the data and extract the provider networks. After preprocessing the data, each reducer job created a subnetwork based on a chunk of browsing data. Finally, the subnetworks were merged into one network. Since the resulting provider networks were small, we then used R[2] in conjunction to the igraph[3] and tnet[4] packages to calculate the metrics for each of the networks.

3.1 Provider Networks

Our aim is to provide insights into user engagement with respect to Yahoo's network of sites. We created two provider networks using a total of 228 sites encompassing diverse services such as news, mail, and search.[5] The first provider network consists of a selection of 73 sites based in the Unites States, whereas the second network is based on sites from five countries from the same continent. We selected the 31 most popular sites that have a counterpart in all of

1. http://hadoop.apache.org/
2. http://www.r-project.org/
3. http://igraph.org/r/
4. http://toreopsahl.com/tnet/
5. We consider all subdomains in Yahoo (e.g., mail.yahoo.com), and other domains that belong to Yahoo (e.g., flickr.com).

the countries, resulting in 155 sites in total. The provider networks are weighted directed networks $G = (N, E)$, where the set of nodes N corresponds to sites and the set of edges E to the user traffic between them. The edge weight $w_{i,j}$ between node (site) n_i and node (site) n_j represents the size of the user traffic between these two nodes, which we define as the number of clicks from n_i to n_j. Whereas the nodes in the network are fixed, the edge weights depend on the selected browsing data. This allows us to create different network instances.

3.1.1 Network Instances

The network instances are listed in Table 1. We defined various network instances of the INT network to evaluate the metrics described at the end of this section. The metrics characterize the inter-site engagement between nodes or within the whole network. The metrics are evaluated in Section 4. We extracted browsing data from each day between August 2013 and July 2014. The browsing data were used to create 365 networks ($INT_{01/08}$, …, $INT_{31/07}$), each representing the traffic of the INT network on a certain day (here 08 refers to August, 07 to July, etc.). We also defined a network INT_{Feb} based on the browsing data of February 2014 to study the differences of inter-site engagement between sites. Finally, five country-based networks were created (INT_{c1},..,INT_{c5}), where each contains the sites and traffic of one country of the INT network. Looking at specific countries enables us to compare the inter-site engagement, on a per country basis.

In the remaining sections, unless otherwise started, we study the US network, more precisely, the network instances based on the browsing data of February 2014 (US_{Feb}), and the daily networks ($US_{01/08}$, …, $US_{31/07}$).

TABLE 1 Network Instances Based on Five Countries in One Continent and the US Network

Network	Number of Instances	Clicks Per Network
Monthly-based networks (February 2014)		
INT_{Feb}	1	16M
US_{Feb}	1	235M
Daily-based networks (August 2013–July 2014)		
$INT_{01/08}$, …, $INT_{31/07}$	356	577K|230K
$US_{01/08}$, …, $US_{31/07}$	356	8.6M|2.9M
Country-based networks		
INT_{c1}, …, INT_{c5}	5	3.2M|3.6M

For each network, we provide the number of network instances and the average and standard deviation of the number of clicks per instance.

3.1.2 Site Categories

We annotated the sites in the networks using an adapted version of the schema described by Lehmann et al. (2013):

- 24 front pages and site maps (e.g., yahoo.com, everything.yahoo.com) [Front page]
- 46 service sites (e.g., mail.yahoo.com, calendar.yahoo.com) [Service]
- 73 news sites (e.g., news.yahoo.com, finance.yahoo.com) [News]
- 32 leisure and social media sites (e.g., tumblr.com, games.yahoo.com) [Leisure]
- 53 provider sites (e.g., account.yahoo.com, help.yahoo.com) [Provider]

Two examples of network instances are displayed in Fig. 1. The nodes represent the sites colored by their category. On the left side, we can see the very densely connected US network. The more central a node is the higher its connectivity to other nodes in the network. We notice that there is no site category that predominates the center in the network, i.e., each site category has densely and loosely connected nodes. On the right side, the INT network is displayed. We observe five densely connected modules, each representing a country. However, we can see connections between the modules implying that some users access sites from different countries.

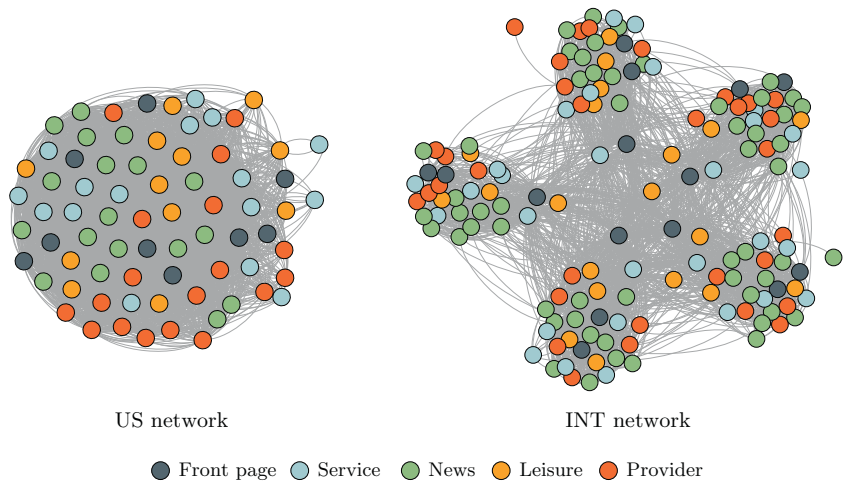


FIGURE 1 The US and INT provider networks based on browsing data of February 2014. The edges are not weighted for confidentiality reasons. The Force-Atlas algorithm is used for the visualization.[6]

6. https://gephi.github.io/

3.2 Network-Level Metrics

Network-level metrics are concerned with the browsing behavior within the whole provider network, which we refer to as the *provider engagement*. We employ three metrics of provider engagement. These are listed in Table 2, and can be of two types. The network *popularity* (first type) is measured by the number of sessions in which a site in the network was visited (*#Sessions*). The browsing *activity* (second type) in the network during an online session is described by the average time users spend in the network (*DwellTime*) and the number of sites visited (*#Sites*).

To measure the inter-site engagement, i.e., the traffic between sites, we employ standard graph metrics and add to them metrics that provide us with further information about the network structure. We refer to them as *inter-site* metrics. Numerous graph metrics exist. In this chapter, we focus on a subset of them, which are sufficient for our purpose. We discard metrics that could not be used to measure user engagement, and metrics for which we observed a very high correlation to those selected. This process resulted into the following five inter-site metrics. When describing them, we specify how they can be used in the context of user engagement.

TABLE 2 Network-Level Metrics: Metrics Used to Analyze the Browsing Behavior Within a Provider Network

		Engagement	
Metric	**Description**	**Low**	**High**
Inter-site engagement			
Flow	Extent of the inter-site engagement	0	$(\|N\| - 1)/\|N\|$
Density	Diversity of inter-site engagement	0	1
Reciprocity	Homogeneity of traffic between sites	0	1
EntryDisparity	Variability of in-going traffic to the network	1	0
ExitDisparity	Variability of out-going traffic from the network	1	0
Provider engagement			
[POP] #Sessions	Total number of sessions in the network	0	∞
[ACT] DwellTime	Avg. time per session in the network	0	∞
[ACT] #Sites	Avg. number of sites per session in the network	0	$\|N\|$

$|N|$ refers to the number of nodes in the network.

3.2.1 Flow

The flow measures the extent to which users navigate between sites. It is defined as follows:

$$\frac{\sum_{i,j} w_{i,j}}{\sum_i v_i}$$

where $w_{i,j}$ is the total number of clicks between node n_i and n_j (i.e., the edge weight) and v_i is the total number of visits on node n_i. For example, a network with 6 visits, 3 on a service and 3 on a news site, can have different levels of flow. If there are 6 users, 3 solely visiting the service and 3 solely visiting the news site, the flow will be $0/6 = 0$. If two of the visits belong to one user accessing both sites, there will be traffic in the network, and the flow value will be $1/6$. A high value indicates a high inter-site engagement; users navigate often between sites in the network.

3.2.2 Density

The density (Wasserman, 1994) describes the connectivity of the network. It is the ratio between the number of edges compared to the number of all possible edges. In Fig. 1, we see that the density of the US network is much higher than in the INT network, as there are few connections between nodes of different countries. A high connectivity (or density) means that the inter-site engagement is highly diverse; users navigate between many different sites.

3.2.3 Reciprocity

The reciprocity measures the homogeneity of traffic between two sites, i.e., the percentage of traffic between two sites that is in both directions. We use the definition of Squartini et al. (2013) for the reciprocity of weighted networks:

$$\frac{\sum_{i<j} \min[w_{i,j}, w_{j,i}]}{\sum_{i \neq j} w_{i,j}}$$

where $w_{i,j}$ is the weight of the edge from node n_i to node n_j. There are two reasons why a high reciprocity can be interpreted as a high engagement. First, the traffic from site i to j is from a different user group than the traffic from site j to i. In this case, a high reciprocity implies that the two user groups engage to the same extent on both sites. Second, the traffic between the sites comes from the same users. In this case, users do not only navigate from one site to another, they also return to the previously visited site.

3.2.4 Entry Disparity and Exit Disparity

The disparity refers to how the traffic to and from the network is distributed over the sites. For instance, a high entry (exit) disparity indicates that there are only

few sites used to enter (leave) the network. We use the group degree measure of Freeman (1979) and adapt it:

$$\frac{\sum_i (g^*_{\max} - g^*_i)}{|N| \cdot \sum_i g^*_i}$$

Here, $|N|$ is the number of nodes in the network, g_i^{in} is the number of network visits that started at node n_i (user entered the network), and g_i^{out} is the number of network visits that ended after visiting node n_i (user left the network). The maximum values of g_i^{in} and g_i^{out} are defined by $g_{\max}^{\text{in}}$ and $g_{\max}^{\text{out}}$, respectively.

We hypothesize that a low disparity (all nodes are equally used to enter and leave the network) reflects a high inter-site engagement. The network itself is less vulnerable, as the outage of one node (e.g., a front page) will not affect users entering the network. Moreover, it suggests that users do not need a front page to access other sites in the network; they know the site and go to it directly (maybe through a bookmark or a search site).

3.3 Node-Level Metrics

Node-level metrics measure the browsing behavior on a site within the network. Table 3 contains a list of all such metrics used in our work, their description, and their value range. We use two metrics that describe the engagement of users on the site (*site engagement*). The popularity of a site is measured by the number of sessions in which the site was visited (*#Sessions*), and the browsing activity on a site is characterized by the average (median) time users spend on the site during a session (*DwellTime*). We additionally employ two *multitasking metrics* defined in Lehmann et al. (2013): the number of times a site was visited during an online session (*#Visits*), and the cumulative activity (*CumAct*) which accounts for the absence time between visits within a session. Many visits and a long absence time between the visits is an indication of high loyalty to the site, i.e., the user is returning to the site to perform some new tasks within the same session (Lehmann et al., 2013).

We also employ four inter-site metrics, each accounting in a different way for the traffic to and from a site. All metrics consider the edge weights (number of clicks) between sites.

3.3.1 PageRank

The importance of pages in the Web is measured by PageRank (Page et al., 1999). The original definition considers the hyperlinks between pages, more precisely, the links leading to a page. Applied to our context, given the traffic between sites, *PageRank* corresponds the probability that a user randomly navigating through the network will arrive at any particular site.

TABLE 3 Node-Level Metrics: Metrics Used to Analyze the Browsing Behavior on a Site in the Provider Network

Metric	Definition	Engagement	
		Low	High
Inter-site engagement			
PageRank	Probability that a user will visit the site	0	1
Downstream	Avg. number of sites visited after visit on site	0	∞
EntryProb	Probability that a user enters the network in this site	0	1
ExitProb	Probability that a user leaves the network in this site	1	0
Site engagement			
[POP] #Sessions	Total number of sessions on site	0	∞
[ACT] DwellTime	Avg. time per session on site	0	∞
Multitasking			
[MT] #Visits	Avg. number of visits per session on site	0	∞
[MT] CumAct	Cumulative activity for the time between visits	0	∞

3.3.2 *Downstream Engagement*

Whereas page rank measures the probability that a random user will visit any particular site, we analyze here the browsing behavior of a random user through the network who starts at a certain site. Motivated by Yom-Tov et al. (2013), we define downstream engagement in the context of traffic networks as follows. We use a discrete-time Markov process to simulate the browsing behavior in the network. Hence, the sites correspond to the states and the edge weights correspond to the transition probabilities. Additionally, we assign to each site its exit probability (the definition is given later in this section), i.e., at each step in the simulation there is a certain probability that a random user will leave the network.

For each site in the network, we now simulate the navigation of users through the network when starting on that site. The simulation ends when all users have left the network. Based on the simulated navigation paths of the users, we are able to compute several metrics; such as the time users spend in the network (i.e., sum of the dwell time of the visited sites), or the number of sites they visited (i.e., the path length). Each metric shows which sites are maximizing the engagement to the network. Since the focus of our work is on interactions

(traffic) between sites, we define downstream engagement as the average number of sites a random user visited according to the simulation.

3.3.3 Entry Probability and Exit Probability

The two metrics measure the probability that users enter or leave the network from the site under consideration. *EntryProb* is the percentage of visits to a site in which the user entered the network. *ExitProb* is the percentage of visits to a site from which the user leaves the network afterwards and thus does not continue browsing in the network. A high entry probability indicates that a site plays an important role in promoting inter-site engagement, whereas a high exit probability refers to a site with a negative effect on the inter-site engagement.

We next evaluate the value of these network- and node-level metrics in the context of user engagement, more precisely, in measuring inter-site engagement in a network of sites, such as those offered by Yahoo. Since our data do not follow a normal distribution, and thus to avoid the influence of heavy outliers, we do not use the metric values per site, but the corresponding ranks.

4 EVALUATING INTER-SITE METRICS

We evaluate the applicability of inter-site metrics, defined in the previous section, to measure user engagement. The metrics at network level enable us to compare provider networks, for instance, from different countries, showing, for instance, that some provider networks have a higher traffic than others. Additionally, the metrics at node level enhance the understanding of how users engage with a single site and how the traffic between sites affects this. First, we compare inter-site metrics with standard engagement and multitasking metrics. Second, we present two case studies showing how inter-site metrics can be used to enhance our understanding of user engagement. We restrict ourselves to the INT network. This network is especially interesting, as it enables us to compare engagement between provider networks from different countries.[7]

4.1 Site and Provider Network Rankings

We use the daily networks introduced in Section 3.1, namely $INT_{01/08}$, …, $INT_{31/07}$. We calculate the network-level metrics for each network, and the node-level metrics for all nodes in each network. We rank networks (nodes) according to each metric and then evaluate the similarity between these rankings using Spearman's rank correlation coefficient ρ. If two metrics produce the same ranking, they are equivalent and hence one is redundant. However, similar rankings may point to interesting dependencies between the engagement and traffic characteristics of a node or the whole network. We only report correlations that are statistically significant (p value < 0.01).

7. Similar results were reached for the US network.

4.1.1 Network-Level Metrics

The correlations between the network-level metrics are presented in Table 4. The density of a network is increasing (more sites become connected) with the number of sessions ($\rho = 0.92$). This means that the more users are visiting the network, the more diverse is the inter-site engagement; users visiting the network for many different reasons since they access different groups of sites. The metrics *Flow* and #*Sites* are moderately correlated ($\rho = 0.65$). The more sites are visited during a session, the higher the flow of traffic. However, the correlation is not high enough to suggest that one of the metrics is redundant. Whether the traffic between two sites is unidirectional or not (*Reciprocity*) does not depend on any of the other considered metrics. We even cannot report a correlation to *DwellTime*, as the p value is above 0.01.

Finally, we observe a strong correlation between the two disparity metrics, *EntryDisparity* and *ExitDisparity* ($\rho = 0.84$), indicating that the volume of in- and out-going traffic of the nodes depend on each other. The two metrics also correlate negatively with the density and the number of sessions of a network. This suggests that low engaging networks have some nodes that are used to enter (leave) the network (e.g., front pages), whereas in high engaging networks users enter (leave) the network over many nodes. However, both metrics are needed, as the correlations are only moderate.

