

Francesca De Luzi
Flavia Monti
Massimo Mecella *Editors*

Engineering Information Systems with Large Language Models



Springer

Engineering Information Systems with Large Language Models

Francesca De Luzi • Flavia Monti •
Massimo Mecella
Editors

Engineering Information Systems with Large Language Models

 Springer

Editors

Francesca De Luzi 
Sapienza Università di Roma
Rome, Italy

Flavia Monti 
Sapienza Università di Roma
Rome, Italy

Massimo Mecella 
Sapienza Università di Roma
Rome, Italy

ISBN 978-3-031-92284-8

ISBN 978-3-031-92285-5 (eBook)

<https://doi.org/10.1007/978-3-031-92285-5>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

The rapid advancement of artificial intelligence (AI), and more specifically generative AI and large language models (LLMs), is transforming various fields, and among them, there is growing interest in its potential application in information systems engineering (ISE). These technologies offer unprecedented opportunities to support the design, development, and reengineering of information systems (ISs). Despite their potential, a systematic and structured approach to integrating LLMs in IS engineering is still in its infancy. Recent studies have explored various applications, such as conceptual design, software development, business process extraction, and IS dimensioning using ChatGPT and similar models. However, these efforts remain fragmented, lacking a comprehensive and unified framework. In this book, we aim at providing a first attempt toward such a framework, fully aware that it might be a challenging and ephemeral effort, in a landscape in which a new paper about LLMs is published on <https://arxiv.org/>¹ every 8 minutes.

In March 2024, we started developing the tutorial *Engineering Information Systems with LLMs and AI-based Techniques*, which was accepted for presentation and presented at CAiSE 2024, 36th International Conference on Advanced Information Systems Engineering, in June 2024. The tutorial aimed to provide attendees with a foundational understanding of AI, generative AI, and LLMs from an IS engineering perspective, followed by an in-depth analysis of emerging methodologies and applications in IS design and development. The positive reception of the tutorial, combined with the growing academic and industrial interest in this field, led to the opportunity to expand its content into this instant book.

The book is a collaborative effort of the research group in *Processes, Services and Software Engineering* at Sapienza Università di Roma, Italy. Each chapter is written by PhD students and postdocs actively working on applying LLMs to IS

¹ arXiv is a free distribution service and an open-access archive, hosted by Cornell University, for over 2 million scholarly articles in various fields, including computer science. Materials are not peer reviewed by arXiv. Nowadays, researchers in computer science, AI, GenAI, and LLMs, in order to have a timely dissemination of their results before a peer-reviewed publication, are largely adopting this service.

engineering, with a deep and practical experience in LLMs and their adoption for building applications and systems. A careful revision of the chapters, and preliminarily the definition of contents of the single chapters, has been carried out by us editors in order to provide the unifying framework we are aiming at.

Adopting an ISE perspective means we consider LLMs, as many other technologies, as black boxes and investigate how they can be adopted in a principled way in order to develop more complex information systems that can take advantage of LLMs. In contrast to other technologies, LLMs pose new challenges to IS engineers, as their behavior is in most cases scarcely explainable, scarcely repeatable, and forecastable. These challenges are not only relevant for IS engineers but also for researchers and practitioners in software engineering, BPM (Business Process Management), and software architectures (including service-oriented ones) and, more broadly, those working with digital technologies. These communities are the target of our book.

As previously stated, we aimed at creating an instant book, i.e., a book that has been produced and published very quickly to meet a specific demand. We took utmost care to provide accurate and precise information, as well as a survey of the relevant literature. All references are up to date as till the end of 2024. Nevertheless, we hope that the book will not fade out as quickly as instant books generally do, as we strongly believe that the concepts here presented are more durable than the single technologies we are describing. Only the future, which luckily is not yet foreseeable by an LLM, will be able to witness whether our aim will be achieved.

Rome, Italy
March 2025

Francesca De Luzi
Flavia Monti
Massimo Mecella

Acknowledgements We would like to acknowledge the following for financial support to our research:

- MICS (Made in Italy–Circular and Sustainable) Extended Partnership funded by the EU—NextGenerationEU PNRR MUR
- SPACE IT UP Extended Partnership funded from ASI (Agenzia Spaziale Italiana) and MUR
- Thales Alenia Space and Regione Lazio, through the fellowships 35752-22066DP000000040-A0627S0030 Utilizzo dell’Intelligenza Artificiale a supporto della progettazione e qualifica del prodotto spaziale and 35757-22066DP000000041-A0627S0031 Advanced Software Based on Cloud Computing and Machine Learning for Space Systems
- Agreement on “Agile&Secure Digital Twins”
- Italian National PhD Program in Artificial Intelligence
- PhD fellowship Pubblica Amministrazione DM118 (CUP83C22003460006)
- Horizon Europe projects GenDAI and UPTAKERS

Contents

| | | |
|-----------|---|------------|
| 1 | Introduction | 1 |
| | Massimo Mecella | |
| 2 | History of Large Language Models | 13 |
| | Francesca De Luzi | |
| 3 | Large Language Models for Dummies | 33 |
| | Marco Calamo and Matteo Marinacci | |
| 4 | A Reference Framework for Information Systems | 61 |
| | Francesca De Luzi and Massimo Mecella | |
| 5 | Exploring Large Language Models in Information Systems: A Survey | 77 |
| | Filippo Bianchini and Matteo Marinacci | |
| 6 | Programming Large Language Models | 111 |
| | Marco Calamo and Jacopo Rossi | |
| 7 | Retrieval-Augmented Generation | 139 |
| | Filippo Bianchini | |
| 8 | Large Language Model Agents | 173 |
| | Jerin George Mathew and Jacopo Rossi | |
| 9 | Applications of Large Language Models in Information Systems | 207 |
| | Jerin George Mathew and Flavia Monti | |
| 10 | Ethics and Governance of Large Language Models | 233 |
| | Gabriele Rendace Presta and Andrea Paolella | |
| 11 | Epilogue | 285 |
| | Massimo Mecella | |

List of Contributors

Filippo Bianchini

Sapienza Università di Roma, Rome, Italy

e-mail: bianchini@diag.uniroma1.it

Filippo Bianchini is a PhD student in Engineering in Computer Science. With a MSc in Management Engineering, his research explores advanced applications of large language models (LLMs), retrieval-augmented generation (RAG), and knowledge graphs (KG) within information systems.

Marco Calamo

Sapienza Università di Roma, Rome, Italy

e-mail: calamo@diag.uniroma1.it

Marco Calamo is a PhD student in Engineering in Computer Science. His research interests are Industry 4.0 and large language models.

Francesca De Luzi

Sapienza Università di Roma, Rome, Italy

e-mail: deluzi@diag.uniroma1.it

Francesca De Luzi is a postdoctoral researcher. She earned a BSc in Management Engineering at the University of Rome Tor Vergata and completed her MSc in Management Engineering at Sapienza. In 2024, she obtained her PhD in Industrial and Management Engineering, defending a thesis that highlighted the role of digital transformation in a hyper-connected world through the analysis of various application domains. Her recent research has focused on the impact of AI techniques in organizations.

Matteo Marinacci

Sapienza Università di Roma, Rome, Italy

e-mail: marinacci@diag.uniroma1.it

Matteo Marinacci is a PhD student. He earned an MSc in Engineering in Computer Science. His research focuses on Industry 4.0, with a particular emphasis on human-in-the-loop systems, optimization of real-world manufacturing processes, and error detection and recognition in production environments. Currently, he is exploring how extended reality technologies can assist operators during the assembly, integration, and testing of complex products, such as satellites, aiming to improve precision and efficiency in the space industry.

Jerin George Mathew

Sapienza Università di Roma, Rome, Italy

e-mail: mathew@diag.uniroma1.it

Jerin George Mathew is a postdoctoral researcher. He earned a PhD in Artificial Intelligence in 2025. His research focuses on applying AI techniques to data cleaning and data management, ranging from entity resolution to fairness.

Flavia Monti

Sapienza Università di Roma, Rome, Italy

e-mail: monti@diag.uniroma1.it

Flavia Monti is a postdoctoral researcher. She earned a PhD in Engineering in Computer Science in 2025. Her research interests focus on Industry 4.0 and smart manufacturing, particularly the integration of computer vision, machine learning, and artificial intelligence to improve production quality, reduce costs, increase machinery uptime, and achieve zero-defect manufacturing.

Andrea Paoletta

Sapienza Università di Roma, Rome, Italy

e-mail: and.paoletta@gmail.com

Andrea Paoletta is a high school teacher of History and Philosophy. He studied History of Philosophy at Sapienza Università di Roma, where he obtained a PhD in Documentation Studies, Linguistics, and Literature in 2017.

Gabriele Rendace Presta

Sapienza Università di Roma, Rome, Italy

e-mail: gabrielepresta.doc@gmail.com

Gabriele Rendace Presta is a researcher in digital humanities and teacher of History and Classics. His main areas of interest include ethical and aesthetic issues related to digital technologies. He is currently exploring the potential philosophical impact of these technologies on policy decision-makers.

Jacopo Rossi

Sapienza Università di Roma, Rome, Italy

e-mail: j.rossi@diag.uniroma1.it

Jacopo Rossi is a PhD student in Engineering in Computer Science. His research spans software engineering, information systems, and process mining techniques in constrained environments (such as satellites).

Massimo Mecella

Sapienza Università di Roma, Rome, Italy

e-mail: mecella@diag.uniroma1.it

Massimo Mecella is a full professor. He conducts research in the fields of information systems engineering, service-oriented computing, mobile and pervasive computing, process management, data and process mining, big data analytics, and human-computer interaction, focusing on smart applications, environments, and communities. He was the General Chair of CAiSE 2019, BPM 2021, and ICSOC 2023. In 2025 he became program co-chair of IEEE CAI 2025-IEEE Conference on Artificial Intelligence. He is currently a member of the Steering Committees of the conference series CAiSE, ICSOC, and SummerSOC.

Chapter 1

Introduction



Massimo Mecella 

In November 2022, OpenAI launched ChatGPT, a text-generating chatbot based on a large language model (LLM). This has been the start of a storm.¹ OpenAI, an originally non-profit research organization started in 2015, later embracing a “capped” for-profit model, had worked on reinforcement learning and Generative Pre-trained Transformer (GPT) models, in order to develop “safe and beneficial artificial general intelligence.”²

Since then, the landscape of information systems engineering has been changing. AI technologies, and specifically LLMs, offer unprecedented opportunities to support the design, development, and optimization of information systems. Recent studies have explored various applications, such as conceptual design, software development, business process extraction, and IS dimensioning using ChatGPT and similar models. However, these efforts remain fragmented, lacking a comprehensive and unified framework.

The primary objective of this book is to bridge the existing gaps by offering a principled and structured perspective on how LLMs can be systematically leveraged for IS engineering. We present a unifying pipeline, integrating state-of-the-art research findings with case studies drawn from our research activities. Additionally,

¹ Cf. <https://techcrunch.com/2025/02/12/chatgpt-everything-to-know-about-the-ai-chatbot/> (link checked February 2025).

² A story of OpenAI can be read on <https://en.wikipedia.org/wiki/OpenAI> (link checked February 2025).

M. Mecella (✉)
Sapienza Università di Roma, Rome, Italy
e-mail: mecella@diag.uniroma1.it

we provide a survey of the latest literature and practical implementations, offering readers a solid foundation to explore and apply these techniques in their own work.

To facilitate practical application, we also provide access to a repository of code and examples on GitHub,³ allowing readers to experiment with and extend the methodologies and techniques presented in this book.

Structured as a comprehensive guide, this book is intended for researchers, practitioners, and students in the fields of information systems, software engineering and architectures, and business process management. By exploring both theoretical foundations and practical applications, we aim to equip readers with the knowledge and tools necessary to harness the power of LLMs in designing, developing, and optimizing modern information systems. We hope this book serves as a valuable resource for those looking to navigate and contribute to this rapidly evolving domain.

Interestingly, we can take ChatGPT and adopt the following prompt.

Explanation Prompt

I need to write an introduction, in English, for a book that originated
 ↳ from a tutorial, Engineering Information Systems with LLMs and
 ↳ AI-based Techniques, presented at the CAiSE 2024 conference. I
 ↳ would like also to have the table of contents of this book.
 Here are some details:

Massimo Mecella, PhD in Engineering in Computer Science, is a full
 ↳ professor at Sapienza, where he is conducting research in the
 ↳ fields of information systems engineering, software architectures,
 ↳ distributed middleware and service oriented computing, mobile and
 ↳ pervasive computing, process management, data and process mining,
 ↳ big data analytics, advanced interfaces, and human-computer
 ↳ interaction, focusing on smart applications, environments, and
 ↳ communities.