To conclude, the metrics flow, reciprocity, and disparity capture distinct aspects of how users engage with a network of sites. The density, on the other hand, relates to the popularity of a network.

TABLE 4 Correlations Between Network-Level Metrics

	Flow	Density	Reciprocity	Entry Disparity	ExitDisparity
Density	–				
Reciprocity	0.15	0.48			
EntryDisparity	0.23	−0.61	−0.38		
ExitDisparity	0.30	−0.60	−0.32	**0.84**	
[POP] #Sessions	–	**0.92**	0.42	−0.54	−0.55
[ACT] DwellTime	0.35	−0.45	–	0.33	0.38
[ACT] #Sites	**0.65**	−0.25	0.25	–	0.20

Correlations with a p value <0.01 are not reported (–), and correlations above 0.5 or below −0.5 are highlighted in bold.

4.1.2 Node-Level Metrics

Table 5 reports the metric correlations at node level. There are no correlations between the inter-site metrics, and the activity or multitasking metrics (all correlations are below 0.4). We therefore focus on the correlations between the inter-site metrics and the popularity metric #*Sessions*.

We observe that popular sites in the provider network (e.g., front pages), are also visited frequently when browsing through the network (ρ(#*Sessions*, *PageRank*) $= 0.85$), but users do not visit many other sites after visiting the site (we have ρ(#*Sessions*, *Downstream*) $= 0.17$).

The strong correlation between *EntryProb* and *ExitProb* ($\rho = 0.79$) suggests that nodes used to enter the network are also frequently used to exit the network. The fact that these two metrics do not correlate with the other inter-site metrics indicates that entry and exit points of a network do not correspond to nodes that play an important role in directing traffic to other nodes (e.g., ρ(*EntryRatio*, *Downstream*) $= -0.27$), and to nodes that are visited frequently when browsing through the network (e.g., ρ(*EntryRatio*, *PageRank*) $= -0.10$).

In conclusion, downstream engagement, and the entry/exit ratio of a node bring new insights regarding the engagement at site level, whereas the page rank relates to sites that are very popular.

4.2 Case Studies

We present two case studies that demonstrate how using inter-site metrics can enhance our understanding of user engagement. First, we compare different provider networks with respect to their inter-site engagement. Then, we use node-level metrics to analyze how the traffic differs between sites.

TABLE 5 Correlations Between Node-Level Metrics

	PageRank	Downstream	EntryProb	ExitProb
Downstream	0.30			
EntryProb	−0.08	−0.27		
ExitProb	−0.10	−0.22	**0.79**	
[POP] #Sessions	**0.85**	0.17	0.12	0.08
[ACT] DwellTime	0.06	0.04	−0.19	−0.18
[MT] #Visits	0.08	0.02	0.13	0.18
[MT] CumAct	0.31	−0.02	0.35	0.32

Correlations above 0.5 or below −0.5 are highlighted in bold.

4.2.1 Comparing Provider Networks

The objective is to show that provider networks vary in their inter-site engagement. We compare the five country-based networks (INT_{c1}, …, INT_{c5}) with each other. Using network-level metrics we compare engagement between networks, as we do it when comparing sites using node-level metrics. Figure 2 depicts the differences between the networks using four selected inter-site metrics. To capture the engagement to the provider network, we employ one provider engagement metric: *DwellTime*. The metrics are normalized by the z-score, hence the figure shows the extent to which the standard deviation of a metric rank is above or below the mean. The countries are ordered by decreasing *Flow*.

The highest inter-site and provider engagement can be observed for the first provider network (Country_1). The network has the highest flow, and dwell time. Also the reciprocity is above average. This shows that users spend a lot of time in that network, and while visiting the network they navigate often between many sites and do so in both directions. Although the network has the highest engagement, the density is only average compared to the other networks, indicating that users do not navigate between many different sites.

We look now at the second provider network (Country_2). In this network, the flow is high and homogeneous (high *Flow* and *Reciprocity*), and also the diversity of the inter-site engagement is high (high *Density*), but the dwell time is below average (low *DwellTime*). This indicates that users access many sites in the network, but they navigate quickly between them.

The opposite can be observed for the fourth network (Country_4). The flow is below average, indicating a low inter-site engagement, but the dwell time per session is above average. We hypothesize that each user visits only a small subset of sites in the network, but spending a lot of time on it. The low value of *EntryDisparity* and the high value of *Density* suggests that still all sites in the network are visited, but from different users.

The last provider network (Country_5) has the lowest inter-site and provider engagement. We can see that users enter and leave the network over a subset of nodes (highest *EntryDisparity*). The users hardly navigate to other nodes (lowest *Flow* and low *Density*), or spend much time in the network (lowest *DwellTime*).

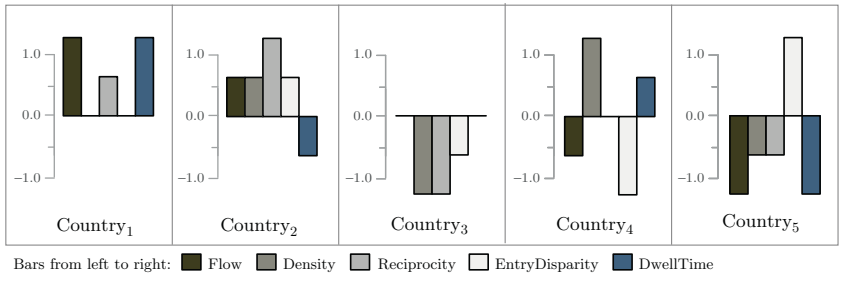


FIGURE 2 Comparing provider networks from different countries using network-level metrics.

In this section, we demonstrated that network-level metrics enhance our understanding of how users engage with a whole provider network. Inter-site and provider engagement of a network can differ, implying that both metric types should be employed when analyzing the engagement of a network of sites. Indeed, we saw that some networks have a high provider engagement, but a low inter-site engagement, and vice versa.

4.2.2 Engagement Patterns at Site-Level

We study the different engagement patterns at site-level. Our goal is to show that inter-site metrics provide additional insights to those coming from assessing user behavior within a site. The engagement patterns are determined based on several node-level metrics, *PageRank*, *Downstream*, *EntryProb*, *DwellTime*, and *CumAct* values. Each site is represented as a 5-nary vector, each dimension corresponding to one such metric. We cluster the vectors using k-means. The number of clusters is determined by a minimal cluster size such that each cluster contains at least 20% of the sites under consideration (155 of them). We use the network constructed for February 2014 (INT_{Feb}).

We obtain four clusters, shown in Fig. 3, each representing an engagement pattern. The first row contains the cluster centers normalized by the z-score, and the second row presents the number of sites within each cluster. The last row shows statistics related to the site categories per cluster. We define $p(c)$ as the probability that a site belongs to category c, and $p(c|cl)$ as the probability that a site in cluster cl belongs to category c. We then define the difference in the probability PD as follows:

$$\frac{p(c|cl) - p(c)}{\max(p(c|cl), p(c))}$$

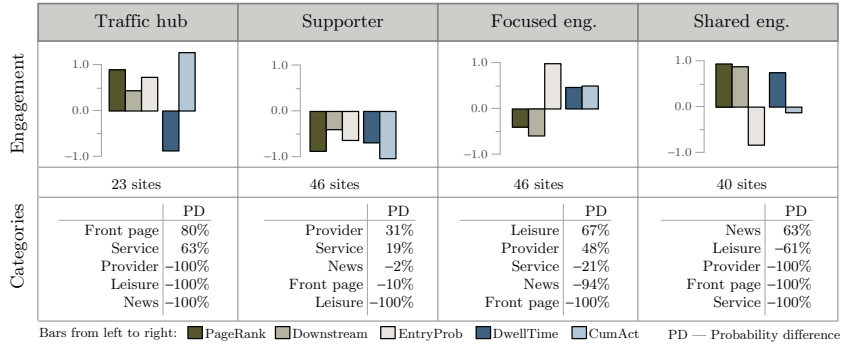


FIGURE 3 (First row) Site clusters and browsing characteristics. (Second row) Number of sites per cluster. (Third row) Likelihood that a category occurs in a given cluster with respect to its likelihood that it occurs at all.

This corresponds to the likelihood that a category occurs in a given cluster with respect to its likelihood that it occurs at all. Each cluster, a pattern, is given a name reflecting its main characteristic.

Traffic Hub. This cluster contains sites with a high inter-site engagement. The sites are important for the network, as they forward traffic to other sites. Users visit these sites when entering the provider network (high *EntryProb*) to access many other sites in the network (high *Downstream*). While browsing through the network, users regularly return to these sites, even after a long period of absence, to access further sites (high *PageRank* and *CumAct*). We also observe the lowest *DwellTime* for this cluster, which indicates that users do not spend much time on the sites; they quickly navigate to their target site. Front pages and service (e.g., search) sites belong to this cluster.

Supporter. Sites belonging to this cluster are sites on which users do not spend much time (low *DwellTime*), and do not return after a longer period of absence (low *CumAct*) during the session. The low *PageRank* indicates that the sites are not very important (central) for the network, and hence they are not visited frequently. Provider (e.g., info.yahoo.com) and service (e.g., address.yahoo.com) sites belong to this cluster. Users only visit the sites when specific information is required. As a result, users do not access these sites directly when entering the network (low *EntryProb*), but from other sites in the network. After they have found the required information, they are done, although they may visit a few other sites in the network (low *Downstream*).

Focused Engagement. Many leisure sites belong to this cluster. We observe that users visiting leisure sites (game and social media sites) spend a lot of time on them (high *DwellTime*), and also return after a longer period of absence within the same session (high *CumAct*). It is also less likely that a user navigating through the network will arrive at a leisure site (low *PageRank*), and that a user that is visiting a leisure site will navigate to other sites in the network (low *Downstream*). Users access leisure sites directly (high *EntryProb*), and then they are solely engaged in their leisure activities. The same can be observed for some provider sites (e.g., messenger.yahoo.com), which are visited to download an application provided from Yahoo.

Shared Engagement. Mainly news sites belong to this cluster, which is characterized by the highest inter-site engagement. Although the sites have the lowest entry probability, they are well connected to many other sites (highest *PageRank*). We hypothesize that users enter the network over front pages, and then visit a news site. Although they dwell long on the news site (high *DwellTime*), it is very likely that they continue browsing to other sites in the network (highest *Downstream*).

To summarize, we observe that sites exhibit different engagement patterns. For leisure sites, the attention of users is mostly focused on the site, whereas when reading news, users exhibit a high inter-site engagement (e.g., visiting several other sites). Then, there are sites that have the function to forward traffic to other sites (e.g., front pages) or to support users in the network (e.g., help sites).

5 STUDYING INTER-SITE ENGAGEMENT

We study inter-site engagement with the metrics we have proposed and showed to bring different insights than standard engagement metrics. Our aim is to analyze how inter-site engagement differs depending on the loyalty of users, whether the user visits the network on a weekday or the weekend, and the upstream traffic. We also investigate the effect of users leaving the network, but returning within the same online session. We do so by defining further network instances and compare them with each other.

The network-level metrics *Flow*, *Density*, *Reciprocity*, *EntryDisparity*, *DwellTime*, and *#Sites* are used to characterize the inter-site engagement with respect to the whole provider network. The node-level metrics *PageRank*, *Downstream*, *EntryProb*, *DwellTime*, and *CumAct* bring additional insights about the inter-site engagement with respect to a site. To study the difference between a metric value v_1 from one network with the metric value v_2 from another network, we measure the relative difference as:

$$d = \frac{v_2 - v_1}{\max(v_1, v_2)}$$

where d is a value between -1 (decrease of -100%) and $+1$ (increase of +100%).

The results for each study are presented in a Figure (e.g. Fig. 4). The top part displays the differences of the network-level metrics, and the bottom part depicts the average differences of the node-level metrics per site category. The differences for each node-level metric are presented in a bar chart where a bar corresponds to a site category.

We start with comparing the network instances constructed to study how inter-site engagement is affected by user loyalty.

5.1 User Loyalty

We group users according to a loyalty criteria, measured in this chapter, by the number of active days within February 2014. Following Lehmann et al. (2012), we define three loyalty levels: Casual (1 active day), Engaged (2–14 active days), and Loyal (more than 14 active days). We then use the browsing data of February 2014 of the Casual, Engaged, and Loyal users to create three user-based networks.

We first analyze the differences at network level (Fig. 4 (top part)). We see that Engaged users navigate more often (*Flow*: +17.8%), and between more sites (*Density*: +43.1%) than Casual users. The values increase again from Engaged to Loyal users. This shows that when the inter-site engagement increases, the more loyal the users are. Although we reported in Section 4.1 a weak positive correlation between the reciprocity and the density of a network ($\rho = 0.5$), we observe here that the reciprocity decreases with increasing loyalty of users (e.g., from Engaged to Loyal: −38.9%). This indicates that, with loyal users,

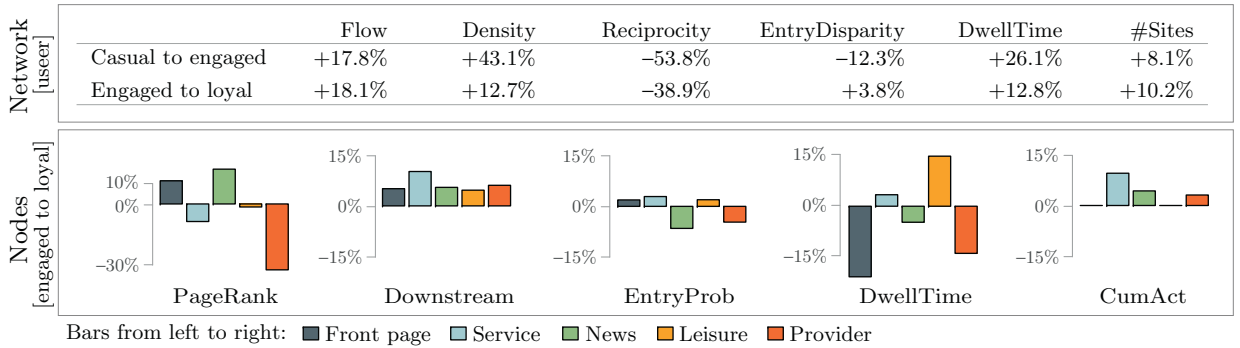


FIGURE 4 Differences between user-based networks.

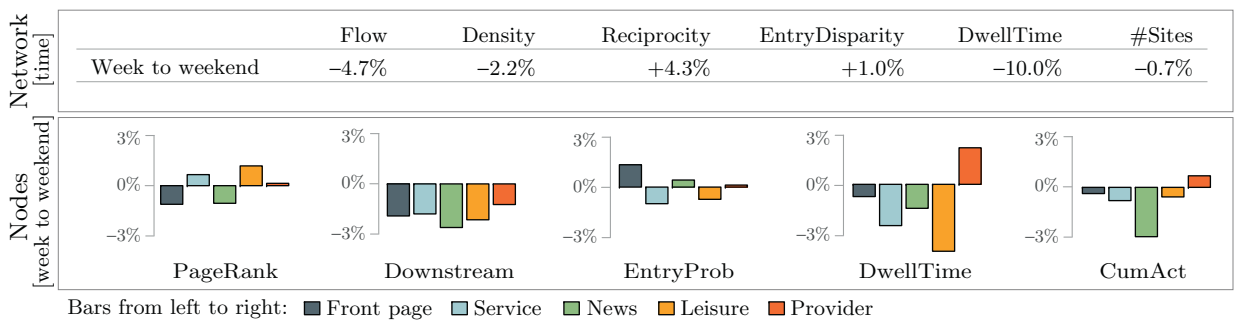


FIGURE 5 Differences between time-based networks.

the traffic in the network becomes more directed, more users go from one site to another but return less to the previous site. We speculate that, for instance, Engaged users always return to the front pages to access other sites in the network (e.g., *frontpage* → *news* → *frontpage* → *leisure*). Loyal users, on the other hand, are "aware" of links that allow them to access other sites directly (e.g., *frontpage* → *news* → *leisure*). We also observe that the engagement on the network increases with the loyalty of users. Loyal users spend more time on sites (*DwellTime* increases), and they also visit more sites (#*Sites* increases).