We achieve the following introduction and table of contents (ToC)

Introduction

The rapid advancements in Artificial Intelligence (AI) -- and
 ↳ particularly the rise of Generative AI through Large Language
 ↳ Models (LLMs) -- are transforming the landscape of information
 ↳ systems engineering. LLMs are increasingly being adopted as
 ↳ powerful tools to support the design, development, and dimensioning
 ↳ of complex information systems (IS). This book originates from the
 ↳ tutorial Engineering Information Systems with LLMs and AI-based
 ↳ Techniques, presented at the CAiSE 2024 conference, which explored
 ↳ how AI and LLMs can be systematically integrated into the
 ↳ engineering process of information systems.

³ https://github.com/DIAG-Sapienza-BPM-Smart-Spaces/EngIS_LLM_AI.

The tutorial was delivered by Massimo Mecella, a prominent figure in the field of information systems engineering. Prof. Mecella holds a PhD in Engineering in Computer Science and is a full professor at Sapienza University of Rome. His research spans a wide range of areas, including software architectures, distributed middleware and service-oriented computing, mobile and pervasive computing, process management, data and process mining, big data analytics, advanced interfaces, and human-computer interaction -- with a particular focus on smart applications, environments, and communities. He is the author of approximately 250 papers, with an h-index of 42, and has been involved in numerous European and Italian research projects, including serving as the technical manager of the WORKPAD and SM4All projects. He has also played a key role in organizing major scientific events, including serving as the General Chair of CAiSE 2019, BPM 2021, and ICSOC 2023.

At present, the adoption of Generative AI for IS engineering remains fragmented. While there have been recent studies exploring how LLMs can assist in software development, system evaluation, and business process extraction, a unified framework is still lacking. The goal of this book is to provide a structured and comprehensive approach to integrating LLMs and AI-based techniques into the entire lifecycle of information systems -- from conceptual design to deployment and optimization. Drawing on both theoretical foundations and practical case studies, this book aims to equip researchers, practitioners, and students with a clear understanding of how to leverage AI to improve the efficiency, scalability, and adaptability of modern information systems. This book consolidates state-of-the-art research and practical insights to provide a structured and actionable framework for integrating AI and LLMs into information systems engineering. By bridging the gap between theory and practice, it offers a valuable resource for researchers, developers, and decision-makers looking to harness the power of AI for building more intelligent and adaptable information systems.

Table of Contents

1. Introduction

Background and Motivation

Scope and Objectives of the Book

2. Foundations of AI and LLMs for Information Systems

Overview of AI Techniques

Evolution and Capabilities of Large Language Models

Challenges and Opportunities in IS Engineering

3. LLMs for Information Systems Design

Conceptual Design of Information Systems with LLMs

Requirement Elicitation and Analysis

Design Automation and Optimization with AI

| |
|---|
| 4. LLMs for Software Development and Implementation |
| Code Generation and Completion |
| Testing and Debugging with LLMs |
| AI-Assisted Refactoring and Performance Tuning |
| 5. Business Process Modeling and Analysis with LLMs |
| Extraction of Process Models from Documents |
| Process Optimization and Simulation with AI |
| Automated Business Rule Generation |
| 6. Data and Process Mining with AI |
| Pattern Recognition and Anomaly Detection |
| Predictive Analytics and Decision Support |
| Mining for Process Improvement and Automation |
| 7. Case Studies and Practical Applications |
| Real-World Examples from Research and Industry |
| Lessons Learned and Best Practices |
| 8. Ethical and Practical Considerations |
| Bias and Fairness in AI-Based Systems |
| Transparency and Interpretability |
| Human-in-the-Loop and AI Governance |
| 9. Future Directions and Open Challenges |
| Emerging Trends in Generative AI and IS Engineering |
| Research Gaps and Opportunities |
| Roadmap for Future Developments |
| 10. Conclusions |
| Key Takeaways |
| Final Reflections and Next Steps |

On Gemini (another chatbot based on LLMs, by Google), we achieve this answer:

Introduction

The dawn of Large Language Models (LLMs) has ushered in a paradigm shift across numerous domains, and Information Systems (IS) engineering stands poised to benefit immensely from this transformative technology. This book, stemming from the tutorial Engineering Information Systems with LLMs and AI-based Techniques presented at the CAiSE 2024 conference, aims to provide a comprehensive and practical guide to leveraging the power of LLMs in the design, development, and dimensioning of modern information systems.

In an era defined by rapid technological evolution, the integration of

- ↪ AI, particularly LLMs, into the IS engineering lifecycle is no
- ↪ longer a futuristic concept but a present-day necessity. This book
- ↪ addresses the critical need for a structured and principled
- ↪ approach to this integration, filling a gap in the existing
- ↪ literature. While recent studies have explored the potential of
- ↪ LLMs like ChatGPT for specific IS tasks, a holistic framework that
- ↪ unifies these efforts has been conspicuously absent.

This work is grounded in the extensive research and practical

- ↪ experience of Massimo Mecella, a full professor at Sapienza
- ↪ University of Rome. Professor Mecella's expertise spans a broad
- ↪ spectrum of IS engineering, including software architectures,
- ↪ process management, data analytics, and human-computer interaction,
- ↪ with a focus on smart applications and environments. His deep
- ↪ understanding of these domains, coupled with his commitment to
- ↪ advancing the field, makes him uniquely qualified to guide readers
- ↪ through the intricacies of LLM-driven IS engineering.

This book builds upon the foundational knowledge shared during the

- ↪ CAiSE 2024 tutorial, offering a structured exploration of AI and
- ↪ Generative AI, with a specific focus on LLMs, through the lens of
- ↪ IS engineering. It presents recent approaches and applications for
- ↪ utilizing these technologies in IS design and development,
- ↪ complemented by real-world case studies from the author's research.
- ↪ Furthermore, it provides a systematic analysis of existing
- ↪ literature and practical applications, culminating in a cohesive
- ↪ pipeline for integrating LLMs into the IS engineering process.

This book is intended to serve as a valuable resource for researchers,

- ↪ practitioners, and students seeking to understand and harness the
- ↪ transformative potential of LLMs in the engineering of information
- ↪ systems. It provides a roadmap for navigating the evolving
- ↪ landscape of AI-driven IS development, empowering readers to
- ↪ leverage these powerful tools effectively.

Table of Contents

Chapter 1: Foundations of AI and Generative AI in Information Systems

- 1.1 Introduction to Artificial Intelligence and Machine Learning
- 1.2 The Rise of Generative AI and Large Language Models
- 1.3 LLMs in the Context of Information Systems Engineering
- 1.4 Key Concepts: Tokenization, Embeddings, and Transformer
 - ↪ Architectures

Chapter 2: LLMs for Conceptual Design of Information Systems

- 2.1 Requirements Elicitation with LLMs
- 2.2 Use Case Modeling and Scenario Generation
- 2.3 Domain Modeling and Ontology Development
- 2.4 Case Study: Utilizing LLMs for Conceptual Design in a Smart City
 - ↪ Project

Chapter 3: LLMs for Software Development and Code Generation

- 3.1 Code Generation and Completion using LLMs
- 3.2 Automated Testing and Debugging with LLMs
- 3.3 Software Architecture Design and Documentation
- 3.4 LLMs for API Design and Development

Chapter 4: LLMs for Dimensioning and Performance Evaluation of ↪ Information Systems

- 4.1 Performance Analysis and Prediction using LLMs
- 4.2 Resource Allocation and Optimization
- 4.3 Simulation and Modeling with LLMs
- 4.4 Case Study: Dimensioning an E-commerce Platform with LLM-based
↪ Analysis

Chapter 5: Extracting Business Process Specifications from Documents ↪ with LLMs

- 5.1 Natural Language Processing for Process Discovery
- 5.2 Automatic Extraction of Process Models
- 5.3 Validation and Refinement of Extracted Processes
- 5.4 Application in Business Process Management

Chapter 6: A Unifying Framework for LLM-Driven IS Engineering

- 6.1 Integrating LLMs into the IS Engineering Lifecycle
- 6.2 Best Practices and Guidelines for LLM Usage
- 6.3 Addressing Challenges and Limitations
- 6.4 Future Directions and Research Opportunities

Chapter 7: Case Studies and Practical Applications

- 7.1 Detailed Examples from Research and Industry
- 7.2 Real-World Scenarios and Implementation Strategies
- 7.3 Lessons Learned and Practical Insights

Chapter 8: Conclusion and Future Outlook

- 8.1 Summarizing Key Findings and Contributions
- 8.2 The Future of AI and LLMs in Information Systems
- 8.3 Ethical Considerations and Responsible AI

References

And finally, DeepSeek provides:

Introduction

In the rapidly evolving landscape of information systems engineering, the integration of Large Language Models (LLMs) and AI-based techniques has emerged as a transformative force. These technologies are not only reshaping how we design, develop, and manage information systems but are also opening new frontiers in automation, decision-making, and user interaction. This book, *Engineering Information Systems with LLMs and AI-based Techniques*, is born out of a tutorial presented at the CAiSE 2024 conference, where leading experts and practitioners gathered to explore the cutting-edge advancements in this field.

Authored by Massimo Mecella, a distinguished full professor at Sapienza University, this book distills years of research and practical experience into a comprehensive guide. Professor Mecella's expertise spans a wide array of domains, including information systems engineering, software architectures, distributed middleware, service-oriented computing, mobile and pervasive computing, process management, data and process mining, big data analytics, and advanced human-computer interaction. His work has consistently focused on the development of smart applications, environments, and communities, making him a pivotal figure in the intersection of AI and information systems.

This book is designed to serve as both a foundational text and a practical resource for researchers, practitioners, and students alike. It provides a thorough exploration of the theoretical underpinnings of LLMs and AI-based techniques, while also offering actionable insights into their application in real-world information systems. Whether you are a seasoned professional looking to stay ahead of the curve or a newcomer eager to understand the potential of these technologies, this book will equip you with the knowledge and tools needed to navigate the complexities of modern information systems engineering.

Table of Contents

1. Introduction to Information Systems Engineering
 - Evolution of Information Systems
 - The Role of AI in Modern Information Systems
 - Overview of Large Language Models (LLMs)
2. Foundations of Large Language Models
 - Understanding LLMs: Architecture and Functionality
 - Training and Fine-Tuning LLMs
 - Ethical Considerations and Challenges
3. AI-Based Techniques in Information Systems
 - Machine Learning and Deep Learning Fundamentals
 - Natural Language Processing (NLP) Techniques
 - Integration of AI with Traditional Information Systems

4. Designing Information Systems with LLMs
 - System Architecture and Design Principles
 - Data Management and Processing
 - User Interaction and Experience Design
5. Applications of LLMs in Information Systems
 - Automated Decision-Making and Support Systems
 - Process Automation and Optimization
 - Intelligent Data Analytics and Insights
6. Case Studies and Real-World Applications
 - Smart Environments and Communities
 - Industry-Specific Applications (Healthcare, Finance, etc.)
 - Lessons Learned and Best Practices
7. Future Trends and Directions
 - Emerging Technologies and Their Impact
 - The Future of AI in Information Systems
 - Research Opportunities and Open Challenges
8. Conclusion
 - Summary of Key Insights
 - The Path Forward for Information Systems Engineering
9. Appendices
 - Glossary of Terms
 - Additional Resources and References
 - Tools and Frameworks for LLM and AI Integration

The reader is invited to continue reading the book in order to discover the final table of contents, which has been defined beforehand trying such prompts. We anticipate that, in the end, most of the topics suggested by the three chatbots are somehow addressed, even if with different levels of details; specifically, the reader will notice that our case studies are deeply different from those considered by the chatbots. We will discuss these aspects in the Epilogue. The reader can be assured that in this book, we will address all the basic notions of the examples above, as prompt, text generation, etc. with the focus of an IS engineer, who wants to adopt such technologies in order to be assisted in the conceptualization, design, and development of an information system.

Before concluding this introduction, by presenting the organization of the book, we would like to outline the value chain of generative AI, including LLMs, as shown in Fig. 1.1, as well as their location shown in Fig. 1.2.

Figure 1.1 shows the main entities involved in the GenAI ecosystem, categorized into regulatory bodies, computing/infrastructure providers (such as NVIDIA, Intel, AMD), model developers (like OpenAI, Meta, and Anthropic), code libraries (e.g., TensorFlow, PyTorch), applications (such as ChatGPT, DALL-E 2, Bard), and services (like Uber and Morgan Stanley). The diagram also highlights the flow of transactions and information across these different layers. Information systems engineers (practitioners and researchers) should consider the applications

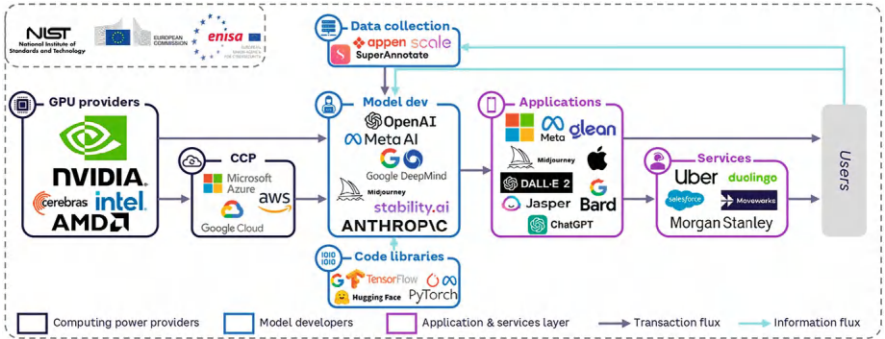


Fig. 1.1 Players across the GenAI value chain⁴

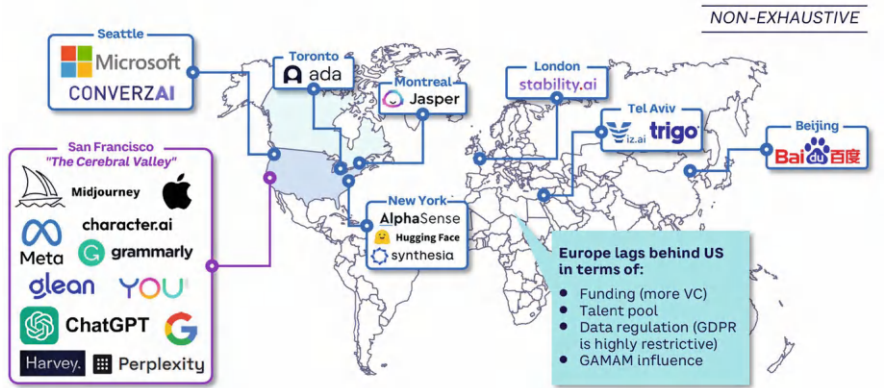


Fig. 1.2 Location of major players across GenAI value chain⁵

and services layer, by investigating either new tools leveraging models for building new applications (cf. Epilogue) or directly new type of applications and services.

Figure 1.2 provides a geographical distribution of major players in the GenAI value chain, indicating key hubs like San Francisco (home to OpenAI, Midjourney, and others), also called “The Cerebral Valley,” Seattle (Microsoft), New York (Hugging Face, Synthesia), London (Stability AI), and Beijing (Baidu). A note mentions that Europe lags behind the USA in funding, talent pool, and regulatory flexibility. Both figures emphasize the complex network of companies, organizations, and services that drive the GenAI landscape globally.

⁴A. Meige, Z. Huczok, R. Eagar, et al., *Generative artificial intelligence: Toward a new civilization?* Arthur D. Little, 2023. Available at: <https://www.adlittle.com/en/insights/report/generative-artificial-intelligence-toward-new-civilization>.

⁵Cf. previous footnote.

The book can be read in multiple ways. The reader interested solely in an introduction to LLMs and a survey of relevant literature can consider Chaps. 2, 3, 4, and 5. Conversely, the researcher and/or practitioner more interested in practical notions and code development can consider Chaps. 6, 7, 8, and 9. Those readers interested in our specific research vision and outcomes can consider Chaps. 9 and 11, in which also case studies investigated by us are considered. Multidisciplinary approaches are considered in Chap. 10, in which ethics and governance of LLMs are discussed.

Specifically, the chapters are organized as follows:

- Chapter 2, “History of Large Language Models,” presents a historical analysis of AI evolution, from symbolic AI to generative AI. The chapter explores key milestones in natural language processing (NLP) and the rise of LLMs. It presents a timeline of innovations leading to modern language models.
- Chapter 3, “Large Language Models,” for dummies, explains fundamental concepts behind LLMs in an accessible way. It covers the attention mechanism, deep learning foundations, and pre-training methods. It also introduces advanced topics such as fine-tuning and retrieval-augmented generation (RAG).
- Chapter 4, “A Reference Framework for Information Systems,” details the IS life cycle, from planning to maintenance. It also discusses data analytics and knowledge management in ISs.
- Chapter 5, “Exploring Large Language Models in Information Systems: A Survey,” presents a review on LLM applications in ISs. It categorizes research findings based on the IS life cycle. The chapter identifies emerging trends and gaps in the current adoption of LLMs.
- Chapter 6, “Programming Large Language Models,” covers the technical aspects of working with LLMs. It introduces tools and libraries, fine-tuning techniques, and prompt engineering. The chapter also explores efficient training methods for LLM-based applications.
- Chapter 7, “Retrieval Augmented Generation,” provides an in-depth explanation of RAG, a technique that combines information retrieval with language generation. The chapter details the retrieval phase, generation mechanisms, and real-world applications. It includes practical implementations using LlamaIndex and LangChain.
- Chapter 8, “Large Language Model Agents,” covers key components like memory, planning, and tools. It also presents multi-agent systems.
- Chapter 9, “Applications of Large Language Models in Information Systems,” presents our applications of LLMs in IS Engineering, including industrial data retrieval and process automation. The chapter introduces architectures such as COSMADS and NL2ProcessOps. Practical case studies illustrate the impact of LLMs in different domains.
- Chapter 10, “Ethics and Governance of Large Language Models,” provides a discussion on the ethical implications and governance frameworks for LLMs. The chapter analyzes different regulatory approaches across countries. It also covers key principles for responsible AI deployment.

- Chapter 11, “Epilogue,” is a reflective conclusion summarizing the book’s insights and key takeaways. The chapter discusses open challenges in IS Engineering with LLMs. It also provides future research directions and final thoughts from the authors.

Chapter 2

History of Large Language Models



Francesca De Luzi 

Abstract This chapter explores the historical development of artificial intelligence (AI) and natural language processing (NLP), focusing on the evolution of language modeling. We begin by outlining the foundations of AI, from symbolic approaches to the emergence of sub-symbolic approaches such as machine learning (ML) and deep learning (DL). We examine NLP advancements, distinguishing between natural language understanding (NLU) and natural language generation (NLG) and highlighting the innovative role of the transformers architectures. The chapter then delves into the progression of language models (LMs), from early statistical LMs to neural LMs and the modern era of large language models (LLMs). We present a historical timeline of key LLMs—including GPT, PaLM, and LLaMA—alongside their technological milestones, shedding light on their increasing scale, capabilities, and impact.

2.1 Artificial Intelligence

The term *artificial intelligence (AI)* originated in the summer of 1956 when the congress titled “The Dartmouth Summer Research Project on Artificial Intelligence” organized by John McCarthy took place in Hanover. AI was defined as *the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable* [30].

Since then, the history of AI has been characterized by many ups and downs. Significant accelerations have been observed in the development of increasingly sophisticated mathematical models designed to mimic certain brain functions. However, progress was hindered by challenges related to hardware development.

F. De Luzi (✉)
Sapienza Università di Roma, Rome, Italy
e-mail: deluzi@diag.uniroma1.it

In 1966, Joseph Weizenbaum published ELIZA,¹ which is considered a milestone in the evolution of AI. However, during its development, natural language processing was a challenging task, requiring substantial effort and innovation. In the same year, researchers developed “Shakey,” the first mobile robot capable of perceiving and reasoning about its surroundings. Despite the successes achieved by early artificial neural networks, to which he himself had contributed, Marvin Minsky, together with Seymour Papert, highlighted their limitations in 1969, as was also the case with ELIZA. Subsequently, there was a significant reduction in funding for research on artificial neural networks, marking the beginning of the first so-called AI winter.

The evolution of AI continued, with a particular focus on symbolic logic. During the 1970s, the first expert systems, characterized by if-then and rule-based reasoning approaches, were created, gaining significant traction in the following decades where computers from Apple and IBM outperformed them also being more affordable, aligning with Moore’s Law.²

By the end of the 1980s, the collapse of the entire sector occurred, initiating the second “AI winter.” Interest in AI research significantly declined until the 2000s, which was marked by substantial improvements in computer performance. Specifically, renewed interest in neural networks and machine learning emerged, driven by the availability of more powerful hardware and abundant data. Computers have become increasingly powerful, more affordable, and equipped with access to public clouds and, most importantly, superior GPUs (graphics processing units) that enable significant computational capacity to process the huge amount of data exponentially growing nowadays.

Nowadays, we are living in the boom of AI (also known as the “AI spring”), which refers to the ongoing period of unprecedented development in the AI field. Central to this boom is the rapid advancement in generative AI, an area of AI gaining popularity with the development of new deep learning models. Figure 2.1 illustrates the categorization of AI into two main branches: symbolic and sub-symbolic AI. Symbolic AI relies on rule-based systems to derive logical conclusions, while sub-symbolic AI focuses on learning patterns from data. Sub-symbolic AI encompasses machine learning (ML), its subset deep learning (DL), and the more recent generative AI (GAI), as explained in the following sections.

¹ ELIZA is a chatbot, i.e., software designed to simulate a conversation with a human, intended to behave like a psychotherapist (<https://web.njit.edu/~ronkowitz/eliza.html>).

² Proposed by Gordon Moore, co-founder of Intel, in 1965, the law observes that the number of transistors on a microchip doubles approximately every two years, leading to exponential growth in computing power while reducing relative costs.

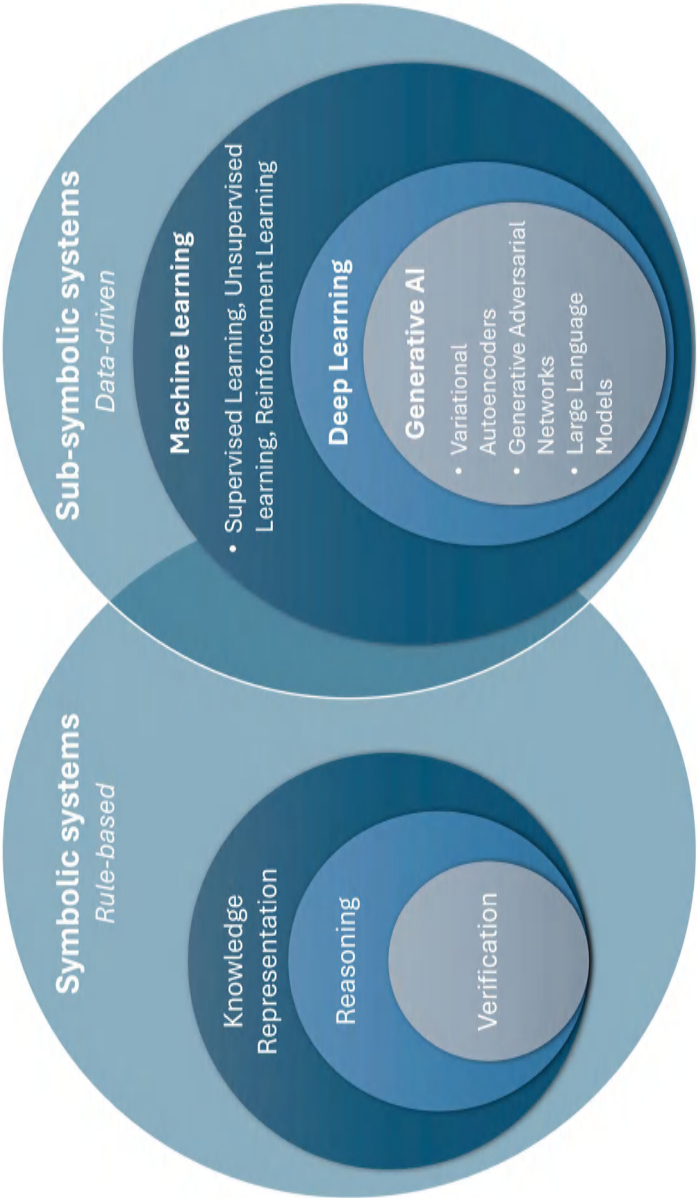


Fig. 2.1 AI approaches

2.1.1 *Symbolic AI*

The classical approach to AI involves the use of formal languages that automatically apply logical inference rules. This approach aimed to build intelligent machines through programs that employ logical rules [36]. The focus was initially on tasks similar to those performed by humans using computers to understand human intelligence. Techniques falling into this category rely on explicit symbolic methods, such as formal methods and programming languages, and are primarily used for deductive knowledge. They involve first-order logic rules, as well as other methods like ontologies, decision trees, planning, and reasoning. Symbolic AI is often associated with knowledge bases and expert systems. One key feature of symbolic methods is their ability to explain and reason about conclusions, with intermediate steps being easily understandable. The human-readable computation flow makes these systems easier to debug, explain, and control. Furthermore, thanks to the modular structure of rules, they allow the insertion or removal of discrete and autonomous knowledge units from a knowledge base, promoting rule modularity. Knowledge interoperability in closely related applications is also facilitated, and they are suitable for abstract problems, as they are not highly dependent on input data.

However, symbolic methods face challenges in cases where datasets exhibit data quality issues and noise, leading to suboptimal results and weaknesses, in addition to difficulties in maintaining rule bases that require complex verification and validation processes. In terms of applications, symbolic methods excel in well-defined and static problems, but they struggle with real-time dynamic assessments and handling massive empirical data streams. Examples of symbolic AI include logic [35], planning [10], and gameplay [15].