We analyze how the inter-site engagement differs at site level. Figure 4 (bottom part) compares the Engaged and Loyal networks. As already observed at network level, loyal users browse more through the network. This is further accentuated at node level by the increase in downstream engagement. For provider-related sites (e.g., help and account setting sites) we observe a significant decrease for page rank value, and also for the entry probability, and dwell time. This indicates that Loyal users are rarely visiting provider-related sites. They do not need to access help sites very much, likely because their account-related settings have already been performed a while ago.

Finally, in terms of time spent on the network, Loyal users seem to spend a considerable amount of time on leisure sites (increase in *DwellTime*), compared to Engaged users. We also observe a significant decrease of dwell time for front pages. Front pages can be compared to search sites; a low dwell time is a sign of a good user experience, as these sites are used to navigate (quickly) to other sites. We speculate that Loyal users know the front page well, and hence are able to move quickly to the site they want to reach. Engaged users, on the other hand, cannot find the hyperlink they are searching for immediately, and thus spend more time on such sites. As such, they may get more distracted by what is on offer on the front page.

5.2 Weekdays and Weekend

Previous research showed that it is important to consider temporal aspects when studying user engagement (Lehmann et al., 2012). We therefore compare inter-site engagement during weekdays with that from weekends. We use the daily networks ($US_{01/08}$, …, $US_{31/07}$) and split them depending on whether the network refers to traffic during a weekend or a weekday.

Although the differences are not as high as for the user-based networks, interesting observations can be made (see Fig. 5). During the weekend, many metrics at network level (e.g., *Flow*: −4.7%, *DwellTime*: −10.0%), and site level (e.g., decrease in *Downstream* and *CumAct*) are lower. This means a lower site and inter-site engagement during the weekend. However, the reciprocity is higher (*Reciprocity*: +4.3%). We speculate that many users, who visit the network during the week, do it to perform specific goal-oriented tasks (e.g., checking mails, or reading news to keep up-to-date), and therefore navigate only from the front page to their desired site (i.e., the traffic is unidirectional).

The tasks during the weekend differ, as shown when looking at the statistics at node level. During the weekend, users may not have to or do not wish to perform these goal-oriented tasks (lower *PageRank* and *CumAct* for front pages and news sites), and therefore they have the time to engage in leisure activities (higher *PageRank*), to do account-related settings and to try out applications offered by Yahoo (higher *DwellTime* and *CumAct* for provider sites).

5.3 Returning Traffic

During their online sessions, users engage in multitasking (Lehmann et al., 2013), and as a result, they revisit sites several times, after a short or long time, with the same session. While doing so, users access sites outside the provider network, for instance, navigating from Yahoo mail to Facebook, and then back to Yahoo mail. In our data, we observed that on average 20% of the page views during an online session belong to sites that are not part of the provider network.

We analyze how this behavior affects the characteristics of the networks. We therefore define a second type of edge, which we call "return edge," which corresponds to users navigating from a site in the provider network to external sites, but returning to another site in the network within the same online session (returning traffic). Figure 6 compares the internal traffic and the returning traffic of the US network of February 2014 (US_{Feb}), where for the latter network, return edges are added to the original network. Thereby, traffic returning to the same site n_i is represented by an additional edge $w_{i,i}$. Note that as our engagement and multitasking metrics do not consider the traffic between sites, their values are the same for the two networks. The reciprocity and closeness metrics do not consider traffic returning to the same site, as the two metrics are used to characterize the traffic between distinct sites. However, the metrics are still useful for analyzing the change in the traffic in the network when accounting for returning traffic.

The results show that leaving the network does not necessarily entail less engagement. Users often return to the network (*Flow*: +59.9%) and more sites become connected through returning traffic (*Density*: +10.3%). A higher density indicates that users leave the network from one site and return to some other site in the network, but hardly ever navigate directly between the two sites. This might point to missing hyperlinks in the provider network. Adding these hyperlinks could increase site and inter-site engagement, as they may help users browse through the network, and thus stay longer. How hyperlinks can influence the engagement in the network is investigated in Section 7.

The value of the entry disparity metric increases significantly (+67.0%), indicating that there are some sites that are less frequently used to enter the network when accounting for returning traffic; these sites are often used to return to the network within the same session. Interestingly, when looking at the entry probability per site category (node-level metrics), we are not able to identify a site category for which the entry probability decreases significantly more or less. The downstream engagement also increases to the same extent for

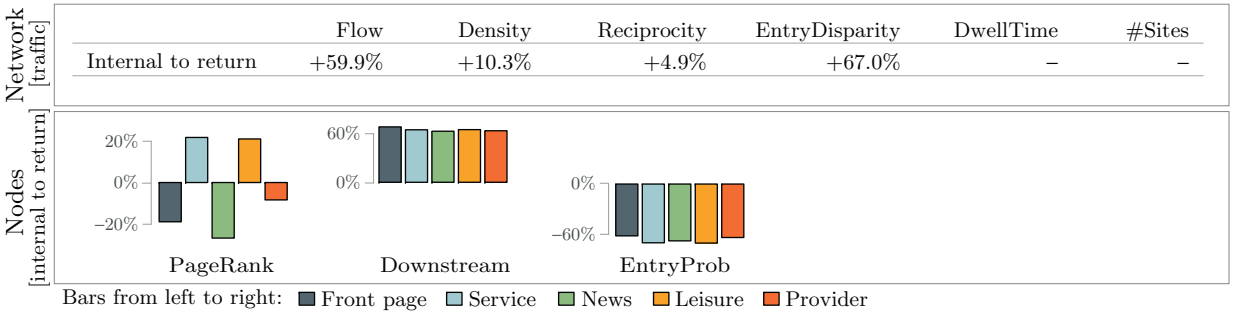


FIGURE 6 Differences between traffic-based networks.

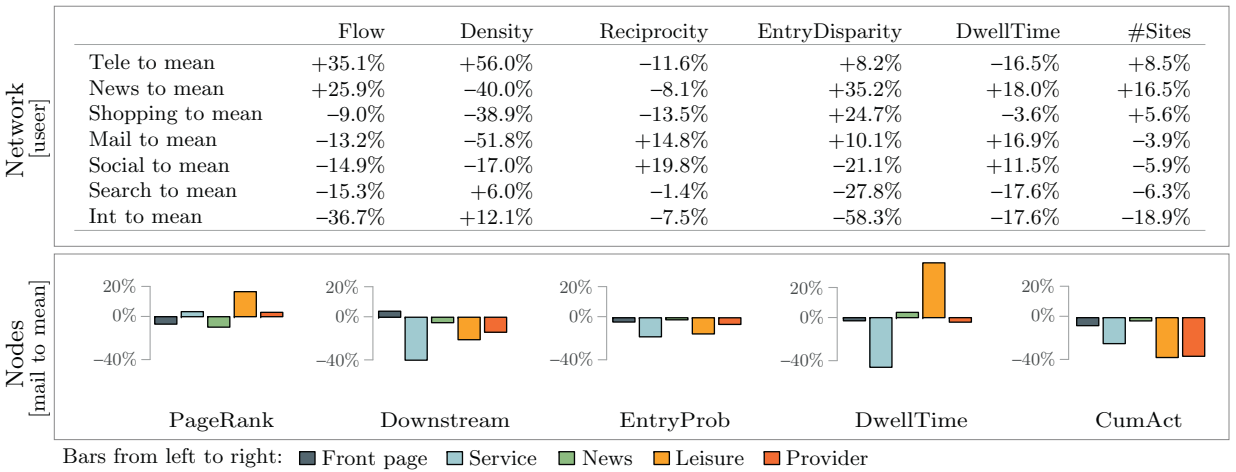


FIGURE 7 Differences between upstream-based networks.

all site categories. This suggests that whether the user is leaving the network and returning to it afterwards does not depend on the site category. In fact, the returning traffic is equally distributed over all categories of sites.

The only difference we can see is with respect to the *PageRank* metric. When we consider the returning traffic, the importance of service and leisure sites increases (higher *PageRank*), i.e., these sites become more connected through returning traffic.

5.4 Upstream Traffic

It was shown by Trevisiol et al. (2012) that user browsing behavior within a network depends on where the user is coming from when entering the network (i.e., the upstream traffic). We investigate whether the upstream traffic has an effect on inter-site engagement. First, we define the upstream type (e.g., search, mail) of an online session. Using the referring URL and the schema from Lehmann et al. (2013), we annotate the sites from which users are coming from when entering the network. Additionally, we added the site category "Int" which refers to Yahoo sites that are not in the considered provider network (e.g., hk.yahoo.com when dealing with the US network). If the user accessed the network by using a bookmark, or entering the URL in the address bar, no referring URL is defined. In this case, the upstream type is "Tele," to refer to teleportation. This resulted into the following upstream types:

- 87.95% of teleportation [Tele]
- 4.32% of internal traffic (e.g., hk.yahoo.com, uk.news.yahoo.com) [Int]
- 1.42% of search (e.g., google.com, bing.com) [Search]
- 0.51% of social media (e.g., facebook.com, twitter.com) [Social]
- 0.19% of shopping (e.g., coupons.com, booking.com) [Shopping]
- 0.10% of news (e.g., cnn.com, forbes.com) [News]
- 0.07% of mail (e.g., live.com, mail.google.com) [Mail]

In total, 5.44% of the sessions could not be assigned with one of the defined upstream types and are discarded in the following analysis. We now create a network for each upstream traffic type and calculate the network- and node-level metrics per network. Since there are many different types of upstream traffic, we do not compare the upstream traffic networks with each other. Instead, we compute the average value of each metric and analyze the difference between the metric value and the average metric value.

Users frequently enter the network using teleportation (87.95% of the sessions). Teleportation is a sign that users are highly engaged with the network, as they use bookmarks, remember the domain name and enter it directly, or simply start typing the URL which then get autocompleted. This is also reflected by the network-level metrics in Fig. 7 (top part) by the high values of density

(+56.0%), traffic flow (+35.1%), and the average number of visited sites during a session (+8.5%). Interestingly, the dwell time in the provider network is below average (−16.5%).

A dwell time above average can be observed for users coming from news, mail, or social media sites. However, we can also see that the *Density* is below average, indicating that users do not navigate between many different sites. We know from Section 4.2.2 that news and social media sites inside the network also have a high dwell time. We speculate that users coming from such sites continue reading (socializing) on news (social media) sites inside the provider network. In doing so, they are highly engaged, as shown by the high value of dwell time. Users who come from external news sites (e.g., cnn.com) even visit many news sites inside the provider network (*Flow* and #*Sites* are above average).

We can see in the bottom part of Fig. 7 (node-level metrics) that users who arrive from mail sites frequently visit leisure sites in the network, and that they spend a lot of time on them. The page rank and the dwell time is above average. Users might have received a notification via email from a leisure site, which led them to visit the site. We also observe that mail users focus much less on service sites, as all five metrics are below average (except *PageRank* which is on average).

The lowest engagement is observed for users who are coming from search or from other Yahoo sites. The flow and the two engagement metrics (*DwellTime* and #*Sites*) are low. However, the entry disparity is also below average (Search: −27.8%, and Int: −58.3%), showing that users enter and leave the network from all sites. We speculate that users access directly the sites they are interested in (front pages are not used), to perform a quick task, and then leave again.

This section shows that inter-site engagement depends on many factors such as the loyalty of users and the day of the week. Accounting for multitasking (i.e., the returning traffic) also leads to a better understanding on how users engage with sites. In addition, considering where users are coming from provides information about what else the users are doing in the network afterwards. We have shown all these using metrics brought in to measure inter-site engagement.

6 THE NETWORK EFFECT

Liu et al. (2008) showed that the popularity of pages depends on the traffic between them, and we already observed in Section 4 that there is a strong correlation between site popularity and the inter-site engagement in the network. In this section, we demonstrate the effect of the network (the traffic between sites) on site engagement. We show that the traffic between sites affects the site popularity, and even slightly the activity on sites. Afterwards, we identify patterns that describe how the traffic is distributed over the network.

We use the daily US networks ($US_{01/08}$, …, $US_{31/07}$) and removed networks modeling weekend browsing behaviors. This is to ensure that our observations are not caused by the difference in browsing behaviors between weekdays and weekends (see Section 5.2).

6.1 Dependencies Between Sites

We start by investigating the dependencies between sites in the provider network, to demonstrate the extent of the network effect. We want to see whether sites change their daily popularity (activity) in the same way. Based on the remaining 261 networks, we represent the daily popularity (activity) of a site by a vector $v_n = (c_1, \ldots, c_{261})$, where c_i is the number of sessions (average dwell time per session) on day i, for site n. We then compare the sites by calculating the Spearman rank correlation coefficient ρ between its vectors. Only statistically significant correlations are reported (p value < 0.01). A high positive or negative correlation between two sites indicates that changes in the network are affecting both sites.

We study the strength of the network effect by grouping sites that are affected in the same way by changes in the network. We group all sites that have correlations above or below a given threshold θ. For instance, if $|\rho(a, b)| \geq \theta$, and $|\rho(a, c)| \geq \theta$, we create a group containing the sites $\{a, b, c\}$. We continued this process until all sites were compared with each other. Our results show that there is one large group, both in terms of popularity and activity of sites. Figure 8a shows the number of sites belonging to the largest group for increasing values of θ.

As expected, the size of the largest group decreases by increasing θ. However, when looking at the number of sessions, we still observe that 43.84% of the sites affect each other with $\theta = 0.7$ (i.e., only considering correlations that satisfy $|\rho| \geq 0.7$). We can also report that only 2.93% of the correlations are negative ($\rho \leq -0.7$). This means that there is a significant positive network effect in terms of site popularity; sites become more *or* less popular together.

The effect on the activity metric *DwellTime* is weaker. Only 14.67% of the sites belong to the largest group with $\theta = 0.7$. This implies that the activity on a site depends more on the site itself (e.g., users always spend more time on mail, but less on search). This was already observed in Lehmann et al. (2012). However, in this case 50% of the correlations are negative ($\rho \leq -0.7$). This shows that there are negative dependencies with respect to the time users spend on sites: an increase of dwell time on one site often leads to a decrease of dwell time on another site. We hypothesize that users have a limited amount of time to spent *on* the network *instead* of per site. Therefore, users switch from one site to another site (e.g., from Yahoo Sport to Yahoo Finance) within that limited time, thus more time spent on one site means less time spent on another site of the network.

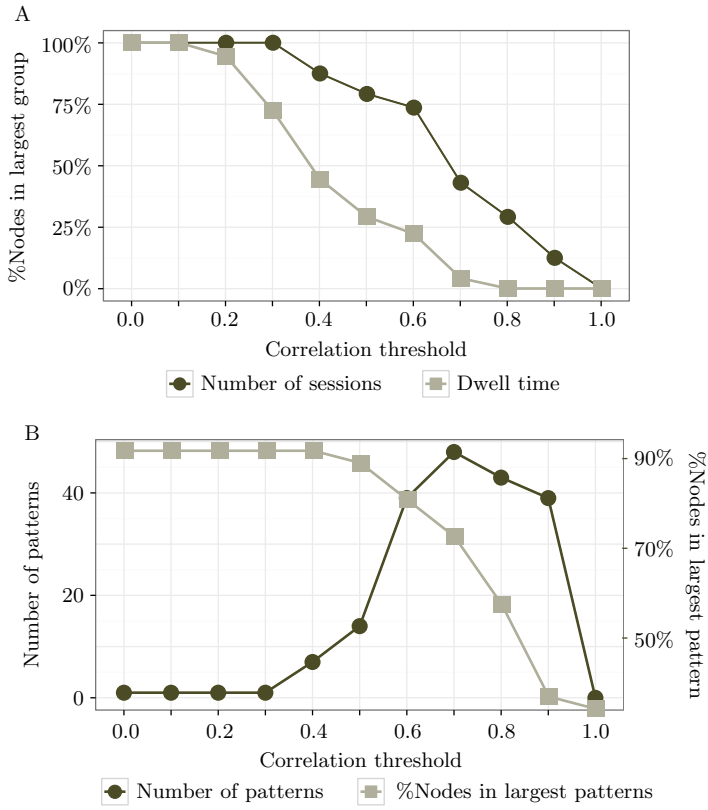


FIGURE 8 (a) Network effect. (b) Network effect patterns. The strength of the network effect and corresponding patterns generated at different correlation thresholds.

6.2 Network Effect Patterns

We now study how the traffic between sites affects the site engagement. In other words, we analyze the spread of traffic through the network. We do so by looking at the dependencies between the edges as opposed to between the sites (as presented in the above section) to identify *network effect patterns* that describe how sites in the network exchange traffic with each other.

Similar to the first part in this section, we characterize the daily popularity of an edge by a vector $v_e = (c_1, \ldots, c_{261})$, where c_i is the number of clicks on day i, for edge e. We then compare the edges by calculating the Spearman rank correlations ρ between its vectors.[8] Finally, if the correlation is above or below a

8. We excluded all edges with less than 30 clicks, to avoid the effect of minor fluctuations (e.g., [1,1,2] and [10,10,200] would have a correlation of 1). Using a threshold of 20 or 40 yields to similar results.

given threshold θ, we says that the two edges are *related* in terms of the relative amount traffic passing by them.

Based on the resulting correlations, several patterns can be identified. If we observe a correlation $|\rho| \geq \theta$ between the edges $a \rightarrow b$ and $a \rightarrow c$, we create a "network effect pattern" containing the sites $\{a, b, c\}$ and the two edges. The pattern reflects that site a forwards traffic to site b and c. If there is also a correlation between $a \rightarrow c$ and $a \rightarrow d$, we add the site d and the edge $a \rightarrow d$ to that same "network effect pattern." We continue this process until all edges are compared with each other, and select all patterns that consist of at least three sites. We note that this approach enables us to analyze different network effect patterns involving the same site. For instance, if there is a correlation between $a \rightarrow b$ and $a \rightarrow c$, and $a \rightarrow d$ and $a \rightarrow e$, two "network effect patterns" can be observed. One pattern shows how site a forwards traffic to sites b and c, and the other one represents how the same site a directs traffic to sites d and e.

We study the number of patterns and the percentage of sites in the largest pattern for various threshold values θ (see Fig. 8b). By increasing the threshold, the number of identified patterns increases, but the size of the largest pattern decreases. This means that parts of the largest pattern are divided into smaller patterns. We observe a peak of 48 patterns at $\theta = 0.7$. However, even with a threshold of $\theta = 0.7$, 71% of the sites are part of the largest pattern. We can conclude again that the network effect is significant. Changes in the network (e.g., increase of popularity of a site) affect many sites and the traffic between them, as shown by the largest network effect pattern. However, there are also smaller patterns that describe the network effect to smaller groups of sites. We can also report that the network effect is mainly positive (i.e., varies in the same direction), as only 1.3% of the correlations are below -0.7.

6.3 Examples of Patterns

We focus on the patterns with correlations $|\rho| \geq 0.7$. We divide the patterns in three groups, shown in Fig. 9. For each, we report the average number of sites, the average reciprocity, and the average transitivity (see Newman, 2003). We recall that the reciprocity describes the probability that the traffic between two sites flow in both directions, whereas the transitivity corresponds to the probability that two randomly selected neighbors of a site exchange traffic with each other. A low transitivity indicates that the pattern has a star-like structure; one site exchanges traffic with many other sites, and the other sites do not exchange traffic directly. A high transitivity reflects that all sites exchange traffic with each other. We refer to this as cluster-like structure.

The first two groups consist of network effect patterns with a star-like structures ($Trans = 0$). Simple star-like patterns have one focal site responsible for most traffic exchange, whereas complex star-like patterns have more than one focal site. In Fig. 9, we see three examples of simple star-like (1,2,3) and two examples of complex star-like patterns (4,5). Surprisingly, other sites than

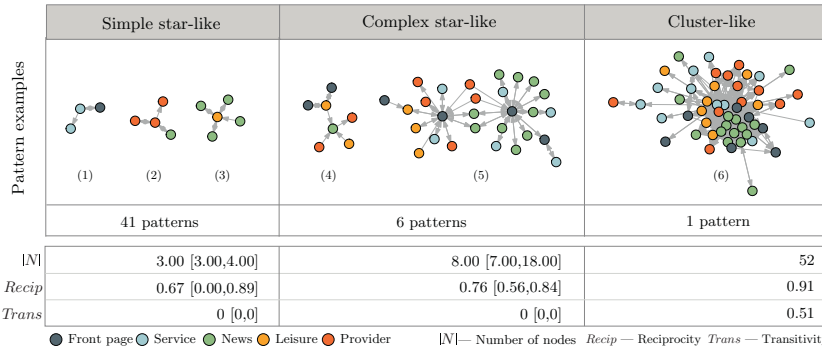


FIGURE 9 (First row) Examples of network effect patterns. (Second row) Number of patterns belonging to that group. (Third row) Median of interquartile range of the number of sites ($|N|$), reciprocity (*Recip*), and transitivity (*Trans*) in each group.

front pages are responsible for the traffic exchange. In our examples, service (1), provider (2), leisure (3,4), and news sites (4) are focal sites.

In addition, focal sites are not necessarily connected with each other. In example (5), there are two provider sites that inject traffic to the focal sites (i.e., two edges of the type $\{provider\} \rightarrow \{frontpage\}$), and two news sites that exchange traffic with the focal sites (i.e., two edges of the type $\{news\} \leftrightarrow \{frontpage\}$). We also observe that the traffic often flows in both directions between two sites (e.g., complex star-like patterns have a median reciprocity of 0.76). This suggests that if a focal site increases the traffic to other sites, it is very likely that the other sites are also returning more traffic back.

On the right side of the figure, we see the largest network effect pattern, containing 52 sites in total. This pattern has a cluster-like structure; many sites exchange traffic directly with each other ($Trans = 0.52$), and also in both directions ($Recip = 0.91$). All site categories (e.g., mail, service, leisure) exchange traffic with each other.

In conclusion, the extent of the network effect in a provider network is significant; the traffic in the network affects the engagement of many sites. Next, we look at whether and how hyperlinks between the sites of a provider network influence this.

7 HYPERLINK PERFORMANCE

Yom-Tov et al. (2013) demonstrated that hyperlinks help directing users to other sites in a provider network. Motivated by this, we analyzed the different types of links on the sites of a provider network, and whether these influence the inter-site and site engagement.

For each site from the US provider network, we selected a random sample of pages accessed during February 2014 (US_{Feb}). We only considered pages

that were accessed at least 10 times. In total, we sampled 43K pages. We downloaded the HTML content of the pages and extracted their hyperlinks.[9] We distinguished whether a link[10] points to a page within the provider network (*internal link*), or to somewhere else on the Web (*external link*). For the first case, we also differentiated between links to pages within the same site (*on-site links*), and links to pages to other sites (*inter-site links*) of the provider network. For each site, we then calculated the average percentage of on-site, inter-site, and external links per page. We did not consider the position and style of hyperlinks (we leave this for future work); our focus was the relationship between links and inter-site engagement.

7.1 Variations in the Link Structure

We first study whether sites differ in their hyperlink structure. Figure 10 shows the distribution of on-site, inter-site, and external links per site category. We report the median values.

Front pages have the highest percentage of inter-site links (62.1%). This is to be expected as they are used to access other sites in the network. However, the percentage of external links is also the highest compared to the other site categories (27.5%). A manual inspection shows that front pages are also used to direct users to sites outside the provider network. There are pages linking to Yahoo sites of other countries (e.g., everything.yahoo.com/world), or to sites of partnership providers (e.g., att.yahoo.com).

With service and news sites, on-site and inter-site links exist in the same frequency. Sites of both categories have around 40% on-site and around 40% inter-site links. This results in 20% of external links.

The highest on-site connectivity is given by leisure sites (68.11%). The on-site and inter-site hyperlink structure differs significantly among leisure sites. The interquartile range is between 38.5% and 90.1%, and 3.9% and 44.9%, respectively. Some leisure sites have many links between their pages, whereas others have many links to other sites in the provider network. However, all leisure sites do not link much to sites outside the provider network (8.7%).

Finally, provider sites do not have many in-site links, as many of them consists only of a few pages (e.g., info.yahoo.com). We observe as well that some provider sites have a high percentage of external links (the interquartile range is between 9.9% and 34.6%). These provider sites are also used from non-US users as an entry point to Yahoo (e.g., messenger.yahoo.com) and as such they link to the sites users are searching for (e.g., fr.messenger.yahoo.com for users from France).

9. Pages from several sites were not considered, as signing in on the sites was required before downloading the pages.
10. We use link and hyperlink interchangeably.

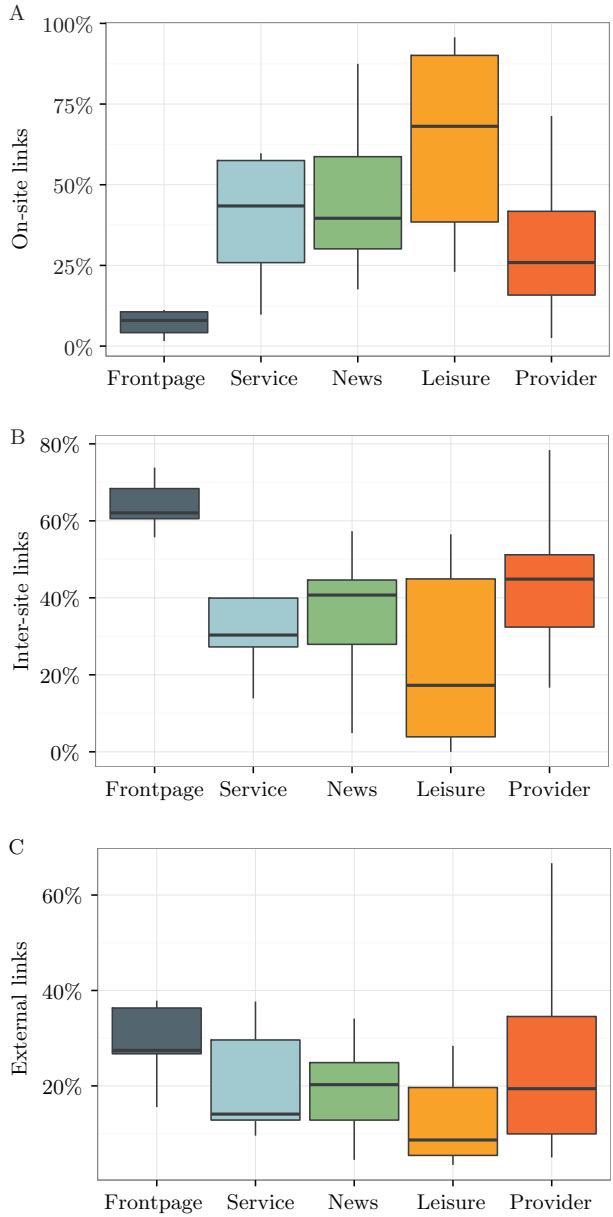


FIGURE 10 (a) On-site links. (b) Inter-site links. (c) External links. Percentage of link types depending on site category.

Overall, this section shows variations in the link structure of a provider network, such as Yahoo US. We investigate next whether the link structure of a provider network has an effect on inter-site engagement.

7.2 Effect of the Link Structure

We investigate how the hyperlink structure of the provider network affects the browsing behavior of the users within the network. We model sites (nodes) and hyperlinks (edges) between them to form a *hyperlink network*. The edge weight is defined by the number of hyperlinks from one site to another.

We compare the hyperlink with the traffic network using the node-level metrics *PageRank* and *Downstream*.[11] Since the site engagement metrics cannot be employed on the hyperlink network, we also investigate how the composition of external, on-site, and inter-site links of a site affects the browsing behavior of users when visiting that site. The browsing behavior on a site in the traffic network is described by the average percentage of traffic to pages of the same site (*on-site traffic*), to other sites of the provider network (*inter-site traffic*), or to somewhere else (*external traffic*).[12] We then rank sites according to each measure in the traffic and hyperlink network and compare these rankings using the Spearman rank correlation coefficient ρ (p value < 0.01). The results are presented in Table 6.

We observe in Table 6a that the importance of sites measured by *PageRank* is similar in both networks ($\rho = 0.54$). This suggests that if many hyperlinks lead to a certain site, it is also likely that users will visit that site. Interestingly, if a site has a high downstream engagement in the hyperlink network, it does not

TABLE 6 Spearman's Correlation Between Site Metrics Using the Link and Traffic Network

(a) PageRank and Downstream

	Traffic	
	PageRank	Downstream
Hyperlinks		
PageRank	0.54	–
Downstream	–	–

(b) On-site, Inter-site, and External

	Traffic		
	On-site	Inter-site	External
Hyperlinks			
On-site	0.54	−0.45	−0.38
Inter-site	−0.40	0.50	–
External	–	–	0.39

In case the p value is above 0.01, we do not report the correlation (–).

11. The exit probability in the hyperlink network is defined as the percentage of external links.
12. The percentage of external traffic is the same as the node-level metric *ExitProb*.

imply that users also navigate deeply into the network when visiting that site (p value > 0.01).

In Table 6b, we can see that the likelihood that a user continues browsing within the same site depends on the percentage of on-site links ($\rho = 0.54$). At the same time, a high percentage of on-site links leads to less navigation between sites in the provider network, and to external sites ($\rho = -0.45$ and $\rho = -0.38$, respectively). On the other hand, sites with many links to other sites in the network have a high inter-site engagement ($\rho = 0.50$) and a low site engagement ($\rho = -0.40$); users navigate frequently to other sites in the network, but dwell less on the site under consideration.

External links do not influence the site and inter-site engagement, but we observe a weak correlation to the external traffic ($\rho = 0.39$). This suggests that providing external links leads to more users leaving the network. However, as we observed in Section 5.3, leaving the network does not necessarily imply less engagement, as users often return to the network within the same session.

In conclusion, whereas downstream engagement differs between the hyperlink and traffic network, the page rank applied to the hyperlink network can be used to identify sites that are also visited frequently by users. Moreover, providing more inter-site links and thus encouraging users to visit other sites in the provider network has a positive effect on inter-site engagement. These findings align with those reported in Yom-Tov et al. (2013). However, in doing so, the site engagement decreases which suggests that users only have a certain amount of time when being online. They use this time either mostly on one site, or across several sites within the network. This shows that there are dependencies between the inter-site and site engagement, and increasing both at the same time is a complex challenge.

8 CONCLUSIONS

This chapter proposed a methodology to study user engagement in a network of sites. We referred to this type of engagement as *inter-site engagement*. Large internet companies (e.g. AOL, Google, Yahoo) operate a network of sites, offering a variety of services, ranging from shopping to news. We model sites (nodes) and user traffic (edges) between them as a network, and employ metrics (at network and node level) from the area of complex graph analysis in conjunction with standard engagement metrics to study inter-site engagement. This enables us to analyze user engagement at a large-scale while accounting for the relationship between sites.

Five network-level and four node-level metrics were used to capture the inter-site engagement within the whole provider network and for individual sites, respectively. In addition to these graph metrics, we defined new metrics to study specific properties of inter-site engagement. For instance, we defined a flow metric to measure the extent to which users navigate between sites in the network. We also employ a metric that measures the downstream engagement,

that is, how deeply users browse into the network after visiting a site. This metric enables us to identify sites that maximize the inter-site engagement. This is particularly important to know for front pages, as linking to them might increase the engagement to the network. We referred to all these metrics as *inter-site* metrics. We demonstrated the value of these metrics by performing several case studies.

First, we compared inter-site metrics with standard engagement metrics. This brought various insights about inter-site engagement, which would not have been possible with standard engagement metrics alone. For instance, we observed that frequently visited sites lead users to access other sites in the provider network. Using network-level metrics, we showed that whole provider networks differ in their inter-site engagement. We could also identify networks that are highly engaging (users spend a lot of time in the network), but where the inter-site engagement is low (users do not visit many sites). In addition, we used node-level metrics to identify typical engagement patterns. We could show that users who visit the network for leisure activities stayed mostly on one site, whereas users interested in reading news visited several news sites.