2.1.2 *Sub-symbolic AI*

Over time, it became apparent that AI's real challenge was acquiring knowledge by extracting patterns from raw external data. This capacity for automatic learning was termed *machine learning* (ML). ML is defined as software that creates algorithms that improve its performance over time, measured within a specific task, after undergoing a series of experiences. The goal was for AI to learn to solve problems that humans intuitively solve automatically, such as recognizing words or faces in images, activities that cannot be precisely described mathematically. Sub-symbolic AI includes statistical learning methods, such as Bayesian learning, deep learning (DL), backpropagation, and genetic algorithms.

2.1.2.1 Machine Learning

ML refers to a set of mechanisms within the realm of AI that enable an intelligent machine to learn and optimize their performance autonomously by creating self-adaptive algorithms that derive knowledge from data [34]. These algorithms are designed to adapt and improve their performance as they process increasing amounts of data. Training data is crucial for ML, and its quality determines the system's accuracy and resilience to biases or distortions. There are three basic ML paradigms [34]:

- *Supervised learning*: it relies on algorithms trained to provide desired results by learning from a set of input labeled data, where each input is paired with a corresponding output. It is commonly used for classification tasks or to predict specific behaviors. Once trained, the algorithm can generalize from the provided data to make accurate predictions on new, unseen data.
- *Unsupervised learning*: in this approach, algorithms operate without explicit guidance from humans regarding the desired output. Instead, it autonomously identifies patterns, structures, and relationships within the data. The primary objective is to discover hidden or previously unknown relationships that may go over human deductive abilities. While unsupervised learning still relies on data, it differs from supervised learning as the input data is unlabeled, meaning it lacks annotations.
- *Reinforcement learning*: this technique enables a machine to make sequential decisions to solve a problem. In this case, algorithms function much like a game, where each action is associated with a reward or penalty based on feedback from its environment. While the system receives feedback, it is not provided with explicit instructions on how to achieve its objective. Instead, it learns through trial and error, gradually discovering strategies to maximize cumulative rewards. This approach is particularly effective in scenarios with limited data, where the system's current state influences future decisions.

2.1.2.2 Deep Learning

DL is a subcategory of ML and refers to a family of techniques within AI that are inspired by the structure and function of the brain [23]. These techniques are widely applied in areas such as computer vision, audio and speech recognition, and natural language processing.

A key feature of DL is the usage of algorithms with multiple layers of non-linear hidden units. These layers are used in cascade to perform tasks that can be classified as problems of transforming features extracted from the data. Each layer uses the output of the preceding layer as input. They incorporate multiple layers of representation, which can be conceptualized as various levels of abstraction, thereby forming a hierarchy of concepts.

DL algorithms simulate certain mechanisms of the human brain, enabling machines not only to learn autonomously and not only as ML systems do but also at multiple levels of abstraction. While DL systems require significant computational resources, their ability to scale with increasing data and complexity distinguishes them from traditional ML, which often reach limitations in performance despite additional data. DL systems improve their performance as data increases, whereas ML applications, once a certain level of performance is reached, are not scalable even with additional examples and training data added to the neural network.

Training a DL system typically involves the use of labeled data, which helps guide the learning process by providing clear examples of inputs and corresponding desired outputs.

DL is already applied in various fields today. Examples include self-driving cars, drones, and robots used for parcel delivery or emergency management. Furthermore, facial recognition [16] and predictive maintenance [41] are among the most relevant emerging applications.

2.1.3 *Toward Generative AI*

Generative artificial intelligence (GAI) represents a major step further in machines' ability to create innovative content autonomously. Unlike traditional approaches such as ML and DL, which focus on modeling the relationship between input features and output labels, GAI learns the intrinsic data structure and generation processes [17]. As defined by [51], *GAI consists of a set of algorithms designed to learn the distribution of a dataset so that its underlying patterns can be characterized and samples that resemble the original data can be generated*. One distinctive feature of GAI is its reliance on *prompts* as a means of interaction. Users can engage with these models using natural language, instructing them to produce desired outputs. This encourages users to employ a trial-and-error approach, refining their input to achieve the intended results. *Prompt engineering*, an emerging field, focuses on systematically designing and refining prompts to optimize the quality and relevance of generated outputs. These systems, often embodied by models such as LLMs, possess the capability to generate diverse and creative outputs.

The applications of GAI span across various domains, revolutionizing sectors such as education, healthcare, manufacturing, retailing, and banking. In addition, it is not confined to a singular modality; instead, GAI can create diverse content, including text, images, videos, code, sound, and even complex structures like molecules or 3D renderings. In the context of content creation, tools like ChatGPT³ are pivotal, in efficiently producing text. Image generation capabilities span from creating abstract art to enhancing image quality by reducing noise. Notably, GAI

³ ChatGPT is an AI-based chatbot model developed by OpenAI that specializes in conversations with a human user.

extends to code generation, automating tasks such as developing Website templates or entire software applications. Additionally, it plays a role in speech-to-speech conversion, facilitating real-time language translation, and contributes to audio generation, enabling the creation of music, sound effects, and voice-overs. In addition, GAI is valuable in generating synthetic data, especially when authentic data is limited, providing unique datasets for training ML models.

2.2 Natural Language Processing (NLP)

Natural language processing (NLP) forms the foundation for GAI in language understanding and generation. While NLP focuses on analyzing, interpreting, and processing human language, GAI builds on these advancements to autonomously produce coherent and contextually relevant text. The evolution of NLP techniques, particularly through DL, has led to powerful language models that drive modern generative systems. This synergy has enabled breakthroughs in applications such as machine translation, text summarization, and conversational agents.

Specifically, NLP is the field of research and application concerning the understanding and production of written texts in natural language. According to [26], NLP is defined as a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts, at one or more levels of linguistic analysis for the purpose of achieving humanlike language processing for a range of tasks or applications. Impressive progress has been made in NLP, surpassing what many previously thought was impossible. The pivotal shift in NLP occurred with the introduction of methods capable of acquiring robust, versatile representations of text from extensive self-supervised textual datasets. This breakthrough enabled the transfer of such representations to various tasks, leading to significant strides in both language understanding and generation. These techniques rely on the fundamental principle that the meaning of a word is intricately linked to the context in which it appears. This principle underpins learning tasks that do not require annotated data and is closely tied to the challenge of language modeling.

As mentioned, NLP enables the understanding and generation of natural language. Techniques for understanding human language are referred to as natural language understanding (NLU), while generative techniques are called natural language generation (NLG) [21].

2.2.1 Natural Language Understanding (NLU)

The primary goal of NLU is to process and interpret natural language contained in documents and texts. NLU encompasses a set of techniques and methodologies to extract meaning, structured information, and semantic relationships from unstruc-

tured texts. Below are among the most commonly used NLU techniques we can find:

- Part-of-speech tagging: identifies parts of speech (nouns, verbs, adjectives) within sentences.
- Named entity recognition: identifies and classifies entities such as person names, locations, organizations, dates, and more.
- Text classification: assigns a document or a sentence to one or more categories. For example, this technique is used for emotion detection, identifying the main topic of a speech, and recognizing spam emails.

2.2.2 Natural Language Generation (NLG)

The primary goal of NLG, also referred to as text generation, is to create an end-to-end solution with minimal human intervention by automatically learning an input-to-output mapping from the data [25]. In essence, NLG involves the process of finding the optimal sequence of words or tokens that can be generated based on a given source of information. This is typically achieved by determining the conditional probability of each token in the generated sequence, given the previous tokens and the source sequence. The final goal is to find the sequence that maximizes the probability of generating the desired output.

2.2.2.1 Language Modeling

Given the importance of tokens' conditional distribution in text generation, the (pre-trained) language model has acquired a central role in NLG in the last decade [25]. A language model is a statistical model whose objective is to learn a function to estimate the probability of a particular sequence of tokens. After the training of a language model is completed, the learned probability distributions can be used directly for text generation, which involves generating the next word in a sentence based on the probabilities estimated by the language model [39]. Alternatively, the language model can be fine-tuned on a small dataset to adapt it for a specific supervised task. The latter is the main idea behind the usage of the PLM [25].

2.2.2.2 Transformers

The paradigm of NLG has been deeply influenced by encoder-decoder models [9]: the encoder maps the input sequence into fixed-size low-dimensional vectors, known as input embeddings; the decoder then generates a target text based on these embeddings. Unlike earlier statistical approaches that relied heavily on explicit feature engineering, encoder-decoder architecture can extract the important

features automatically during training [25]. This makes it easier to capture complex relationships between inputs and outputs, resulting in better performance on text generation tasks.

Among the various encoder-decoder architectures, transformers [52] have proven to be particularly successful in learning precise latent feature representations⁴ for language modeling [9]. Both the encoder and decoder of the original transformer can be described as composed by the repetition of identical blocks. The encoder's block has a multi-head self-attention module and a position-wise feed-forward network [27]. The encoder has the task of mapping the input to a latent representation of the whole sequence [52]. The decoder block has cross-attention modules added between the multi-head self-attention modules and the position-wise feed-forward networks (FFNs), and its task is to transform the latent representation in the desired target [27]. A key mechanism in transformers is the attention mechanism.⁵ This mechanism enables the model to dynamically weigh different words in the input sequence, improving its ability to capture long-range dependencies and contextual information effectively.

2.3 Stages of LMs

In this section, we explore the stages of LMs, providing a brief introduction to the different models. Figure 2.2 illustrates a visual representation of the history of LMs development [7, 54].

2.3.1 Statistical Language Models

Statistical Language Models (SLM) [18, 40, 43] emerged in the 1990s, leveraging statistical learning methods to predict word sequences based on Markov's assumption. This principle states that the prediction of the next word depends only on the most recent context. The resulting models, known as n-gram models, define fixed-length contexts for their predictions. They suffer from a fundamental limitation known as the "curse of dimensionality." As the model order increases, the number of possible word combinations grows exponentially, making it challenging to estimate transition probabilities accurately and store them.

⁴ A latent feature representation refers to the underlying, hidden characteristics or variables within a dataset that are not directly observed but are inferred from the available data through patterns, relationships, or statistical methods.

⁵ Attention is a technique that allows the model to focus on the most relevant parts of the input sequence while processing it. In transformers, self-attention computes attention scores that determine how much each word in the input should contribute to the representation of another word, enabling the model to capture dependencies regardless of distance.

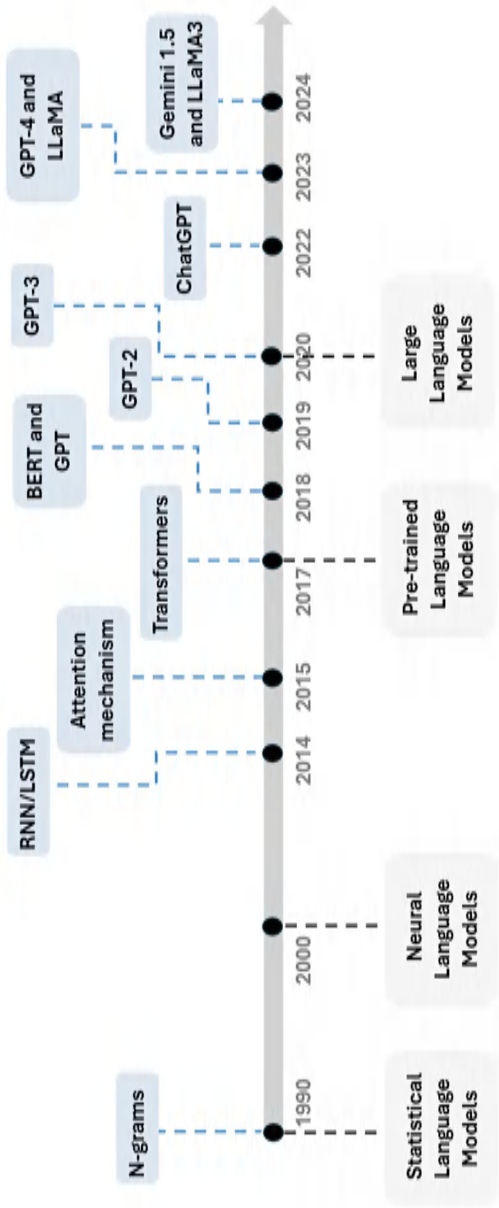


Fig. 2.2 History and development of language models

2.3.2 *Neural Language Models*

Neural language models (NLM) represented a paradigm shift by incorporating neural networks to model the probability of word sequences [22, 32]. A significant contribution in this field was the introduction of distributed word representations [3], which enabled the prediction function to be conditioned on aggregated context features, represented as distributed vectors. This approach allowed the development of unified, end-to-end solutions for a broad range of NLP tasks. Another example is the word2vec model [33], which used a shallow neural network to learn distributed word representations. These word vectors demonstrated great effectiveness in capturing both the meaning and structure of language, representing a significant advancement in NLP applications.

2.3.3 *Pre-trained Language Models*

Pre-trained Language Models (PLM) revolutionized NLP by introducing the “pre-training and fine-tuning” paradigm. The advent of models like ELMo [37] and ULMFiT [14], based on long short-term memory (LSTM) networks [13], pioneered this approach. ELMo introduced dynamic, task-specific word representations, while ULMFiT focused on layer-wise fine-tuning of the language model, adding classifier layers. These models highlighted the power of using large pre-training datasets along with flexible architectures. This foundational work paved the way for later innovations like BERT [8], which further refined the paradigm with transformer architectures and novel pre-training tasks such as masked language modeling (MLM), establishing a new standard for representation learning in NLP.