We also analyzed how returning traffic (users leaving the network but returning within the same session), user loyalty, and other aspects affect inter-site engagement. We saw that leaving the provider network does not necessarily entail less engagement, as many users return later on. As already observed in Lehmann et al. (2013), users often switch between sites and thereby access sites several times within an online session, effectively engaging in online multitasking. This suggests that providers should rethink about their "user engagement" strategy, which often comes down to keeping users as long as possible on their sites. Instead, it may be beneficial (long-term) to entice users to leave the network (e.g., by offering them interesting off-network content in the context of news sites) in a way that users will want to return to it (e.g., become a reference site).

We investigated the dependencies between site engagement and traffic between sites in the provider network, which we call the *network effect*. We showed that there is a strong network effect with respect to site popularity, i.e., changes in the network affect the traffic (on edges) and hence many sites in the network. Although the activity on a site depends more on the site itself, we still observed that an increase in activity of one site can lead to a decrease in activity on another site. This suggests that users will very often only have a limited amount of time when online. If they have to visit several sites on the network, they are likely to do so quickly, so that to keep within their available time.

Finally, we compared the traffic and hyperlink network with each other, and showed that hyperlinks can influence the user browsing behavior in the network. Whereas the downstream engagement in the two networks did not align, the importance of sites measured by *PageRank* is similar in both networks; if many hyperlinks lead to a certain site, it is also likely that users will visit that site. We also found that more hyperlinks between sites lead to a higher inter-site

engagement (users access more sites), but to a lower engagement on sites (users spend less time on sites). This means that site and inter-site engagement influence each other, and improving both at the same time may be difficult.

9 FUTURE WORK

Several lines of research can be pursued. We looked at the effect of several dimensions on inter-site engagement separately. An important extension of our work will be to combine these dimensions, for instance, studying the behavior of loyal users in each country. Other dimensions should also be considered, including demographics and in particular finer grained time analysis, such as morning versus evening.

We also did not differentiate between the ways users navigate between sites, whether clicking on hyperlinks, using browser tabs, or bookmarks. We could build additional types of traffic networks that, for instance, account only for the traffic produced by clicking on hyperlinks and compare it with the hyperlink network. This can bring new insights into how hyperlinks influence inter-site engagement, and which are the hyperlinks that are important in directing traffic to other sites. In this context, one could also analyze how the position and style of hyperlinks contribute to this.

ACKNOWLEDGMENTS

This work was partially funded by Grant TIN2012-38741 (Understanding Social Media: An Integrated Data Mining Approach) of the Ministry of Economy and Competitiveness of Spain. This work was carried out as part of Janette Lehmann's PhD internship at Yahoo Labs, Barcelona. We thank the Yahoo Toolbar users for providing their browsing log data.

REFERENCES

Barabási, A.-L., Albert, R., Jeong, H., 2000. Scale-free characteristics of random networks: the topology of the world-wide web. Phys. A Stat. Mech. Appl. 281 (1), 69–77.

Beauvisage, T., 2009. The dynamics of personal territories on the web. In: Proceedings of the 20th ACM Conference on Hypertext and Hypermedia (HT), pp. 25–34.

Bucklin, R.E., Sismeiro, C., 2003. A model of web site browsing behavior estimated on clickstream data. Mark. Res. 40 (3), 249–267.

Catledge, L.D., Pitkow, J.E., 1995. Characterizing browsing strategies in the World-Wide Web. Comput. Netw. ISDN Syst. 27 (6), 1065–1073.

Chen, W., Wang, Y., Yang, S., 2009. Efficient influence maximization in social networks. In: Proceedings of the 15th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 199–208.

Cheng, S., Shen, H., Huang, J., Zhang, G., Cheng, X., 2013. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In: Proceedings of the 22nd ACM Conference on Information and Knowledge Management (CIKM), pp. 509–518.

Chmiel, A., Kowalska, K., Hołyst, J.A., 2009. Scaling of human behavior during portal browsing. Phys. Rev. E 80 (6), 066122.

Cockburn, A., McKenzie, B., 2001. What do web users do? An empirical analysis of web use. Int. J. Hum. Comput. Stud. 54 (6), 903–922.

Dellarocas, C., Katona, Z., Rand, W., 2013. Media, aggregators, and the link economy: strategic hyperlink formation in content networks. Manag. Sci. 59 (10), 2360–2379.

De Maeyer, J., 2010. Links and journalism: what is at stake? http://juliettedm.wordpress.com/2010/07/12/links-and-journalism-what-is-at-stake/.

Dupret, G., Lalmas, M., 2013. Absence time and user engagement: evaluating ranking functions. In: Proceedings of the 6th ACM Conference on Web Search and Data Mining (WSDM), pp. 173–182.

Freeman, L.C., 1979. Centrality in social networks conceptual clarification. Soc. Netw. 1 (3), 215–239.

Jiang, Q., Tan, C.-H., Wei, K.-K., 2012. Cross-website navigation behavior and purchase commitment: a pluralistic field research. In: Proceedings of the 16th Pacific Asia Conference on Information Systems (PACIS), p. 193.

Johnson, E.J., Moe, W.W., Fader, P.S., Bellman, S., Lohse, G.L., 2004. On the depth and dynamics of online search behavior. Manag. Sci. 50 (3), 299–308.

Kempe, D., Kleinberg, J., Tardos, É., 2003. Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 137–146.

Koidl, K., Conlan, O., Wade, V., 2014. Cross-site personalization: assisting users in addressing information needs that span independently hosted websites. In: Proceedings of the 25th ACM Conference on Hypertext and Hypermedia (HT), pp. 66–76.

Kumar, R., Tomkins, A., 2010. A characterization of online browsing behavior. In: Proceedings of the 19th Conference on World Wide Web (WWW), pp. 561–570.

Lehmann, J., Lalmas, M., Yom-Tov, E., Dupret, G., 2012. Models of user engagement. In: Proceedings of the 20th Conference on User Modeling, Adaptation, and Personalization (UMAP), pp. 164–175.

Lehmann, J., Lalmas, M., Dupret, G., Baeza-Yates, R.A., 2013. Online multitasking and user engagement. In: Proceedings of the 22nd ACM Conference on Information and Knowledge Management (CIKM), pp. 519–528.

Liu, Y., Gao, B., Liu, T.-Y., Zhang, Y., Ma, Z., He, S., Li, H., 2008. Browserank: letting web users vote for page importance. In: Proceedings of the 31st ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 451–458.

Meiss, M.R., Menczer, F., Fortunato, S., Flammini, A., Vespignani, A., 2008. Ranking web sites with real user traffic. In: Proceedings of the 1st ACM Conference on Web Search and Data Mining (WSDM), pp. 65–76.

Meiss, M.R., Gonçalves, B., Ramasco, J.J., Flammini, A., Menczer, F., 2010. Agents, bookmarks and clicks: a topical model of web navigation. In: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia (HT), pp. 229–234.

Newman, M.E., 2003. The structure and function of complex networks. SIAM Rev. 45 (2), 167–256.

O'Brien, H.L., Toms, E.G., 2008. What is user engagement? A conceptual framework for defining user engagement with technology. J. Am. Soc. Inform. Sci. Technol. 59 (6), 938–955.

Page, L., Brin, S., Motwani, R., Winograd, T., 1999. The pagerank citation ranking: bringing order to the web. Stanford InfoLab.

Park, Y.-H., Fader, P.S., 2004. Modeling browsing behavior at multiple websites. Mark. Sci. 23 (3), 280–303.

Peterson, E.T., Carrabis, J., 2008. Measuring the immeasurable: visitor engagement. Web Analytics Demystified.

Roos, J.M., 2012. Hyper-media search and consumption. Ph.D. thesis, Duke University.

Simkin, M., Roychowdhury, V., 2008. A theory of web traffic. Europhys. Lett. 82 (2), 28006.

Squartini, T., Picciolo, F., Ruzzenenti, F., Garlaschelli, D., 2013. Reciprocity of weighted networks. Sci. Rep. 3. http://www.nature.com/srep/2013/130923/srep02729/full/srep02729.html?message-global=remove.

The PEW Research Center, 2010. Understanding the participatory news consumer. http://www.pewinternet.org//media/Files/Reports/2010/PIP_Understanding_the_Participatory_News_Consumer.pdf.

The PEW Research Center, 2012. In changing news landscape, even television is vulnerable. http://www.people-press.org/files/legacy-pdf/2012NewsConsumptionReport.pdf.

Trevisiol, M., Chiarandini, L., Aiello, L.M., Jaimes, A., 2012. Image ranking based on user browsing behavior. In: Proceedings of the 35th ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 445–454.

Trevisiol, M., Aiello, L.M., Schifanella, R., Jaimes, A., 2014. Cold-start news recommendation with domain-dependent browse graph. In: Proceedings of the 8th ACM Conference on Recommender Systems (RecSys), pp. 81–88.

Wasserman, S., 1994. Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press.

Wu, X., Yu, K., Wang, X., 2011. On the growth of internet application flows: a complex network perspective. In: Proceedings of the 30th IEEE Conference on Computer Communications (INFOCOM), pp. 2096–2104.

Xue, W., Shi, J., Yang, B., 2010. X-RIME: cloud-based large scale social network analysis. In: Proceedings of the 7th IEEE Conference on Services Computing (SCC), pp. 506–513.

Yom-Tov, E., Lalmas, M., Baeza-Yates, R., Dupret, G., Lehmann, J., Donmez, P., 2013. Measuring inter-site engagement. In: Proceedings of the IEEE Conference on Big Data (BigData), pp. 228–236.

Chapter 14

Scaling RDF Triple Stores in Size and Performance: Modeling SPARQL Queries as Graph Homomorphism Routines

Vito Giovanni Castellana[*,1], **Jesse Weaver**[*], **Alessandro Morari**[*], **Antonino Tumeo**[*], **David Haglin**[*], **John Feo**[†], **Oreste Villa**[‡]

[*] *Pacific Northwest National Laboratory, Richland, Washington, USA*

[†] *Context Relevant, Seattle, Washington, USA*

[‡] *NVIDIA Research, Santa Clara, California, USA*

[1] *Corresponding author: e-mail: vitoGiovanni.castellana@ppnl.gov*

ABSTRACT

This chapter discusses the approaches integrated in GEMS (Graph database Engine for Multithreaded Systems) for managing and querying datasets of RDF (Resource Description Framework) triples. GEMS is a software stack that implements graph databases on top of commodity, high-performance clusters. GEMS is composed of a SPARQL-to-C++ compiler, a library of data structures and parallel graph methods, and a multithreaded runtime library. Differently from other RDF databases, which resort to more conventional relational databases approaches, and largely employ table-based methods for query processing, GEMS mostly employs graph methods. Query processing in GEMS is performed through the conversion of SPARQL queries into graph homomorphism routines, which then are directly executed on the graph database (resulting from the RDF triples ingestion) through its runtime library. The runtime library, which implements a Partitioned Global Address Space (PGAS), lightweight software multithreading, and network message aggregation, mitigates some of the typical issues of graph processing on modern commodity clusters, enabling scaling in performance and size while new cluster nodes are added. In fact, although very powerful, these systems are built for regular computation and easily partitionable workloads, while graph processing typically has an irregular behavior. The chapter explains how SPARQL queries can be naturally modeled as graph pattern-matching algorithms and details how GEMS performs the conversion

Handbook of Statistics, Vol. 33. http://dx.doi.org/10.1016/B978-0-444-63492-4.00014-9

to C++ routines. It briefly discusses the other components of GEMS and then shows the results of the full stack on the Berlin SPARQL Benchmark (BSBM) and the SPARQL Performance Benchmark (SP2B). We discuss effects of the automatic conversion and present a comparison with a full custom appliance for data analytics (YarcData Urika).

Keywords: RDF, SPARQL, Big data, Graph analytics, In-memory processing, Multithreading

1 INTRODUCTION

Many fields are experiencing an explosion in the availability of data. Areas such as security, communication, transportation and social networks, open government, healthcare, environmental science, biomedical research, and finance are now collecting inordinate amounts of data, with exponentially growing volumes, at increasingly faster rates, and more and more heterogeneous in nature. This creates problems not only in storing and managing, but also in fully exploiting the value of the data itself (i.e., performing queries on data that return relevant results in reasonable times).

The Resource Description Framework (RDF) (Klyne et al., 2004) is a flexible data model originally proposed by the World Wide Consortium (W3C) for the Semantic Web that has recently seen a significant uptake for publishing/integrating the data from all these areas. RDF organizes the data in the form of subject–predicate–object triples. A set of RDF statements naturally maps to directed, labeled graph (Fig. 1). Thus, RDF triples represent a convenient way to organize and store graph databases. An analyst can query a RDF dataset through ad hoc languages such as SPARQL (W3C SPARQL Working Group, 2013). In SPARQL, the fundamental query operation is graph matching. Hence, a SPARQL query on a RDF dataset could potentially be performed through (nested) graph walks.

Graph databases appear well suited to organize the data of these emerging fields. Graph databases exploit graph structures, with nodes, edges and properties to represent and store data. They provide index-free adjacency, meaning that every element contains a direct pointer to its adjacent element. Thus, no index lookups are necessary. Data manipulation is expressed by graph-oriented operations and type constructors. Graph databases well fit these new emerging network-like datasets, which expose a large number of interconnections among their elements. In contrast to relational databases, they can manage ad hoc and changing data with evolving schemas, they can represent and explore more easily the relationships among the data, and they can naturally scale to large datasets, because they do not require space and time expensive join operations.

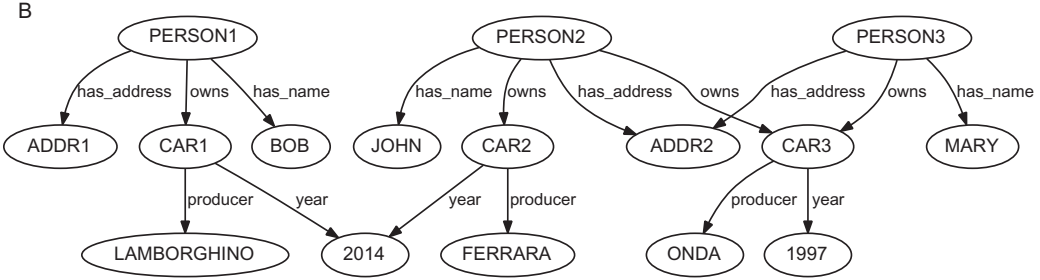
Graph methods based on edge traversal are inherently parallel: a system may generate a concurrent activity for each edge to be traversed. High-Performance Computing clusters are an interesting target platform for in-memory crawling

A

```
PERSON1 has_name BOB .
PERSON1 has_address ADDR1 .
PERSON1 owns CAR1 .
CAR1 producer LAMBORGHINO .
CAR1 year 2014 .
PERSON2 has_name JOHN .
PERSON2 has_address ADDR2 .
PERSON2 owns CAR2 .
PERSON2 owns CAR3 .
CAR2 producer FERRARA .
CAR2 year 2014 .
CAR3 producer ONDA .
CAR3 year 1997 .
PERSON3 has_name Mary .
PERSON3 has_address ADDR2 .
PERSON3 owns CAR3
```

RDF Dataset

B



Graph Representation

FIGURE 1 Example RDF dataset. (a) RDF dataset and (b) graph representation.

of big graphs. They present fast and powerful multicore processors, increasing availably of memory per node, and fast network interconnections. However, they are not perfectly suited for the type of operations performed by graph algorithms. Graph methods generally are irregular: depending on the characteristics of the input datasets, they exhibit poor spatial and temporal locality, perform fine-grained data accesses, generally are memory bandwidth bound and synchronization intensive, and may present load imbalance among parallel tasks. Processors in modern clusters, instead, are designed to reach peak performances with arithmetic/floating-point intensive workloads. They presents deep and complex cache hierarchies and data prefetchers to exploit locality and regular computations, reducing latency of predictable memory accesses. Furthermore, network interconnections of modern clusters reach peak bandwidth only with large, batched data transfers, while they are subject to significant overheads for fine-grained traffic. We have addressed many of these shortcomings by implementing GMT (Global Memory and Threading) (Morari et al., 2014), a runtime library that implements a Partitioned Global Address Space across the distributed memory of a cluster, integrates lightweight software multithreading to tolerate fine-grained data accesses, and provides message aggregation to maximize network bandwidth.