2.3.4 *Large Language Models*

Large language models represent the latest evolution of PLMs, with a focus on scalability both in terms of model size and the volume of training data. Based on observed trends, increasing the number of model parameters and the size of the dataset leads to notable improvements in performance on various tasks. For example, GPT-3 [5], with its 175 billion parameters, demonstrated emergent abilities such as in-context learning, which allows it to perform tasks without requiring explicit task-specific training, solving complex problems with minimal instructions. These capabilities set LLMs apart from their smaller predecessors, like GPT-2 [39] or BERT, and have garnered significant attention within the AI research community. The popularity of LLMs has been further increased with practical applications like ChatGPT, which fine-tuned the GPT series for conversational tasks. The main LLMs so far are shown in Fig. 2.3. We present below the three most

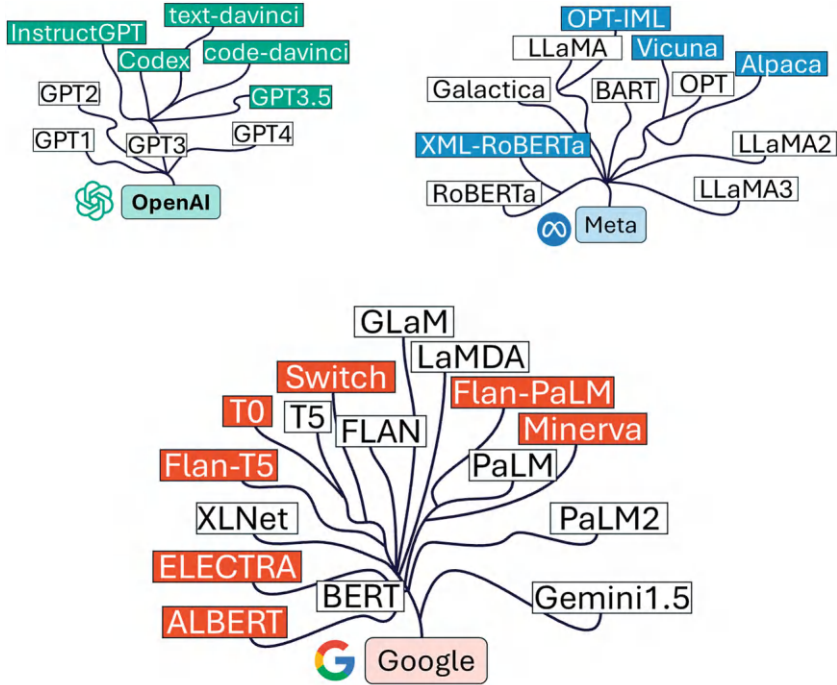


Fig. 2.3 Popular LLMs

popular LLMs families currently used by users, as well as an overview of their features in Table 2.1.

2.3.4.1 Generative Pre-trained Transformer (GPT) Family

Developed by OpenAI, GPT was first introduced with GPT-1 [38] in 2018 and had 117 million parameters. In 2019 it was followed by GPT-2 [39], which boasted 1.5 billion parameters, significantly surpassing the learning capabilities of GPT-1, thanks also to training on a much larger dataset. In 2020 OpenAI releases GPT-3 [5], which has revolutionized AI reaching 175 billion parameters. The GPT models are based on the Transformer architecture and are designed as generative language models. As the series progressed, the models grew in size and capabilities. GPT-3 quickly became the focal point for research on LLMs and served as the foundation of ChatGPT software, released in November 2022 by OpenAI. Released in March 2022, GPT-3.5 augmented the system’s capabilities, becoming more streamlined and less costly.

Table 2.1 Overview of popular LLM families

| Type | Model | #Parameters | Release | Open source | Training dataset |
|---------------------|------------------|---|---------|-------------|--|
| <i>GPT family</i> | GPT-1 | 120M | 2018 | ✓ | BooksCorpus |
| | GPT-2 | 1.5B | 2019 | ✓ | Reddit outbound |
| | GPT-3 | 125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B | 2020 | X | Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia |
| | GPT-4 | 1.76T | 2023 | X | – |
| <i>PaLM family</i> | PaLM | 8B, 62B, 540B | 2022 | X | Web documents, books, Wikipedia, conversations, GitHub code |
| | PaLM-2 | 340B | 2023 | ✓ | Web documents, books, code, mathematics, conversational data |
| | Med-PaLM | 540B | 2022 | X | HealthSearchQA, MedicationQA, LiveQA |
| <i>LLaMA family</i> | LLaMA1 | 7B, 13B, 33B, 65B | 2023 | ✓ | Online sources |
| | LLaMA2 | 7B, 13B, 34B, 70B | 2023 | ✓ | Online sources |
| | Alpaca | 7B | 2023 | ✓ | GPT-3.5 |
| | Vicuna | 13B | 2023 | ✓ | GPT-3.5 |
| | LLaMA3 | 8B, 70B | 2024 | ✓ | Online sources |
| | LLaMA3.1 | 8B, 70B, 405B | 2024 | ✓ | Online sources |
| | LLaMA3.2 | 1B, 3B | 2024 | ✓ | Online sources |
| <i>Other LLMs</i> | LLaMA3.3 | 70B | 2024 | ✓ | Online sources |
| | DeepSeek-Coder | 1.3B, 6.7B, 33B | 2023 | ✓ | GitHub's Markdown and StackExchange |
| | DeepSeek-LLM | 7B, 67B | 2024 | ✓ | GitHub's Markdown and StackExchange |
| | DeepSeek-V2 | 236B (total), 21B (active) | 2024 | ✓ | Internet data |
| | DeepSeek-V3-Base | 671B (total), 37B (active) | 2024 | ✓ | |
| | DeepSeek-R1 | 671B (total), 37B (active) | 2025 | ✓ | |

(continued)

Table 2.1 (continued)

| Type | Model | #Parameters | Release | Open source | Training dataset |
|------|--------------|-----------------------------------|---------|-------------|---|
| | Mistral-7B | 7B | 2023 | ✓ | Instruction dataset |
| | Mixtral-8x7B | 46.7B | 2024 | ✓ | Instruction dataset |
| | Gemma 1 | 2B, 7B | 2024 | ✓ | English data from Web documents, mathematics, and code |
| | Gemma 2 | 2B, 9B, 27B | 2024 | ✓ | English data from Web documents, code, and science articles |
| | Qwen | 1.8B, 7B, 14B | 2023 | ✓ | Public Web documents, encyclopedia, books, codes |
| | Qwen2.5-Max | 0.5B, 1.5B, 3B, 7B, 14B, 32B, 72B | 2025 | ✓ | Qwen2.5-Math, Qwen2.5-Coder, synthetic data (mathematics, code) |

More recently, OpenAI presented GPT-4 [1] in March 2023. Unlike previous models, OpenAI has not publicly stated the number of parameters of the GPT-4, which is estimated at about one trillion. In addition to a larger number of parameters, this model has been trained on an even more diversified and extended dataset, through the introduction of zero-shot and one-shot learning capabilities, allowing the model to perform tasks with few or no examples provided, thus making GPT-4 more humanlike than any previous GPT model. In addition, GPT-4 has the ability to use computer vision to interpret visual data and accepts both text and images as input. GPT-4o, released in May 2024, represents the latest advance in OpenAI language models and offers more concise and disciplined answers, better structured explanations, and a particular improvement in creative writing activities.

2.3.4.2 Pathways Language Model (PaLM) Family

The PaLM family are developed by Google and are largely closed source, offering limited public access but contributing significantly to advancements in few-shot learning and complex reasoning tasks. PaLM was trained using a combination of English and multilingual data, including high-quality Web pages, books, Wikipedia, conversations, and code from GitHub.

The first PaLM [6] model was launched in 2022 (540 billion parameters), but only thanks to PaLM-2 [2] (340 billion parameters), the model behind Google Bard, the provider proves able to offer conversational features similar to ChatGPT. Google Bard is an experimental conversational AI service, previously based on the LaMDA

[48] model (137 billion parameters), and released in March 2023. After LaMDA, Google decided to move to the more advanced PaLM-2 model for its AI products. Bard was renamed to Gemini on February 8, 2024 (Gemini was already the LLM that powered Bard). With the move from Gemini 1.0 [44] to Gemini 1.5 [45], the model has improved its ability to handle extended contexts and optimized language performance. The latest version, Gemini 2.0,⁶ represents a significant technological leap, with improvements in speed, accuracy, and functionality.

Google also introduced *Minerva* [24], a language model based on PaLM that solves mathematical and scientific problems by explaining the process through formulas and text. Another model from the PaLM family is *Med-PaLM* [42], a variant of PaLM-2 trained on a set of medical data, previously verified and screened, which Google plans to implement in a product dedicated to providing medical information.

2.3.4.3 Large Language Model Meta AI (LLaMA) Family

LLaMA is a collection of open-source language models developed by Meta. Unlike GPT models, these models' weights are published to the research community in a non-commercial, licensed format. The LLaMA models are auto-regressive decoder-only models, and they are based on transformer architecture with some modifications and were trained using 15 trillion text tokens collected from public sources. The first group of LLaMA models was released in 2023, starting with LLaMA 1 [49] (65 billion parameters), LLaMA 2 [50] (70 billion parameters), and LLaMA 3 [31] in 2024 (70 billion parameters). The latest version is Llama 3.3,⁷ released in December 2024, and it is a text-only instruction tuned model that offers improved performance compared to previous Llama 3.1 and Llama 3.2, especially when used for text-only applications. Llama 3.3 is therefore only provided as an instruction-tuned model; a pretrained version is not available. The main feature is that all the LLaMA models are open source, anyone can download LLaMA and use it locally, and developers and researchers can access the source code and customize it to meet specific needs.

2.3.4.4 Other Representative LLMs

In addition to the models discussed in the previous sections, other popular LLMs do not belong to these three families but have achieved remarkable performance and pushed the LLM field forward. We briefly describe these LLMs in this subsection.

⁶ <https://blog.google/technology/google-deepmind/>.

⁷ https://www.llama.com/docs/model-cards-and-prompt-formats/_llama3_3/.

- *DeepSeek-R1* is an open-source model based on reinforcement learning techniques, developed by the Chinese company DeepSeek, and offers emerging self-reflection capabilities, making it a valuable resource, thanks to its accessibility and flexibility. In 2023 DeepSeek released the first series of models, DeepSeek-Coder [12] and DeepSeek-LLM [4], while in 2024 it released the series DeepSeek-V2 [28], DeepSeek-V3-Base [29], and DeepSeek-R1 [11]. In January 2025, it released its first free chatbot app, based on the DeepSeek-R1 model, surpassing ChatGPT as the most downloaded free app on the iOS App Store in the USA.
- *Mixtral-8x7B* [20] is the second LLM released by mistral.ai, after Mistral-7B [19], a decoder-only transformer trained on a smaller set of data (7 billion parameters). Instead, Mixtral is a Mixture-of-Experts (MoE) model that activates only part of the parameters during inference, making it more efficient than models of the same size.
- *Gemma* is a new set of open models based on a decoder transformer and has been pre-trained on large datasets. Gemma 2 [47] is available in sizes 9B and 27B, while Gemma 1 [46] is available in sizes 2B and 7B. It is text-to-text LLM, decoders with pre-trained and tuned variants on the instructions. Gemma is not as large and powerful as the popular AI models (like OpenAI's ChatGPT-4 and Google's Gemini), but it is a lightweight compact model that can work on laptops or desktops because they have faster inference speeds and lower computational demands.
- *Qwen* is a series of open-source decoder-only models, pre-trained on over 2.2 trillion tokens and consisting of different model sizes. The Qwen models are developed by Chinese giant Alibaba and feature new neural architectures that improve written and spoken language processing as well as reasoning and contextual understanding. The latest model, Qwen2.5-Max [53], stands out for its multimodality, so it is able to understand both text and images and generate video content, and it has been pre-trained on over 20 trillion tokens and further post-trained with reinforcement learning from human feedback (RLHF) methodologies.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Anil, R., Dai, A.M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., et al.: Palm 2 technical report. arXiv preprint arXiv:2305.10403 (2023)
3. Bengio, Y., Ducharme, R., Vincent, P.: A neural probabilistic language model. In: T.K. Leen, T.G. Dietterich, V. Tresp (eds.) *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000*, Denver, CO, USA, pp. 932–938. MIT Press (2000)

4. Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., et al.: Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954* (2024)
5. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020)
6. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**(240), 1–113 (2023)
7. Chu, Z., Ni, S., Wang, Z., Feng, X., Li, C., Hu, X., Xu, R., Yang, M., Zhang, W.: History, development, and principles of large language models—an introductory survey. *CoRR abs/2402.06853* (2024)
8. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805* (2018)
9. Dong, C., Li, Y., Gong, H., Chen, M., Li, J., Shen, Y., Yang, M.: A survey of natural language generation. *ACM Comput. Surv.* **55**(8), 1–38 (2022)
10. Geffner, H., Bonet, B.: *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers (2013)
11. Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al.: Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025)
12. Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y., et al.: Deepseek-coder: When the large language model meets programming—the rise of code intelligence. *arXiv preprint arXiv:2401.14196* (2024)
13. Hochreiter, S.: *Long Short-Term Memory*. Neural Computation MIT-Press (1997)
14. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018)
15. Hsu, F.H.: *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press (2022)
16. Hu, G., Yang, Y., Yi, D., Kittler, J., Christmas, W., Li, S.Z., Hospedales, T.: When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition. In: *Proceedings of the IEEE international conference on computer vision workshops*, pp. 142–150 (2015)
17. Jebara, T., Jebara, T.: Generative versus discriminative learning. *Machine learning: discriminative and generative*, pp. 17–60 (2004)
18. Jelinek, F.: *Statistical Methods for Speech Recognition*. MIT Press (1998)
19. Jiang, A.Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D.S., Casas, D.d.l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al.: Mistral 7b. *arXiv preprint arXiv:2310.06825* (2023)
20. Jiang, A.Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D.S., Casas, D.d.l., Hanna, E.B., Bressand, F., et al.: Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024)
21. Khurana, D., Koli, A., Khatter, K., Singh, S.: Natural language processing: state of the art, current trends and challenges. *Multimedia Tools Appl.* **82**(3), 3713–3744 (2023)
22. Kombrink, S., Mokolov, T., Karafiát, M., Burget, L.: Recurrent neural network based language modeling in meeting recognition. In: *12th Annual Conference of the International Speech Communication Association, INTERSPEECH 2011, Florence, Italy, August 27–31, 2011*, pp. 2877–2880. ISCA (2011)
23. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
24. Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al.: Solving quantitative reasoning problems with language models. *Adv. Neural Inf. Process. Syst.* **35**, 3843–3857 (2022)
25. Li, J., Tang, T., Zhao, W.X., Nie, J.Y., Wen, J.R.: Pretrained language models for text generation: A survey. *ACM Comput. Surv.* **56**(9), 1–39 (2022)

26. Liddy, E.D.: Natural language processing. In: Encyclopedia of Library and Information Science, 2nd edn. Marcel Decker, Inc., NY (2001)
27. Lin, T., Wang, Y., Liu, X., Qiu, X.: A survey of transformers. *AI Open* **3**, 111–132 (2021)
28. Liu, A., Feng, B., Wang, B., Wang, B., Liu, B., Zhao, C., Deng, C., Ruan, C., Dai, D., Guo, D., et al.: Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. arXiv preprint arXiv:2405.04434 (2024)
29. Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al.: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024)
30. McCarthy, J.: What is artificial intelligence? (2004). Available at <https://www-formal.stanford.edu/jmc/whatisai.pdf>
31. Meta: Introducing meta llama 3: The most capable openly available llm to date (2024). <https://ai.meta.com/blog/meta-llama-3/>. Accessed: 2025-01-31
32. Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S.: Recurrent neural network based language model. In: T. Kobayashi, K. Hirose, S. Nakamura (eds.) 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, Makuhari, Chiba, Japan, September 26–30, 2010, pp. 1045–1048. ISCA (2010)
33. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: C.J.C. Burges, L. Bottou, Z. Ghahramani, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a Meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States, pp. 3111–3119 (2013)
34. Mitchell, T.: Machine Learning, vol. 1. McGraw Hill, New York (1997)
35. Newell, A., Simon, H.: The logic theory machine—a complex information processing system. *IRE Trans. Inf. Theory* **2**(3), 61–79 (1956)
36. Newell, A., Simon, H.A.: Computer science as empirical inquiry: Symbols and search. *Commun. ACM* **19**(3), 113–126 (1976)
37. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: M.A. Walker, H. Ji, A. Stent (eds.) Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1–6, 2018, Volume 1 (Long Papers), pp. 2227–2237. Association for Computational Linguistics (2018)
38. Radford, A.: Improving Language Understanding by Generative Pre-training. Princeton University Press (2018)
39. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. *OpenAI Blog* **1**(8), 9 (2019)
40. Rosenfeld, R.: Two decades of statistical language modeling: where do we go from here? *Proc. IEEE* **88**(8), 1270–1278 (2000)
41. Serradilla, O., Zugasti, E., Rodriguez, J., Zurutuza, U.: Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Appl. Intell.* **52**(10), 10934–10964 (2022)
42. Singhal, K., Azizi, S., Tu, T., Mahdavi, S.S., Wei, J., Chung, H.W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al.: Large language models encode clinical knowledge. arXiv preprint arXiv:2212.13138 (2022)
43. Stolcke, A.: SRILM - an extensible language modeling toolkit. In: J.H.L. Hansen, B.L. Pellom (eds.) 7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16–20, 2002, pp. 901–904. ISCA (2002)
44. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)
45. Team, G., Georgiev, P., Lei, V.I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al.: Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv preprint arXiv:2403.05530 (2024)

46. Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M.S., Love, J., et al.: Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295 (2024)
47. Team, G., Riviere, M., Pathak, S., Sessa, P.G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al.: Gemma 2: Improving open language models at a practical size. arXiv preprint arXiv:2408.00118 (2024)
48. Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.T., Jin, A., Bos, T., Baker, L., Du, Y., et al.: Lambda: Language models for dialog applications. arXiv preprint arXiv:2201.08239 (2022)
49. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
50. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
51. Triguero, I., Molina, D., Poyatos, J., Del Ser, J., Herrera, F.: General purpose artificial intelligence systems (gpais): Properties, definition, taxonomy, societal implications and responsible governance. *Inf. Fusion* **103**, 102135 (2024)
52. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. (2017). https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
53. Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al.: Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115 (2024)
54. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J., Wen, J.: A survey of large language models. *CoRR* **abs/2303.18223** (2023)

Chapter 3

Large Language Models for Dummies



Marco Calamo  and Matteo Marinacci 

Abstract The rapid expansion of interconnected systems and the increasing need for intelligent data processing have driven the evolution of artificial intelligence, particularly in the field of natural language processing (NLP). Large language models (LLMs) represent a significant leap in AI capabilities, surpassing traditional machine learning approaches by leveraging deep neural networks and transformer architectures. This chapter explores the foundation of LLMs, beginning with deep learning principles and the evolution from recurrent neural networks (RNNs) to the transformer model. Key advancements such as attention mechanisms and pre-training strategies are discussed, highlighting their role in enabling LLMs to understand, generate, and manipulate humanlike text. Furthermore, the chapter examines fine-tuning techniques, prompt engineering, retrieval-augmented generation, and LLM-based agents, which enhance model performance across diverse applications. Additionally, computational efficiency and alignment with human values are addressed. By providing a comprehensive overview of LLM development, optimization, and deployment, this chapter aims to equip researchers and practitioners with insights into the current state and future potential of LLMs.

3.1 Introduction

In recent years, the growing and continuous flow of information resulting from the expansion of interconnected systems over the Internet, combined with the increase in hardware and software platforms capable of generating vast amounts of raw data within seconds, has led to the need to develop technologies that can surpass the capabilities offered by traditional machine learning (ML) algorithms.

ML has emerged as a fundamental paradigm within the field of artificial intelligence (AI) research, addressing data analysis problems where humans are unable to define the governing rules of a phenomenon [22]. This paradigm provides

M. Calamo (✉) · M. Marinacci
Sapienza Università di Roma, Rome, Italy
e-mail: calamo@diag.uniroma1.it; marinacci@diag.uniroma1.it; diag.uniroma1.it

algorithms capable of learning from data and improving performance over time without explicit knowledge of the domain rules under analysis. Traditional ML algorithms leverage statistical theory to recognize common patterns within datasets, encompassing methods such as decision trees and support vector machines [16]. However, classic ML struggles to efficiently handle raw datasets due to their size, complexity, and lack of explicit semantics. For this reason, the field of deep learning (DL) has been introduced as a subfield of ML.

The innovative work of Rosenblatt in 1958 [36] introduced the concept of artificial neurons, providing the fundamentals for modern deep neural networks (DNN) that can be composed of thousands or millions of interconnected neurons. DL uses DNNs, which are specifically designed to process raw data and store large amounts of information within the interconnections between the network layers. Since the work of Rosenblatt, the implementation of neurons and DNNs has become increasingly complex, involving functions such as non-linear operations, convolution, attention, and skip connections. This allowed the development of high-performing systems for various tasks like games [29, 40], vision [35, 42], robotics [23, 26], and natural language processing [5, 45]. While DNN architectures like recurrent neural network (RNN) and convolutional neural network (CNN) improved our ability to model complex data, they struggle with understanding and maintaining long-range dependencies, especially in sequential data like natural language. To address these limitations, the attention mechanism was introduced, and subsequently transformers, which allow models to weigh the importance of different parts of an input sequence, making it possible to understand the entire context. This mechanism laid the foundation for the development of large language model (LLM), significantly advancing the field and the concept of AI.

3.1.1 *Deep Neural Network*

The goal of a neural network (NN) is to learn how to solve a task based on data observations. A data observation is composed of several elements identified as features. The collection of observations forms the training set, which is then used to train the NN in a process known as the training process. In particular cases, each observation is associated with a ground truth label, which denotes the desired output and is used as feedback during the learning process.

A NN is a hierarchical mathematical model composed of multiple layers of interconnected artificial neurons. It takes an input observation and produces an output referred to as a prediction. The layers of an NN can be categorized based on their position within the architecture: input, hidden, and output layers (see Fig. 3.1). The lowest layer in the hierarchy is the input layer, which feeds input features to subsequent layers. Following this are one or more hidden layers, where each layer receives the output of neurons from the previous layer and provides input to the next. The outputs of a hidden layer form the latent representation of the input, and the collection of all such representations constitutes the latent space. At the top of

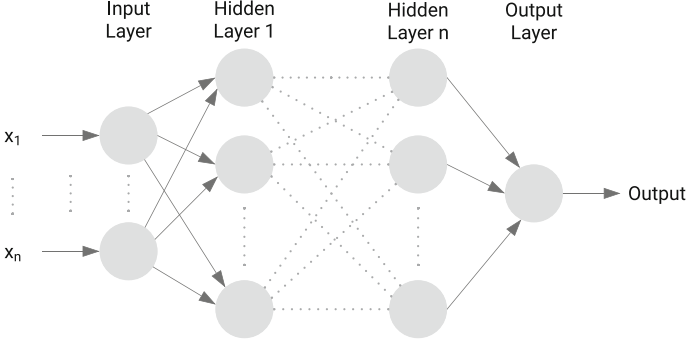
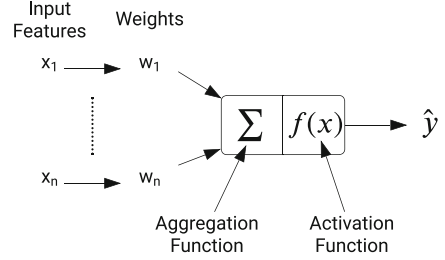


Fig. 3.1 Structure of a neural network

Fig. 3.2 Artificial neuron



the hierarchy is the output layer, which takes the output of the preceding layers and generates the final prediction. A network with at least two hidden layers is referred to as a DNN.

Each neuron between layers is connected via edges called weights, with each weight assigned a specific value. An artificial neuron (see Fig. 3.2) processes the outputs of all neurons from the preceding layer. Each neuron's output is multiplied by the weight corresponding to its connection to the current neuron. An aggregation function, such as summation, then combines the weighted outputs from the previous layer. Finally, an activation function is applied to the aggregated value to compute the neuron's output.

In the absence of hidden layers, the weights, which are adjusted during the training process to achieve the desired output [36], are updated using the *error-correction learning rule*, expressed as:

$$w_i(t+1) = w_i(t) + r(d_j - y_j(t))x_{j,i}$$

Here, w_i represents the i -th weight of the vector w , t is the training iteration, d_j is the desired output for the feature vector x_j in the training set $D = (x_1, d_1), \dots, (x_s, d_s)$, and $x_{j,i}$ is the individual feature. The parameter $r \in [0, 1]$ is the learning rate, controlling the magnitude of the update. A small value of r results in gradual adjustments, averaging past inputs, while a large value of r enables rapid adaptation but with less consideration for past errors.

In the case of hidden layers, where no predefined desired output exists (i.e., no ground truth), the *backpropagation algorithm* updates the weights of each neuron in the hidden layers. The algorithm computes the error for each neuron j as follows:

$$\delta_j = \Phi'(z_j) \sum_k \delta_k w_{jk}$$

Here, δ_j is the error for the j -th neuron in the current layer, $\Phi'(z_j)$ is the derivative of the activation function with respect to the input z_j , δ_k is the error of the k -th neuron in the next layer, and w_{jk} is the weight connecting hidden neuron j to neuron k in the next layer.