However, GMT only addresses shortcomings of graph explorations (and irregular applications in general) on commodity clusters. On top the runtime layer, we have developed a set of components to implement a complete graph database. This complete software stack is called GEMS (Graph database Engine for Multithreaded Systems) and, beside GMT, it includes a library of data structures and related parallel graph algorithms to manage and explore the graph database (SGLib), and a SPARQL-to-C++ compiler that converts SPARQL queries to graph algorithms. A key difference with respect to other RDF triple stores is in the way that GEMS performs the query processing. Currently existing solutions usually store, retrieve and query triples on top conventional relational databases, or still resort to relational approaches for some of their components. GEMS, instead, primarily uses graph-based algorithms at all levels of the stack. Although, approaching the query processing mainly as a graph-matching problem provides advantages in terms of memory space (because it does not need to generate tables for storing intermediate results while performing conventional joins) and, potentially, in terms of performance (because of the possibility to exploit efficient graph algorithms tailored to the query), novel solutions with respect to the preexisting databases are required to make it possible. From this point of view, the SPARQL-to-C++ compiler plays a crucial role. In fact, it has to provide full coverage of the language (which includes filtering, deduplication, and optional patterns) and generate efficient graph exploration routines. Because it mainly relies on generating graph walks, the selection of the staring points, the ordering of the graph walks, and the selection of the walks to early prune or to continue have a fundamental impact on the performance and on the computational complexity of the query.

In this chapter, we describe GEMS and focus our attention on the SPARQL-to-C++ compiler, detailing how we approach the conversion of SPARQL queries into graph homomorphism routines, and, subsequently, how we can generate C++ code from these. After a brief introduction of the basic concepts of our approach, we describe the GEMS stack and then delve into the details of the compiler. We present an experimental evaluation of our approach and discuss how it is different from other RDF databases. We compare our results, in particular, to YarcData Urika (YarcData, 2014), which is a full appliance (including hardware and software) for data analytics, on two of the reference benchmarks for SPARQL engines: the Berlin SPARQL Benchmark (BSBM) (Bizer and Schultz, 2009) and the SPARQL Performance Benchmark (SP2B) (Schmidt et al., 2010).

2 SPARQL QUERIES AS GRAPH HOMOMORPHISM ROUTINES

SPARQL queries mainly consist of a pattern to match against the RDF data. The information acquired through the matching, typically structured in form of tables, is then processed to construct the final answers. Thus, queries may be modeled as the composition of two main components: a *pattern matching* part and the *solution modifiers*.

2.1 Graph Pattern Matching

The core syntax of SPARQL is a conjunctive set of triple patterns called Basic Graph Pattern (BGP). A triple pattern is similar to an RDF triple, except that any component in the triple pattern can be a variable (identified through a *?* prefix).

BGPs basically describe subgraphs to match against the RDF data. Thus, we can fully consider query processing as pattern-matching problem. Typical techniques to solve this class of problems are based on the (*structural*) subgraph isomorphism algorithm introduced in Ullmann (1976). The basic approach enumerates all the possible mappings of the vertices in the graph pattern (P) onto those in the graph-data (D) through a Depth First (DF) tree search. A path from root to leaves in the DF tree denotes a complete mapping (Fig. 6). If all vertices that are neighbors in the path are also neighbors in both P and D (i.e., they preserve adjacency), then the path represents a legal match. The resulting solution space has exponential complexity; however, some subtrees in the DF tree that do not lead to a feasible mapping can be pruned following three criteria:

1. *Vertex degree*. Given two nodes $v_P \in P$ and $v_D \in D$, if $\textit{out-degree}(v_P) > \textit{out-degree}(v_D)$, then v_D cannot map node v_P.
2. *One-to-one mapping*. Each node of G cannot map more than one node in P.
3. *Forward checking*. Given that node $v_P \in P$ is mapped on $v_D \in D$, if such mapping cannot satisfy adjacency then the subtree rooted in v_D is pruned.

Even when pruning the DF tree, the algorithm has an exponential worst-case time complexity. However, the particular structure of RDF queries and data allow a substantial pruning of the search tree, making the adoption of similar pattern-matching algorithms feasible and profitable in practice, also when scaling the size of both P and D. First, both P and D are directed graphs (see Fig. 6). Second, the data graph D is valued, and any element (subject, predicate or object) of each triple in the BGP may be valued as well. Performing value checks while exploring the solution space dramatically reduces the number of legal mappings. Finally, languages such as SPARQL allow specifying filtering constraints, i.e., conditions that must be satisfied for a match to be legal. An early evaluation of such conditions allow further pruning of the solution space. In GEMS, BGP matching is performed by exploring the solution tree in Breadth First order. The traversal is composed of a sequence of *Graph Walk* Operations (GWOs), each associated with a particular triple pattern, and with a specific level of the solutions tree. In the simplest case, queries feature a single BGP, as in the example proposed in Fig. 2. However, SPARQL allows describing more complex patterns, obtained by combining BGPs. The most common operators are:

- *UNION*. SPARQL allows specifying matching alternatives through the UNION operator, as shown in Figs. 3 and 4. Patterns in union may be matched separately, as distinct graph walks; however, if they have triple patterns in common, these may be matched just once.
- *OPTIONAL*. Graph patterns may include optional components. Given the pattern P_A *OPTIONAL*$\{P_B\}$, if P_A matches and P_B does not, then P_A is still a valid solution, and P_B does not create any binding. This is useful to express queries that allow information to be added to the solution where the information is available, but do not reject the solution because some part of the query pattern does not match. We match patterns such as P_A *OPTIONAL*$\{P_B\}$ through two different graph walks. The first one matches P_A, generating *partial* bindings. Then, a second graph walk resumes the first one: when

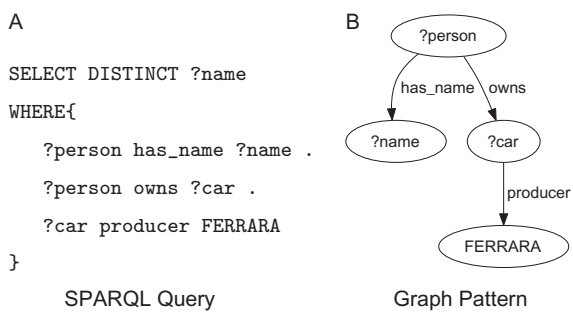


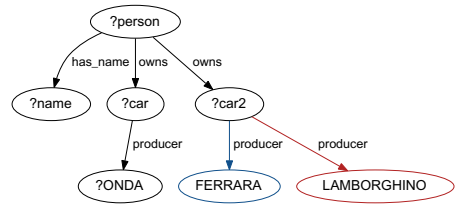
FIGURE 2 Example query Qa: *names of the people owning a FERRARA*. (a) SPARQL query and (b) graph pattern.

A

```
SELECT DISTINCT ?name
WHERE{
   ?person has_name ?name .
   ?person owns ?car .
   ?car producer ONDA .
   ?person owns ?car2 .
   {?car2 producer FERRARA}
   UNION
   {?car2 producer LAMBORGHINO}
}
```

SPARQL Query

B



Graph Pattern

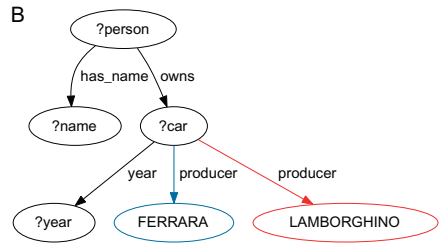
FIGURE 3 Example query Qb: *names of the people owning an ONDA and a FERRARA or a LAMBORGHINO*. (a) SPARQL query and (b) graph pattern.

A

```
SELECT DISTINCT ?name
WHERE{
   ?person has_name ?name .
   ?person owns ?car .
   ?car year ?year .
   {?car producer FERRARA}
   UNION
   {?car producer LAMBORGHINO}
   FILTER (?year>2010)
}
```

SPARQL Query

B



Graph Pattern

FIGURE 4 Example query Qb: *names of the people owning a FERRARA or a LAMBORGHINO, produced after 2010*. (a) SPARQL query and (b) graph pattern.

a match is found, the corresponding partial binding is marked. The overall solution is obtained as the union between the complete matches (both P_A and P_B) and the unmarked partial bindings.

- *NEGATION*. Negation allows specifying patterns that should not match for a solution to be valid. In SPARQL, negation can be expressed in several ways, e.g., with a *NOT EXISTS* operator or through *closed world negation*. The latter is obtained in patterns such as P_A *OPTIONAL*$\{P_B\}$ *FILTER*$(!bound(x))$,

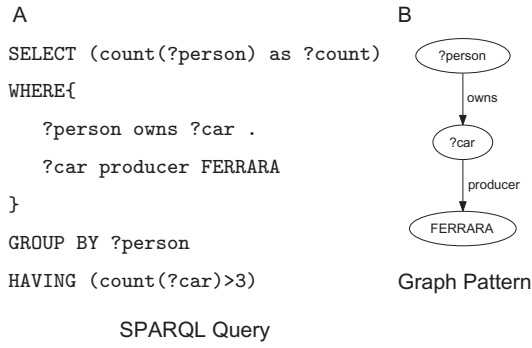


FIGURE 5 Example query Qb: *number of persons owning at least three FERRARAs*. (a) SPARQL query and (b) graph pattern.

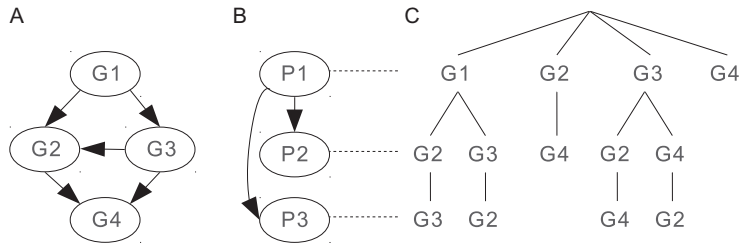


FIGURE 6 Example data graph D (a), pattern graph P (b), and search tree for matching P against D (c). Paths from root to leaves represent valid matches.

where x is a variable appearing in P_B but not in P_A. The matching procedure is composed of two phases also in this case, like in optional patterns. The second graph walk is associated with the pattern subject to negation. As in the previous case, it is used to mark partial solutions. In this case, however, the final matching solution is obtained by discarding marked partial solutions.

2.2 Solution Modifiers

Solution modifiers process the information retrieved through pattern matching in order to construct the final answers. These include aggregation (Fig. 5), projection, distinct, order, limit, and offset. In our system, we organize matching results in tables. For this reason, we refer to solution modifiers, as well as to other operator, which manipulates tables, as *Table Operations* (TOs). As detailed in Section 3.2, the library layer of GEMS (SGLib) provides efficient parallel implementations of such TOs, which can be considered as building blocks: queries are implemented by mapping SPARQL constructs onto GWOs and TOs, and by composing them (Fig. 7).

A

```
1  f3(person, has_name, name){
2    results.insert({name});
3  }
4  f2(person, owns, car){
5    forEach(person, has_name, ?name, f3);
6  }
7  f1(car, producer, ferarra){
8    forEach(?person, owns, car, f2);
9  }
10
11 query_Qa(){
12   //Graph Walk
13   forEach(?car, producer, ferrara, f1);
14   results.distinct();
15 }
16
137
```

Qa

B

```
1  f5(person, producer, car_prod, args){
2    results.insert({args.name});
3  }
4  f4(person, owns, car2, args){
5    forEach(?car2, producer, FERRARA, args, f5);
6    forEach(?car2, producer, LAMBORGHINO, args, f5);
7  }
8  f3(person, has_name, name){
9    args={name};
10   forEach(person, owns, ?car2, args, f4);
11 }
12 f2(person, owns, car){
13   forEach(person, has_name, ?name, f3);
14 }
15 f1(car, producer, onda){
16   forEach(?person, owns, car, f2);
17 }
18
19 query_Qb(){
20   //Graph Walk
21   forEach(?car, producer, onda, f1);
22   results.distinct();
23 }
```

Qb

C

```
1  f4(person, has_name, name){
2    results.insert({name});
3  }
4  f3(person, owns, car){
5    forEach(person, has_name, ?name,f4);
6  }
7  f2(car, yr, year, f3){
8    if(year>2010)
9      forEach(?person, owns, car, f3);
10 }
11 f1(car, producer, car_prod){
12   forEach(car, year, ?year, f2);
13 }
14
15 query_Qc(){
16   //Graph Walk 1
17   forEach(?car, producer, FERRARA, f1);
18   //Graph Walk 2
19   forEach(?car, producer, LAMBORGHINO, f1);
20     results.distinct();
21 }
```

Qc

D

```
1  f2(person, owns, car){
2    results.insert({person});
3  }
4  f1(car, producer, ferarra){
5    forEach(?person, owns, car, f2);
6  }
7
8  query_Qd(){
9    //Graph Walk
10   forEach(?car, producer, ferrara, f1);
11   results.groupBy(?person);
12   results.aggregate(count, ?car, ?count);
13   results.filter(?count>3);
14   results.distinct();
15 }
16
17
18
19
20
21
```

Qd

FIGURE 7 Pseudo code for the pattern-matching routines of example queries Qa, Qb, Qc, Qd. (a) Qa, (b) Qb, (c) Qc, and (d) Qd.

3 GEMS: GRAPH DATABASE ENGINE FOR MULTITHREADED SYSTEMS

Figure 8 schematically shows the structure of the GEMS software stack. As previously stated, GEMS has three main components: the SPARQL-to-C++ compiler, the library of supporting data structures (SGLib, Semantic Graph Library), and the custom Partitioned Global Address Space (PGAS) runtime with lightweight software multithreading (GMT, Global Memory and Threading runtime library). An analyst can use GEMS by loading RDF databases and by writing/compiling SPARQL queries through a web client interface. Nevertheless, users are also allowed to write their custom data analytics applications, using the APIs offered by SGLIB. In addition, the C++ implementation of the queries generated by the query translator is accessible to the users: this potentially allows hand tuning of the application, or introducing custom features not supported by the SPARQL language.

Since each layer of the stack exploits features offered by the following lower level layer, we describe the system bottom up, starting from the GMT runtime system.

3.1 GMT

GMT (Morari et al., 2014) is the underlying runtime library that enables managing and querying the graph database on top of a commodity cluster, hiding most of the complexities. GMT is heavily customized to meet the database requirements. It provides three main features: a virtual global address space that spans the memories of all the nodes in the cluster, lightweight software multithreading and network messages aggregation (a.k.a. coalescing). We designed the runtime from the ground-up focusing on these features, in order to address some of the typical shortcomings exhibited by graph algorithms applied to large graphs on distributed-memory clusters. Exploring large graphs usually generates fine-grained accesses (traversing pointers, or elements of linked-list like data structures) to unpredictable memory locations, especially for graphs that have very high edge degrees. This behavior does not cope well

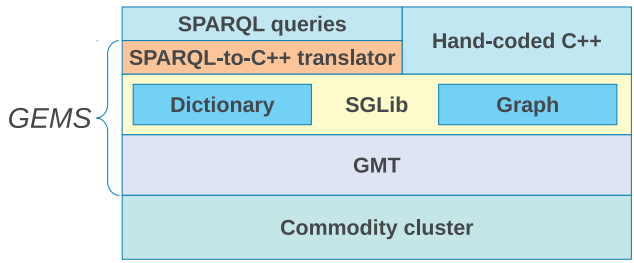


FIGURE 8 Organization of the GEMS (graph database engine for multithreaded systems) layers.

with the processors integrated in modern high-performance computing clusters of commodity nodes, which focus on exploiting locality and regular computation through deep cache hierarchies. Furthermore, the network interconnections of commodity clusters reach their effective peak bandwidth only with large data transfers, while small data transfers usually waste lot of bandwidth for headers and control information. GMT employs lightweight software multithreading to tolerate, rather than reduce, access latency, when an algorithm needs to access data that are stored in the memory of a remote node. Furthermore, GMT aggregates messages directed towards remote nodes to maximize network bandwidth utilization. Partitioning large graph data structures, which do not fit in the memory of a single node, on distributed memory systems is usually difficult. Partitioning algorithms may have higher computational complexity than exploration algorithms, and uniform partitioning is prone to load unbalancing, especially for strongly interconnected graphs. GMT provides a Partitioned Global Address Space (PGAS), which allows treating the distributed memories across the cluster nodes as a virtual shared address space, removing the requirement to partition the data structures. However, it still allows providing locality hints to optimize performance.