Once the error terms δ are computed for all neurons, the weights are updated using *gradient descent*. Specifically, the gradient of the loss L with respect to a weight w_{jk} , which connects neuron j in layer l to neuron k in layer $l + 1$, is given by:

$$\frac{\partial L}{\partial w_{jk}} = \delta_k a_j$$

Here, δ_k is the error term of the k -th neuron in the current layer, and a_j is the activation of the j -th neuron in the previous layer. The set of partial derivatives for all the weights in the network forms the gradient vector ∇L , which is used to update the weights as follows:

$$w^{t+1} = w^t - r \nabla L$$

These equations are further extended by optimization algorithms (e.g., Adam [20], AdaGrad [11], RMSProp). Each optimizer differs in how weights are updated at each step and in its learning schedule. The interconnected structure and training process enable each neuron to specialize in detecting particular feature correlations, allowing the network to represent virtually any function.

The training process of a network is divided into epochs, in which the whole training set is fed to the network to update weights. As the dataset may be too large to be stored in memory, it could be divided into batches of samples progressively fed during the epoch. At each step, the optimizer computes the loss associated with the prediction and accordingly updates the weights. Depending on the task in analysis, different loss functions can be employed.

When using a DNN for a particular task, there are two common approaches, i.e., training a new model from scratch or leveraging a pre-trained model. Pre-trained models are trained on large corpora to capture fundamental features shared across multiple tasks. These models serve as starting configurations for training models on smaller, task-specific datasets, a process often referred to as transfer learning. The foundation of transfer learning lies in the observation that most DNNs can be decomposed into two primary components: the feature extraction function f and the decision-making function g . Mathematically, this can be expressed as:

$$h = f(x, \theta_f)$$

$$y = g(h, \theta_g) = g(f(x, \theta_f), \theta_g) = (g \circ f)(x)$$

On the one hand, the feature extraction function f maps the input x into a meaningful representation in a latent space h , capturing feature relations. The latent space denotes the output, and θ_f denotes the parameters, i.e., the weights, of the feature extraction layers. On the other hand, the feature representation h is fed to the decision-making function g together with the parameters of the decision-making layers θ_g . For novel tasks, pre-trained models can be trained in two different ways, i.e., feature freezing or fine-tuning. In the feature freezing approach, the parameters of the feature extraction function θ_f are kept frozen, and only the parameters of the decision-making function θ_g are adjusted. In the fine-tuning approach, the pre-trained model's parameters θ_f are used to initialize the network, and the entire model is fine-tuned for the new task by updating all associated weights.

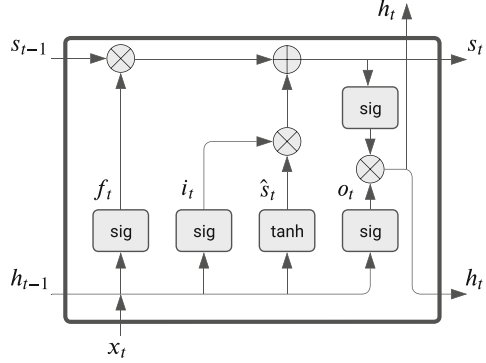
At the end of the training process, the network's performance is evaluated using a set of samples unseen during training, known as the testing dataset. The quality of the learning process is assessed through various metrics, such as accuracy or ROC-AUC, that summarize the network's performance on the testing dataset. Accuracy, which ranges from 0 (no correct predictions) to 1 (all predictions correct), is defined as the ratio between the number of correctly predicted samples and the total number of samples in the dataset. ROC-AUC, on the other hand, considers the ratio between true positive (e.g., the model predicts 1 when the ground truth is 1) and false positive (e.g., the model predicts 1 when the ground truth is 0), and it is useful when the dataset is unbalanced in the number of samples per class.

In a DNN, it is possible to connect layers and neurons in various ways, and the output of neurons can be computed by exploiting multiple activation functions. Depending on the configuration of neurons and layers, different families of architectures can be identified, each suited to solve specific types of tasks. Among these architectures, we mention long short-term memory (LSTM), which excels at processing sequential data and capturing long-term dependencies. Similarly, the attention mechanisms allow models to focus dynamically on the most relevant parts of input data, enhancing their ability to solve complex tasks such as natural language understanding and image captioning.

Long Short-Term Memory (LSTM)

LSTM [18] is a specific type of RNN designed to handle sequential data. Sequential data consists of data points that have temporal dependencies with one another, which are organized into structures called sequences. Each sequence represents a chain of data points linked by temporal relationships, with each item corresponding to a specific timestep. LSTM processes one timestep at a time, using both the current input and the output of the previous cell to generate a prediction. Thus, for a sequence with n timesteps, the LSTM produces n predictions.

In LSTMs, each cell acts as a memory unit with its own state s . At each timestep, the cell processes three inputs: the current element of the sequence x_t , the hidden

Fig. 3.3 LSTM cell

state h_{t-1} , and the previous cell state s_{t-1} . The purpose of the cell is to store, update, and propagate information across timesteps, enabling the model to capture long-term dependencies. This is achieved through the use of gates that manage the flow of information (see Fig. 3.3), including the *forget gate* f_t that determines what information to discard from the cell state, the *input gate* i_t that determines what new information to add to the cell state, and the *output gate* o_t that determines how much of the cell state contributes to the output.

The information that needs to be analyzed is obtained from the forget gate as follows:

$$f_t = \sigma(W_f \cdot [x_t, h_{t-1}] + b_f)$$

where σ is the activation function applied to the weighted matrix W_f between the current input x_t and the previous hidden state h_{t-1} plus the bias vector b_f at t .

The input gate performs the following operations needed to update the cell status:

$$i_t = \sigma(W_i \cdot [x_t, h_{t-1}] + b_i)$$

$$\hat{s}_t = \tanh(W_s \cdot [x_t, h_{t-1}] + b_s)$$

where σ is the activation function applied to the weighted matrix W_i between the input x_t and the previous hidden state h_{t-1} plus the bias vector b_i at t and \tanh is the activation function applied to the weighted matrix W_s between the input x_t and the previous hidden state h_{t-1} plus the bias vector b_s at t .

The cell state is then updated with the information from the forget and input gate:

$$s_t = f_t s_{t-1} + i_t \hat{s}_t$$

with f_t as forget gate, s_{t-1} as previous cell state, i_t as input gate, and \hat{s}_t as value generated by \tanh .

Finally, the output is obtained by multiplying the output gate o_t and the candidate output computed from the updated cell state:

$$o_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o)$$

$$h_t = o_t \tanh(s_t)$$

where σ is the activation function applied to the weighted matrix W_o between the input x_t and the previous hidden state h_{t-1} plus the bias vector b_o at t .

The architecture of LSTMs, combined with their memory cells and gating mechanisms, allows them to retain information over longer sequences compared to classical RNNs. This capability enables LSTMs to achieve superior performance on tasks involving sequential data, such as time-series analysis, language modeling, and speech recognition.

Attention Mechanism

Attention [3] is a mechanism designed to model and leverage long-range correlations between features. The attention mechanism is characterized by three main components: the queries Q , the keys K , and the values V . These components can either be vectors or linear projections of the same input [45]. The output of the attention mechanism is computed by comparing each query $q \in Q$ with the keys to generate a score value for each pair:

$$e_{q,k_i} = f_{scoring}(q, k_i)$$

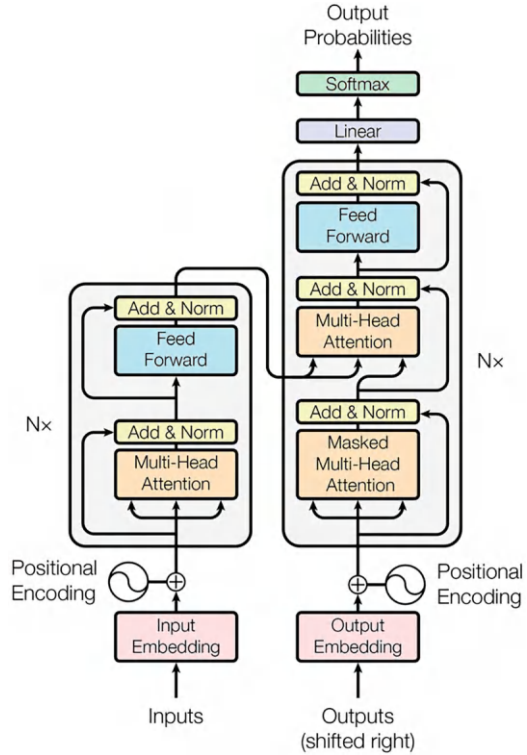
The score value e_{q,k_i} is obtained using a scoring function $f_{scoring}$, which varies based on the type of attention being implemented. Then, the scores are normalized using the softmax function to compute the attention weights:

$$w_{q,k_i} = \sigma_{soft}(e_{q,k_i})$$

Finally, the layer produces the output as a weighted sum of the values, with the weights determined by the attention weights:

$$y = Attention(q, K, V) = \sum_i w_{q,k_i} v_i$$

Different variants of the attention mechanism can be implemented by modifying the scoring function [28, 47], projection [4], or computational steps. For example, the transformer architecture [45], which comprises multiple attention layers, forms the backbone of several state-of-the-art models.

Fig. 3.4 Transformer

3.1.2 From the Attention Mechanism Toward Large Language Models

RNNs, including LSTMs, are foundational for sequence modeling and transduction tasks, such as language modeling and machine translation [3, 7, 41]. However, their sequential nature prevents parallel processing during the training process. This significantly limits computational efficiency, especially when dealing with large sequences, where processing an entire sequence in a single memory batch becomes excessively costly. To address such limitations, systems that support parallel processing are essential.

The attention mechanism has played a crucial role in overcoming the limitations of RNNs. In 2017, the Transformer architecture [45] was introduced (see Fig. 3.4), relying exclusively on the attention mechanism to capture global dependencies between input and output. This innovation eliminated the intrinsic sequentiality of RNNs, enabling parallel data processing and significantly improving computational efficiency.

The transformer employs an encoder-decoder structure, where both the encoder and decoder are composed of a stack of $N = 6$ identical layers. The encoder takes as input a sequence of symbol representations (x_1, \dots, x_n) and transforms it into a sequence of continuous representations $z = (z_1, \dots, z_n)$. The decoder uses z

along with the previously generated symbols as additional input, to produce an output sequence (y_1, \dots, y_m) of symbols one element at a time in an autoregressive manner. This means that each token is generated sequentially, conditioned on the tokens generated so far and the encoder's output.

Both the encoder and decoder begin with a learned embedding layer that converts input and output tokens into dense vector representations of fixed dimension d_{model} . These embeddings allow the model to work in a continuous, high-dimensional space where semantic relationships between tokens can be effectively captured. To encode positional information, positional embeddings are added to the token embeddings. These embeddings, having the same dimension d_{model} , ensure that the model can leverage sequential information despite lacking recurrence or convolutional structures. Positional embeddings can be implemented in various ways, such as fixed sinusoidal functions, learned embeddings, or relative encodings, each suited to different tasks and sequence characteristics.

We can now examine the structure of the encoder and decoder layer stacks. The encoder consists of two sub-layers: (i) a multi-head self-attention mechanism and (ii) a fully connected feed-forward network. On the opposite, the decoder is composed of three sub-layers: (i) a masked multi-head self-attention mechanism, also referred to as encoder-decoder attention, which prevents each position from attending to future positions, ensuring predictions for position i depend only on outputs from earlier positions, (ii) a multi-head attention applied to the output of the encoder stack, and (iii) a fully connected feed-forward network. Each multi-head attention layer processes keys, values, and queries as its input, with the exception of the encoder-decoder attention layer. In this case, the queries come from the previous decoder layer, while the keys and values are derived from the encoder's output.

The attention mechanism used in the transformer, known as Scaled Dot-Product Attention, simultaneously computes the attention function on the keys K , values V , and a set of queries stored Q . The queries and keys have a dimension of d_k , while the values have a dimension of d_v . The attention mechanism determines the relevance of each key-value pair to a query by computing a dot product between the queries and keys, scaling it by $\sqrt{d_k}$ (to stabilize gradients), and applying a softmax function to obtain the attention weights. The matrix of outputs is determined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Instead of performing a single attention function, the transformer utilizes h parallel execution of the attention mechanism, each yielding d_v -dimensional output values. Each head uses different learned linear projections to d_k, d_k , and d_v for the queries, keys, and values, respectively. These outputs are concatenated and linearly projected to produce the final values. This mechanism, known as multi-head attention, allows the model to analyze information from different representation subspaces at different positions, and its final result is computed as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_n)W^O$$

$$\text{with } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

The projection matrices are $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. This mechanism enables the model to process multiple representation subspaces simultaneously, enhancing its capacity to capture complex dependencies in the input.

In each encoder and decoder layer, the final sub-layer consists of a fully connected feed-forward network, applied independently and identically to each position in the sequence. The network is characterized by two linear transformations with a ReLU activation in between:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

where W_1 and W_2 are the weight matrices and b_1 and b_2 are the bias vectors.