3.2 SGLib

SGLib offers several methods for loading/accessing the database, performing graph exploration and manipulating data structures such as Tables. These methods have been implemented exploiting GMT primitives, with the intent of hiding any complexity to the upper levels, or possibly to users who wish to develop their own application.

3.2.1 Graph and Dictionary Representation and Access Methods

The loading of the database consists in an *ingest* phase, which generates a graph and the related dictionary starting from sets of RDF triples (for example, in the N3 format). The dictionary associates string labels with unique integer identifiers (UIDs). Each RDF triple is encoded as a sequence of three UIDs. Each index is range partitioned so that each node in the cluster gets an almost equal number of triples. The graph indexes RDF triples in subject–predicate–object and object–predicate–subject orders. We do not currently use indices for predicates, in order to reduce the memory footprint of the data structure and to scale the size of the input RDF data. SGLib range partitions each index so that each node in the cluster gets an almost equal number of triples. Subject–predicate and object–predicate pairs with highly skewed proportions of associated triples are automatically specially distributed among nodes by exploiting the PGAS features of the GMT runtime, so to avoid load imbalance as a result of data skew. We currently use a simple heuristic for testing skewedness of subject–predicate (respectively object–predicate) pairs. Given the average count and the standard

deviation of the count for such pairs (which we compute automatically before building the indexes), we simply say that a pair that occurs more often than *threshold* = *average* + *stddev* times is highly skewed (although the threshold is configurable). For those pairs, we range partition the associated triples across all nodes. In effect, there is a range partitioning for each highly skewed subject–predicate (resp. object–predicate) pair, and then there is another range partition over all the other triples that are not associated with a highly skewed pair. SGLib exposes *lookup* methods for accessing the dictionary, e.g., for retrieving elements values given their UID, or for UID–UID and UID-value comparison. Graph exploration is supported through the *forEach(S, P, O, [args], CB)* primitive. The primitive spawns several tasks, which match the S–P–O triple in the data (performing value checking, if the elements are valued), and executing one instance of the specified Call Back (CB) function. CB functions may, for example, check for constraint satisfaction, buffer/update data for composing temporary or result tables, proceed the graph walk through further forEach calls. Each call spawns up to one task per matching triple: since the size of the graph in a modern graph databases may well include several billions of edges and/or vertices, there are large degrees of exploitable parallelism. This is actually well suited to the lightweight multithreading software feature of the GMT runtime.

3.2.2 Table Operations

SGLib defines parallel data structures such as arrays and tables and access methods to support query processing. In addition to procedures to buffer, insert or update the content of these structures, SGLib also implements solution modifiers, which mainly act on tables. These perform deduplication (SPARQL construct *D*ISTINCT), sorting (*ORDER BY*), grouping (*GROUP BY*), aggregation (e.g., *count*, *max/min*, *sum*, *avg*), projection (e.g., *LIMIT*, *OFFSET*, *HAVING*), and join of tables.

3.3 SPARQL-to-C++ Compiler

Figure 9 shows the structure of GEMS' SPARQL-to-C++ translation engine, which mainly consists of two phases: *front-end* and *back-end* phases. The front-end is responsible for the definition of an optimized execution plan for the query, exposed to the back-end through a Low Level Internal Representation (LLIR). The front-end is unaware of the underlying layers, producing an IR independent from GEMS APIs. The back-end generates the output C++ code by exploiting SGLIB primitives as building blocks. In turn, input language details are hidden from the back-end. The clear separation between the two components potentially allows supporting multiple input languages/output APIs by modifying only one of them.

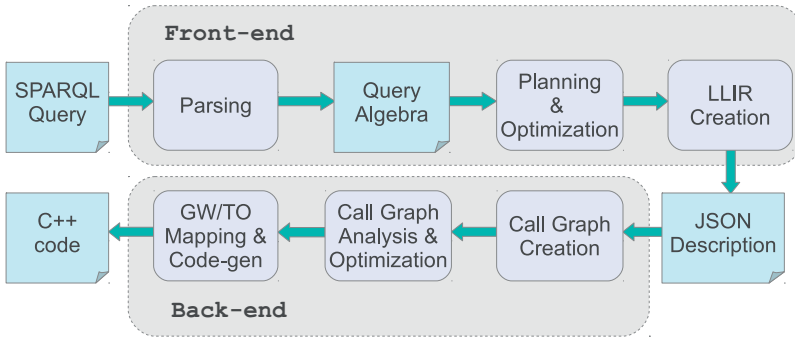


FIGURE 9 Automatic code generation flow.

3.3.1 *Front-end*

The first step of the process is the input query parsing, which generates a tree representation aligned with the SPARQL algebra as defined in the standard. The front-end purposely processes the algebraic representation to construct the query plan. The plan is a Direct Acyclic Graph (DAG) that defines a partial ordering among the basic operations composing the query. The planner identifies the most promising ordering among several candidates, according to a cost model and a cardinality estimator. The optimization process exploits an iterative dynamic planning algorithm for ordering the triple patterns to traverse within a graph walk; in the presence of multiple graph walks, it also identifies whether they share components, allowing matching such patterns only once, within the same graph walk. An example is provided in Figure 10a, which shows a possible query plan for the example query *Qb*. The first four triple patterns are matched once through a single graph walk; after matching the subpattern, the walk proceeds through the two alternative patterns. However, in some cases it results more profitable to perform two completely disjoint graph walks, as for example for query Qc. In this case, the first three patterns may be matched only once; nevertheless, none of these triples feature a valued subject or object. Since in our system we do not build indexes for predicates, starting a graph walk from any of these triple patterns would result in exploring all the graph triples. Thus, a better solution is to perform two distinct graph walks, starting from the alternative triples, which both feature a valued object. The corresponding plan is proposed in Figure 10b. Once the optimal plan is computed, it is represented as a sequence of operations, in JSON (JavaScript Object Notation) format. Operations are characterized by indicating used variables and references to previous operations.

3.3.2 *Back-end*

The backend processes the JSON LLIR to build the Call Graph (CG) of the query procedure. The backend then analyzes the CG to identify:

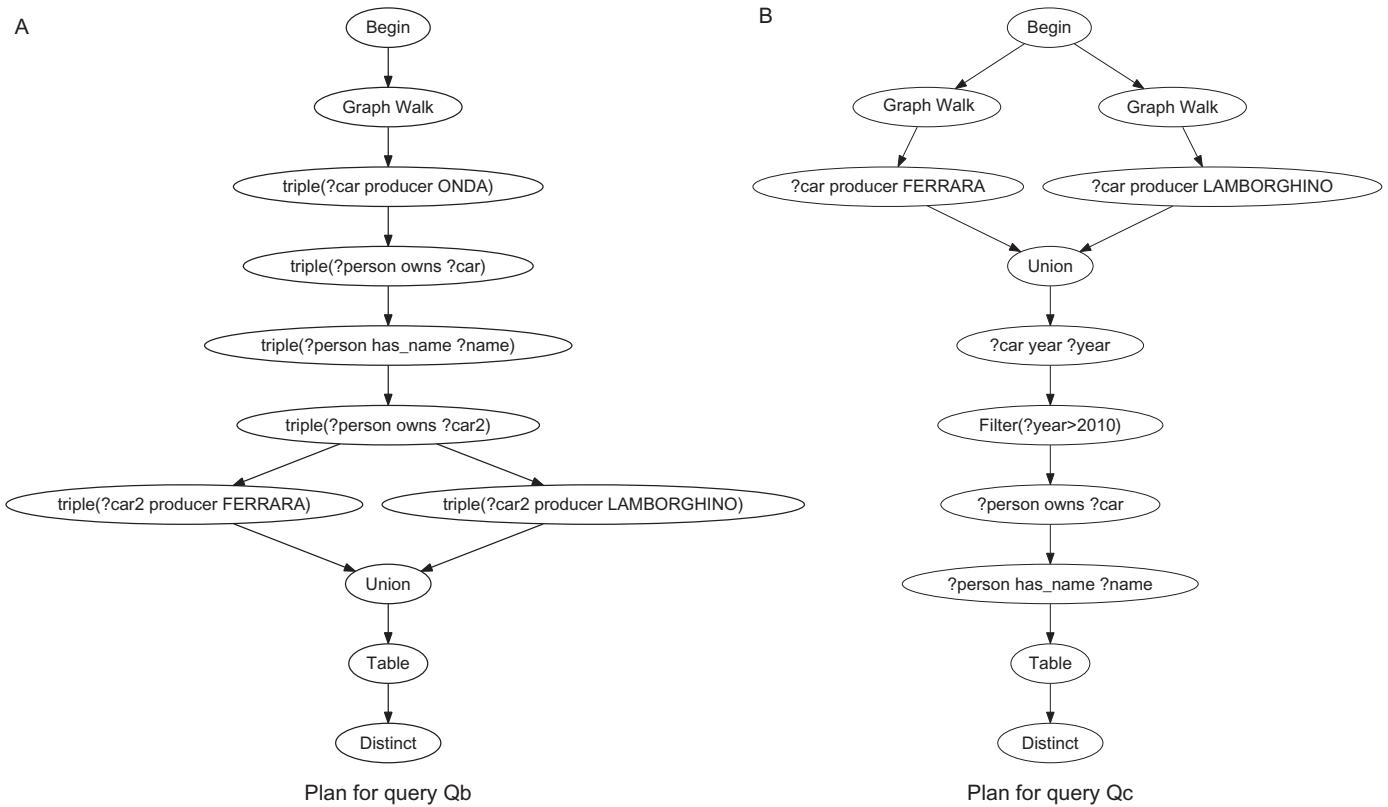


FIGURE 10 Example query plans for queries featuring UNION. (a) Plan for query Qb and (b) plan for query Qc.

- portions of the procedure that may execute in parallel;
- the minimum set of variables that must be carried-on from call to call while performing a graph walk; this reduces the amount of data passed as argument to forEach functions;
- the minimum set of variables that must be stored in tables; this reduces the memory footprint of temporary data structures;
- lifetime of data structures; this allows destroying them as soon as possible, freeing the allocated memory;
- particular sequences of operations, which may be executed as a single SGLIB operation (e.g., filtering and projection of a table).

After the analysis, the backend traverses the CG, mapping each operation to a templated C++ implementation. It finally composes all the generated portions of code, obtaining the final C++ implementation of the query.

4 RELATED WORK

The exponential growth in the availability of data, especially of datasets that have significant amount of relationships among their elements (such as all kinds of networks, from social, to security, to biomedical, or the internet at large itself), has led to the development of a large number of database systems (both commercial and at the research level) able to manage RDF datasets.

Some databases do not support named graphs. Most (if not all) RDF databases support—to varying degrees—the standard SPARQL query language. Open source RDF databases include: Jena SDB (Apache Jena, 2014) backed by relational databases; Jena TDB (Apache Jena, 2014) backed by native, disk-based storage; Sesame (SesameRDF, 2014) with support for layering on top of relational databases or a native backend; Virtuoso Open Source edition (Virtuoso, 2014); and 4store (Harris et al., 2009). Research-level RDF database approaches include: RDF-3X (Neumann and Weikum, 2010), Hexastore (Weiss et al., 2008), YARS2 (Harth et al., 2007), SHARD (Rohloff and Schantz, 2010), BitMat (Atre et al., 2010), and SPARQL queries on PIG (Kotoulas et al., 2012). Commercial RDF databases include: Bigdata (BigDataDB, 2014); BigOWLIM (Ontotext, 2014) with a single-node edition (OWLIM-SE) and a replication cluster edition (OWLIM-Enterprise); Virtuoso (Virtuoso, 2014), which has a cluster edition; 5store (successor to 4store) implemented for clusters; Urika (YarcData, 2014), a data analytics appliance featuring a graph database, built on the Cray XMT.

Solutions that try to implement RDF databases on top of conventional relational SQL-based systems usually incurs in overheads. Obtaining feature-complete SPARQL-to-SQL translation is difficult, and may introduce performance penalties. Translating SPARQL to SQL implies the use of relational algebra to perform optimizations, and the use of classical relational operators (e.g., conventional joins and selects) to execute the query. By translating

SPARQL to graph pattern-matching operations, GEMS reduces the overhead for intermediate data structures and can exploit optimizations that look at the execution plan (i.e., order of execution) from a graph perspective. Many of these solutions rely on front-ends running on Java that generate more or less standardized representations and API calls. GEMS, instead, provides a full, custom, software stack. Queries are translated to efficient C++ code with calls and accesses to fully customized, and optimized, API and data structures. With a typical high-performance computing approach, the code effectively executed by GEMS is compiled, efficient, C++ code.

Some solutions process queries in-memory while others, instead, store data on disks and perform swapping. Jena (Apache Jena, 2014) and Sesame (SesameRDF, 2014), for example, provide libraries that natively implement in-memory RDF storage and support integration with some disk-based, SQL backends. OpenLink Virtuoso (Virtuoso, 2014) implements an RDF/SPARQL layer on top of their SQL-based column store for which multi-node, cluster support is available. GEMS adopts in-memory processing: it stores all data structures in RAM. In-memory processing potentially allows increasing the dataset size while maintaining constant query throughput by adding more cluster nodes. In-memory processing is, in general, the current approach of choice for processing large graphs at the highest speed possible, given the high irregularity of the data structures and the algorithms. Similarly to our system, Trinity.RDF (Zeng et al., 2013), built on top of the Trinity graph processing platform (Shao et al., 2013), operates in memory, processing queries through graph exploration procedures. However, it mainly focuses on matching Basic Graph Patterns; the primitives provided by SGLib instead, allow supporting various forms of SPARQL queries, including solution modifiers or featuring complex graph patterns.

Some of the mentioned approaches leverage Map Reduce frameworks to achieve scalability on clusters. For example, SHARD (Rohloff and Schantz, 2010) is a native triplestore built on top of Hadoop, while YARS2 (Harth et al., 2007) is a bulk-synchronous, distributed, query answering system. Both exploit hash partitioning to distribute triples across nodes. Mapping SPARQL queries to PIG (Kotoulas et al., 2012) abstracts operations over the PIG language, which is compiled into MapReduce jobs. These approaches work well for simple index lookups, but they also present high-communication overheads for moving data through the network with more complex queries, as well as introduce load-balancing issues in the presence of data skew. GEMS abstracts, instead, operations at a lower level, using data structures and primitives from SGLIB, which implement graph primitives and table operations.

4store (and its successor 5store) (Harris et al., 2009) directly interfaces to low-level operations. 4store differentiates processing and storage nodes (although processing backends can execute on the same node with storage backends, they still incur in TCP/IP communication overheads). Segments of the datasets are non-overlapping and uniformly distributed. However, segments

for which there is significant dataskew can be replicated. There is not direct communications among storage nodes. Processing nodes always send a single request at a time to storage nodes. However, since a storage node may host multiple segments, and there is a different connection per segment, a request can be sent to all the segments and then all the replies aggregate. 4store has been demonstrated to scale up to 32 nodes (and the authors used it up to 9 nodes). The commercial successor, 5store, is projected to reach thousands of nodes, although it appears that storage scaling, rather than query throughput, is the objective of such approach. GEMS, instead, adopts a distinct, custom runtimes to implement hardware-specific optimizations to enable scaling in size of the dataset while also scaling performance. The runtime focuses on optimizing critical aspects of the irregular behavior of graph methods applied to large graphs, while also providing the features required for query processing.

Urika is a commercial shared memory system from YarcData (2014) targeted at big data analytics. Urika exploits a Cray XMT 2 machine, which implements custom nodes with purpose-built multithreaded processors (barrel processors with up to 128 threads and a very simple cache). Beside multithreading, which allows tolerating latencies for accessing data on remote nodes, the system has hardware support for a scrambled global address space and fine-grained synchronization. These features allow more efficient execution of irregular algorithms, such as graph methods. On top of this hardware, YarcData interfaces with the Jena framework to provide a front-end API. GEMS, instead, exploits clusters built with commodity components. These are cheaper to acquire and maintain, and evolve more rapidly than custom hardware. On the other hand, the software layers of GEMS have been completely custom built to support query processing. GEMS has a custom compiler and dedicated API and data structures, aiming at providing the most optimized path possible from query entry to result return.

5 EXPERIMENTAL RESULTS

We evaluated GEMS on the Olympus supercomputer at Pacific Northwest National Laboratory's Institutional Computing center. Olympus is a cluster of 604 nodes interconnected through a QDR Infiniband switch with 648 ports (theoretical peak of 4 GB/s). Each of Olympus' nodes features two AMD Opteron 6272 processors at 2.1 GHz and 64 GB of DDR3 memory clocked at 1600 MHz. Each socket hosts eight processor modules (two integer cores, one floating-point core per module) on two different dies, for a total of 32 integer cores per node. A module includes a L1 instruction cache of 64 KB, two L1 data caches of 64 KB, and a 2 MB L2 cache. Each four-module die hosts a shared L3 cache of 8 MB. Dies and processors communicate through HyperTransport. We compared our system against YarcData Urika (YarcData, 2014) (ver. 6.0.4). As previously introduced, Urika is a multi-node system specifically designed for data analytics that integrates the Cray XMT 2 with a database front end based on

the Jena framework. Like in our GMT layer, multithreading in the Cray XMT 2 provides latency tolerance when accessing remote data. The global address space allows the database layer to be implemented as if working on a shared memory system, without caring for partitioning the data or balancing the workload across nodes. Our Urika system features 128 nodes and a total memory of 4 TB. Each node has 32 GB of DDR2 memory with a data rate of 800 MHz. The custom processors in the Cray XMT 2 run at 500 MHz. Because the system exploits a shared memory abstraction, even when Urika executes on a single node, the database and the queries can still use the memory of the entire machine. We compared the two systems on two sets of benchmark queries, the first from the Berlin SPARQL Benchmark (BSBM) (Bizer and Schultz, 2009) and the second from the *SP2B* SPARQL Performance Benchmark (Schmidt et al., 2010). We choose these query sets because they present a variety of features (e.g., different query patterns, operators, size of intermediate and final results, and time complexity) which stress different components of the processing engines. For all the experiments, we checked that both Urika and GEMS provided correct answers to the queries. Since the SPARQL-to-C++ compiler presently does not yet provide support for all SPARQL 1.1 features, we hand coded the BSBM queries in a similar fashion as we would expect from a mature version of the compiler. For this reason, we discuss experiments for the two benchmarks separately. In both cases, we compiled the queries code using g++ version 4.6.2 with the -O3 optimization flag; GMT used OpenMPI 1.6.3.

5.1 BSBM: Hand-Coded Queries

The BSBM benchmark is built around an e-commerce use case in which a set of products is offered by different vendors and consumers have posted reviews about products. We evaluate the systems on queries 1–6 from the *business intelligence* use case, which simulates different stakeholders asking analytical questions against the dataset. The benchmark allows to specify values for some of the triple patterns variables. We set these parameters in order to force the queries to find legal matches on the data. Tables 1 and 2 report execution time for running the queries against 100 million and 1 billion triples respectively, on both GEMS and Urika. We don't report the result sizes since most of the queries feature LIMIT operators. For GEMS, we varied the number of allocated nodes on the cluster according to the data size. Reported execution latencies do not include the compilation time, which is of 5.2 s on average. For low-latency queries, this cost is often higher than the actual execution time. However, we assume that compilation time will ultimately be amortized to an almost negligible value for high-latency queries and queries that will be used frequently, since they need to be compiled once.

When comparing execution latencies only, in all the experiments GEMS perform significantly better than Urika. Nevertheless, taking into account also the cost of compilation, the two systems offer similar performance. Aside from query time, there are other factors that can be taken into account when comparing

TABLE 1 Time (in Seconds) to Execute BSBM *Business Intelligence* Queries 1–6 Against 100M Triples

	GEMS		
Query	8 Nodes	16 Nodes	Urika
Q1	3.35	2.65	4.245
Q2	1.71	1.70	6.754
Q3	3.44	2.26	5.266
Q4	1.35	1.69	4.571
Q5	1.02	1.30	7.483
Q6	0.238	0.316	5.649

TABLE 2 Time (in Seconds) to Execute BSBM *Business Intelligence* Queries 1–6 Against 1B Triples

	GEMS		
Query	16 Nodes	32 Nodes	Urika
Q1	9.89	6.41	7.775
Q2	2.99	3.01	25.62
Q3	12.8	7.21	11.133
Q4	2.54	2.50	7.717
Q5	1.87	1.99	10.048
Q6	0.358	0.482	7.091

with Urika. Commodity clusters are likely to be more affordable (in terms of purchase and maintenance) and more available than specialized hardware. Also, to our knowledge, the Urika appliance can run only one query at a time. On the contrary, GEMS allows running multiple queries in parallel, potentially achieving much higher query throughput. We evaluated also the scalability of the proposed approach, varying the number of the allocated nodes. While in some cases we report significant performance improvement, on average increasing the number of nodes does not result in substantial speed-ups. On the contrary, for low latency queries, such as Q6, we report a slight performance degradation, mainly due to the higher relative cost of communication.

5.2 SP2B Queries

We automatically generated the C++ implementation of SP2B queries 1–10 through GEMS' SPARQL-to-C++ compiler. SP2B queries provide a complex environment on which evaluating performance of RDF engines: they describe different kinds of graph patterns, including *long path chains*, *bushy patterns* (single nodes with many successors), and combinations of the above; they present different selectivity, dramatically varying memory requirements and time complexity, especially when increasing the dataset size.

Tables 3 and 4 show result counts and execution time for running the queries against 100 million and 1 billion triples, respectively. On average, the C++ code generation process required 0.12 s, while the compilation of the generated implementations required 3.86 s. While for BSBM queries, increasing the number of nodes in the system did not lead to significant performance improvement, in this case we observe better scalability. For complex queries, characterized by

TABLE 3 Number of Result Tuples and Time (in Seconds) to Execute SP2B Queries 1–10 Against 100M Triples

		GEMS			
Query	**#Answers**	**8 Nodes**	**16 Nodes**	**32 Nodes**	**Urika**
Q1	1	0.041053	0.051353	0.127594	1.244
Q2	9050604	11.103515	7.191075	5.063025	9.909
Q3a	1466402	0.159287	0.108663	0.102876	2.348
Q3b	10143	0.141681	0.094862	0.099148	1.193
Q3c	0	0.139884	0.089665	0.100040	0.96
Q4	460135248	244.082552	149.840050	64.886178	62.575
Q5a Q5a+	– 2016996	–[a] 102.614320	–[a] 60.674561	–[a] 34.977476	–[b]
Q5b	2016996	3.492491	1.993362	1.311534	2.437
Q6	9812030	24.896199	14.594010	10.430950	13.142
Q7	14645	13.882659	10.363371	8.919411	4.216
Q8 Q8+	493 493	1177.359682 0.149563	747.210641 0.216421	473.620011 0.246394	67.621
Q9	4	12.250798	6.472046	3.496389	2.214
Q10	656	0.019665	0.033179	0.052079	0.718

[a] Timeout Reached
[b] Memory Exceeded

TABLE 4 Number of Result Tuples and Time (in Seconds) to Execute SP2B Queries 1–10 Against 1B Triples

		GEMS		
Query	**#Answers**	**32 Nodes**	**64 Nodes**	**Urika**
Q1	1	0.052522	0.088859	1.105
Q2	95481441	31.522307	18.305085	64.221
Q3a	10157400	0.308892	0.219484	7.165
Q3b	70613	0.291832	0.210325	1.594
Q3c	0	0.283186	0.200547	0.938
Q4	4509421916	671.622605	348.719	673.937
Q5a Q5a+	– 9197350	–[a] 341.864665	–[a] 182.436236	–[b]
Q5b	9197350	5.692948	3.746873	5.492
Q6	120405337	75.462789	57.318893	115.66
Q7	20441	286.427132	250.132178	12.841
Q8 Q8+	493 493	5604.322 0.229329	4510.38 0.302651	814.554
Q9	4	35.869012	18.887735	9.614
Q10	656	0.048982	0.086379	0.942

[a] Timeout Reached
[b] Memory Exceeded

higher latencies, we report a quasi-linear speed-up when increasing the number of nodes. This demonstrates the effectiveness of approach, since execution of several SP2B queries (e.g., Q2 and Q4) is dominated by the pattern-matching component, modeled as graph exploration and well supported by the custom runtime layer. Again, for simple queries, GEMS did not provide significant benefits when increasing the scale of the system; in three cases (Q1, Q8, and Q10) we reported a slight performance degradation. When compared against Urika, GEMS provides valuable speed-ups for most of the queries. In one case, query Q7, Urika outperforms GEMS when scaling the data size to 1 Billion triples. Query 7 mainly tests nested *closed world negation*, and offers several opportunities for optimization, related in particular to the reuse of graph pattern results (Schmidt et al., 2010). We do not currently exploit such type of optimizations. For query Q5a, the Urika appliance quickly runs out of memory for both the datasets. Similarly, GEMS is not able to complete the query as automatically implemented by the code generator, within a time threshold of

3 h. Q5a features a simple BGP constrained by a FILTER. What makes the query challenging is that it is composed of two disjoint patterns, resulting in a very high number of partial matches. In addition, the FILTER can be evaluated only at the last levels of the graph walk, thus it does not allow effective pruning of the solution space. We manually modified Q5a, splitting the original graph walk in two distinct ones, associated with the disjoint patterns, and evaluating the FILTER condition afterwards. The modified version of Q5a completed execution, for all the datasets, in reasonable time. We are currently improving the compiler for automatically applying these optimizations. Nevertheless, this experiment demonstrates the value of allowing the user tuning or modifying the generated code. We exploited the possibility of tuning the generated code for query Q8 also. Q8 presents the UNION of two partially overlapping patterns, characterized by triples with different selectivity. We modified the generated code, merging the matching of the two patterns in a single Graph Walk, and starting the Graph Walk by matching the most selective triple pattern. Exploiting the selectivity has a tremendous impact on performance: the tuned query runs thousands of times faster when compared to Urika, and even more when compared to GEMS' initial implementation. Currently, the code generator is not able to estimate the selectivity of a single triple pattern. We are improving the optimization process by considering statistics collected through data probing. These may include, for example, the number of vertices in the graph with a specific label. Exploiting this information will result in significant performance improvement for all the queries characterized by triple patterns having highly different selectivity, such as Q8.

6 CONCLUSIONS

This chapter discussed our approach for querying RDF triple stores, which relies on the conversion of SPARQL queries to graph homomorphism routines. Differently from other databases, these routines substantially are parallel C++ graph-matching algorithms generated by a SPARQL-to-C++ translator. The routines execute through a library (SGLib), which provides methods to access and manage the graph and the dictionary (i.e., the graph database) generated from the ingestion of RDF triples. Underneath the database, a runtime library (GMT) provides a Partitioned Global Address Space (PGAS), lightweight software multithreading and network messages aggregation to enable more efficient in-memory execution of the graph search algorithms on commodity high-performance clusters, which usually present performance issues with these types of workload. The runtime enables scaling the size of the database, while maintaining near constant query throughput, when adding new cluster nodes. We call the full software stack (SPARQL-to-C++ translator, SGLib, and GMT) GEMS, Graph database Engine for Multithreaded Systems. This chapter briefly presented the software stack, and detailed how the translation from SPARQL to graph homomorphism routines is performed. We highlighted differences

between completely automatically generated code and code that has been hand optimized, and compared our system to a custom appliances for data analytics such as the Cray Urika. We believe that GEMS is a significant step towards providing a more efficient way to address the challenges brought by the "big data" revolution, enabling more cost-effective analysis of the exponentially growing amount of data in many application areas.

REFERENCES

Apache Jena, 2014. Apache Jena–Home. URL https://jena.apache.org/ (last accessed 19 March 2014).

Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A., 2010. Matrix Bit loaded: a scalable lightweight join query processor for RDF data. In: Proceedings of the 19th International Conference on World Wide Web, pp. 41–50.

BigDataDB, 2014. bigdata®. URL http://www.systap.com/bigdata.htm (last accessed 17 April 2014).

Bizer, C., Schultz, A., 2009. The Berlin SPARQL benchmark. Int. J. Semantic Web Inf. Syst. 5 (2), 1–24.

Harris, S., Lamb, N., Shadbolt, N., 2009. 4store: the design and implementation of a clustered RDF store. In: 5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009), pp. 94–109.

Harth, A., Umbrich, J., Hogan, A., Decker, S., 2007. YARS2: a federated repository for querying graph structured data from the web. In: ISWC'07/ASWC'07: 6th International Semantic Web and 2nd Asian Semantic Web Conference, pp. 211–224.

Klyne, G., Carroll, J.J., McBride, B., 2004. Resource description framework (RDF): concepts and abstract syntax. W3C Recommendation. W3C, Cambridge, MA. URL http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/.

Kotoulas, S., Urbani, J., Boncz, P., Mika, P., 2012. Robust runtime optimization and skew-resistant execution of analytical SPARQL queries on pig. In: Proceedings of the 11th International Semantic Web Conference.

Morari, A., Tumeo, A., Chavarria-Miranda, D., Villa, O., Valero, M., 2014. Scaling irregular applications through data aggregation and software multithreading. In: IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 1126–1135. doi:10.1109/IPDPS.2014.117.

Neumann, T., Weikum, G., 2010. The RDF-3X engine for scalable management of RDF data. VLDB J. 19 (1), 91–113.

Ontotext, 2014. Owlim–ontotext. URL http://www.ontotext.com/owlim (last accessed 17 April 2014).

Rohloff, K., Schantz, R.E., 2010. High-performance, massively scalable, distributed systems using the MapReduce software framework: the SHARD triple-store. In: PSI EtA '10: Programming Support Innovations for Emerging Distributed Applications, pp. 4:1–4:5. URL http://doi.acm.org/10.1145/1940747.1940751.

Schmidt, M., Hornung, T., Meier, M., Pinkel, C., Lausen, G., 2010. SP2Bench: a SPARQL performance benchmark. In: de Virgilio, R., Giunchiglia, F., Tanca, L. (Eds.), Semantic Web Information Management. Springer, Berlin, pp. 371–393. URL http://dx.doi.org/10.1007/978-3-642-04329-1_16.

SesameRDF, 2014. openRDF.org: Home. URL http://www.openrdf.org/ (last accessed 17 April 2014).

Shao, B., Wang, H., Li, Y., 2013. Trinity: a distributed graph engine on a memory cloud. In: Proceedings of the 2013 International Conference on Management of Data, pp. 505–516.

Ullmann, J.R., 1976. An algorithm for subgraph isomorphism. J. ACM 23 (1), 31–42. ISSN 0004-5411.

Virtuoso, 2014. OpenLink Virtuoso Universal Server. URL http://virtuoso.openlinksw.com/ (last accessed 17 April 2014).

W3C SPARQL Working Group, 2013. SPARQL 1.1 overview. W3C Recommendation. W3C, Cambridge, MA. URL http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/ .

Weiss, C., Karras, P., Bernstein, A., 2008. Hexastore: sextuple indexing for semantic web data management. Proc. VLDB Endowm. 1 (1), 1008–1019.

YarcData, 2014. YarcData, Inc. Urika Big Data Graph Appliance. URL http://www.cray.com/Products/BigData/uRiKA.aspx.

Zeng, K., Yang, J., Wang, H., Shao, B., Wang, Z., 2013. A distributed graph engine for web scale RDF data. Proc. VLDB Endowm. 6 (4), 265–276 (VLDB Endowment).

Index

Note: Page numbers followed by "*f* " indicate figures, "*t*" indicate tables.

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z