Each sub-layer includes a residual connection [15] followed by a normalization [2]. The output of each sub-layer is computed as:

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

where $\text{Sublayer}(x)$ represents the function implemented by the respective sub-layer (e.g., multi-head attention or feed-forward network).

Since 2017, the transformer architecture, originally composed of an encoder-decoder structure, has been adapted into various models. In 2018, Generative Pre-Training (GPT) was introduced [33], leveraging only the transformer decoder. GPT employs a multi-layer transformer decoder [27] that applies multi-headed self-attention on input context tokens and position-wise feedforward layers to generate an output distribution over target tokens.

In the same year, another variant of the original transformer architecture was introduced, i.e., Bidirectional Encoder Representations from Transformers (BERT) [9]. BERT leverages only the transformer encoder. In contrast to GPT, which uses a left-to-right architecture where each token can only attend to previous tokens in self-attention layers, BERT employs a masked language model (MLM) pre-training objective. The MLM objective involves randomly masking a subset of input tokens and training the model to predict the masked tokens based on the surrounding context. This enables BERT to create a deep bidirectional transformer, allowing each token to attend to both its left and right contexts. This bidirectional approach enhances the ability to capture relationships within the entire sequence. Additionally, BERT incorporates a next sentence prediction task during pre-training, which jointly learns representations for text-pair relationships. Together, these objectives make BERT highly effective for tasks like question answering and natural language inference.

In 2019, a model combining Bidirectional and Auto-Regressive Transformers (BART) was introduced [24]. BART is a denoising auto-encoder based on a sequence-to-sequence architecture. During pre-training, BART corrupts text using

a noising function, such as randomly shuffling the order of sentences or applying an in-filling scheme, where spans of text are replaced with a single mask token, and then trains a sequence-to-sequence model to reconstruct the original text. BART combines a bidirectional encoder, like BERT, with a left-to-right autoregressive decoder, like GPT. However, there are key differences between BART and its predecessors. Unlike GPT, BART uses GeLU [17] activation functions instead of ReLU. Unlike BERT, its decoder layers perform cross-attention over the final hidden states of the encoder and do not include an additional feed-forward network before word prediction. The introduction of BART marked a return to the original transformer structure (i.e., encoder-decoder), demonstrating the versatility and power of this architecture for complex text reconstruction and generation tasks. Moreover, recent advancements in transformer-based models have increasingly shifted their focus toward scalability and multi-tasking, aiming to create general-purpose systems capable of excelling across a wide range of natural language processing tasks.

The development of models like BERT and BART constitutes the basis for modern large language models (LLMs). These models leverage modern architectural innovations, such as attention mechanisms and pre-training objectives, and scale them with significantly larger datasets and parameter counts. LLMs like GPT-3, PaLM, and LLaMA extend the autoregressive and bidirectional principles introduced by earlier models to perform well across diverse tasks with little to no fine-tuning. By leveraging pre-training and fine-tuning, LLMs are capable of understanding and generating humanlike text, transforming fields like conversational AI, text summarization, and machine translation.

3.2 Pre-training of Large Language Models

Nowadays, there are many available LLMs already trained on large corpora, capable of processing data and generating valuable and understandable responses [8]. The pre-training phase is a fundamental step in the development of a LLM. This phase has the role of establishing the basis of the skills of an LLM. In this step, many variables need to be addressed, such as the scale and the quality of the pre-training corpus.

The first and most important aspect that needs to be considered while pre-training an LLM is the quality of the data. The performances and capacities of a LLM heavily depend on how data has been preprocessed and the sources from which they are considered. The steps necessary in preparing data include:

1. Collect a large amount of natural language data. These data can be obtained from many sources that could be categorized as general and specialized data. General data comprehend sources like Web pages, books [13], and conversational text [49]. These sources are used to improve the generalization abilities of an LLM due to their large and diverse nature. Conversely, specialized data involving, for

instance, multilingual [6], scientific, and code [6] data are adopted to enhance specific task-solving capabilities of LLMs [30, 44].

2. Clean collected data from noisy, redundant, irrelevant, and toxic data, which may reduce the performance of an LLM. To clean data is possible to apply multiple techniques, each of which aims to address a specific problem. To remove low-quality data, it is possible to apply filtering and selection through two different approaches named classifier and heuristic respectively. The former involves the training of a selection classifier based on high-quality texts to identify and remove low-quality data [10]. The latter relies on a set of well-designed filtering rules based on Language, Metric (e.g., perplexity), Statistic (e.g., punctuation distribution, symbol-to-word ratio, sentence length), and Keyword (e.g., HTML tags, hyperlinks, boilerplates, offensive words). To remove duplicates, it is necessary to apply a technique called de-duplication that may operate at different levels, including sentences, documents, and datasets. To remove personally identifiable information is possible to employ approaches based on rules like keyword spotting. Furthermore, the cleaning step involves also preparing data for the input of a LLM. To prepare data, the text is segmented into sequences of individual tokens, and this method is called Tokenization. There exist many specializations of this method, which are called subword tokenizers, like Byte-Pair Encoding (BPE) [12, 39], WordPiece [38], and Unigram [21].
3. Schedule data to pre-training a LLM. The data schedule foresees data mixture (i.e., how much data from each source needs to be used) and data curriculum (i.e., which is the submission order of each data source).

Once the data are successfully preprocessed, they have to be encoded into the massive model parameters through the pre-training phase. To train a LLM, it is possible to adopt various approaches, among which there are:

- Language modeling (LM): a fundamental task in natural language processing that predicts the next tokens in a sequence based on their history. It is the main pre-training task when talking about LLMs. From a sequence of tokens $w = \{w_1, \dots, w_n\}$, it autoregressively predicts the target tokens w_i based on the preceding tokens $w < i$ in a sequence. The objective of this approach can be formalized as:

$$LLM(w) = \sum_{i=1}^n \log P(w_i | w < i)$$

This approach is commonly used in decoder-only models and, thanks to its autoregressive nature, allows for learning various tasks, often without the need for task-specific fine-tuning.

- Denoising autoencoding (DAE) [34]: a task in which the model is trained to recover the original parts of the input text that are intentionally corrupted. From corrupted text inputs $w_{\setminus w \sim}$ with random replaced spans, the language model is

trained to recover the replaced tokens w^\sim . The objective of this approach can be formalized as:

$$L_{DAE}(w) = \log P(w^\sim | w \setminus w^\sim)$$

- Commonly used in encoder-only or encoder-decoder models.
- Mixture-of-Denoisers (MoD): a combination of both LM and DAE [43]. Depending on the initial special tokens in input sentences, different types of denoisers could be used. The types of denoisers are S-denoiser whose objective is similar to LM, R-denoiser that is a variant of DAE where short parts of text are corrupted, and X-denoiser that is a variant of DAE with longer parts of text corrupted or more frequent corruption. Commonly used in encoder-decoder models.

3.3 Advanced Usage of Large Language Models

Baseline LLMs obtained from pre-training are not inherently specialized for solving complex tasks in specific target domains, such as chatting or document summarization. While pre-training enables the model to generate the most probable token sequence given an input, additional instructions are required to solve domain-specific tasks. *Fine-tuning* (cf. Sect. 3.3.1), *retrieval-augmented generation*¹ (cf. Sect. 3.3.2), *prompt engineering* (cf. Sect. 3.3.3), and *agents*² (cf. Sect. 3.3.4) are the main tools to specialize an LLM. Each of these techniques is orthogonal to each other (using one does not exclude using others), but each one comes with upsides and downsides based on the dataset availability and time constraints. Table 3.1 reports the consumption of resources and the advantages of using each technique.

Table 3.1 Comparison of methods in terms of dataset size, time consumed, and performance increase

| Method | Dataset size | Time consumed | Increase in performance |
|--------------------------------|--------------|---------------|-------------------------|
| Fine-tuning | High | High | Moderate/High |
| Prompt engineering | Low | Low | Low/Moderate |
| Retrieval-augmented generation | Low/Moderate | Low | Moderate |
| Agent | Variable | High | High |

¹ For further details on retrieval-augmented generation, see Chap. 7.

² For further details on Agents, see Chap. 9.

3.3.1 *Fine-Tuning*

Most LLMs today excel in general performance but often struggle with specific task-oriented problems. Fine-tuning provides significant benefits, such as reduced computational costs and the ability to leverage state-of-the-art models without developing one from scratch. Transformers-based models offer access to a wide range of pre-trained models designed for various tasks. Fine-tuning these models enhances their accuracy for specific applications like sentiment analysis, question answering, or document summarization.

Fine-tuning is the process of taking a pre-trained model and further training it on a domain-specific dataset. Different fine-tuning techniques on different datasets can lead to different outputs depending on the goal [32]. There are four kinds of fine-tuning:

- Unsupervised fine-tuning
- Supervised fine-tuning
- Instruction fine-tuning
- Alignment fine-tuning

Of these, only the last two steps are necessary to fine-tune a base LLM to obtain a model that is fully capable of solving complex tasks aligned to human requests and standards. The first two steps are important for extending the model knowledge to specific domains that were not present (or present in little parts) in the original training dataset. Table 3.2 highlights the key differences between these techniques. In the next sections, each fine-tuning step will be analyzed, explaining how it works and its ideal use cases.

3.3.1.1 **Unsupervised Fine-Tuning**

The most straightforward type of fine-tuning is the unsupervised fine-tuning. It consists of repeating the same training method presented in Sect. 3.2 using new data. This allows the LLM to acquire domain specific knowledge (e.g., medical, legal).

Table 3.2 Comparison of fine-tuning methods in terms of dataset kind and goal

| Method | Dataset kind | Goal |
|--------------------------|------------------------|--|
| Unsupervised fine-tuning | Plain text | Learn a new domain (e.g., medical) |
| Supervised fine-tuning | Plain text with labels | Learn a new task (e.g., text classification) |
| Instruction fine-tuning | Instruction dataset | Learn to follow instructions (e.g., chat) |
| Alignment fine-tuning | Instruction dataset | Learn to follow guidelines (alignment) |

3.3.1.2 Supervised Fine-Tuning

Supervised fine-tuning is the first approach that requires a specific dataset. It is used to train a model for a new downstream task (e.g., sentiment analysis). It requires an annotated dataset that exposes comprehensively the new task. Since creating a high-quality rich dataset for each new task is often unfeasible in terms of time and resources, in-prompt learning [31] (cf. Sect. 3.3.3) is frequently preferred as an alternative.

3.3.1.3 Instruction Fine-Tuning

To address the gap between the next-word prediction objective of LLMs and user-specific tasks, instruction tuning (IT), a particular type of supervised fine-tuning (SFT), is employed. This technique involves further training LLMs using pairs of instructions and corresponding outputs, enhancing their capabilities and controllability. It offers benefits such as aligning models with user objectives, improving controllability for domain-specific tasks, and enabling computationally efficient adaptation without significant architectural changes.

Despite its advantages, instruction fine-tuning faces challenges, such as crafting diverse and high-quality instruction datasets, the risk of overfitting to tasks present in the training data, and the critique that SFT may focus on superficial patterns rather than true task comprehension. Addressing these limitations requires further research into optimizing fine-tuning techniques and understanding the behavior of instruction-tuned LLMs. Figure 3.5 illustrates the architecture underlying this approach.

3.3.1.4 Alignment Fine-Tuning

Alignment fine-tuning is an approach that involves training language models through interactions with human feedback, aligning the model to human expecta-

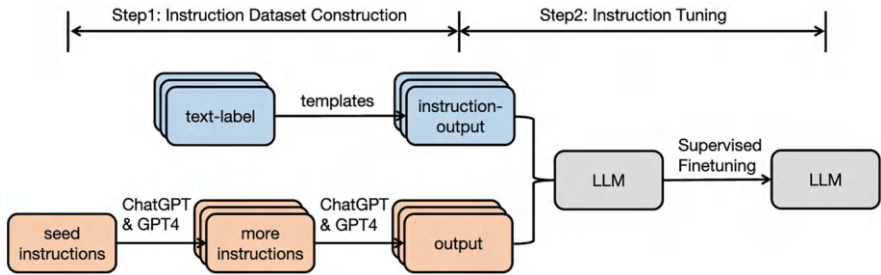


Fig. 3.5 Instruction fine-tuning general pipeline [48]

tions. This involves techniques like reinforcement learning from human feedback (RLHF), which integrates human judgment into the learning process. RLHF facilitates the continuous enhancement of language models so they produce more accurate and contextually appropriate responses. This approach not only leverages the expertise of human evaluators but also enables the model to adapt and evolve based on real-world feedback, ultimately leading to more effective and refined capabilities. Several techniques are employed in this context:

- **Reward modeling:** In this technique, the model generates several possible outputs or actions, and human evaluators rank or rate these outputs based on their quality. The model then learns to predict these human-provided rewards and adjusts its behavior to maximize the predicted rewards. Reward modeling provides a practical way to incorporate human judgment into the learning process, allowing the model to learn complex tasks that are difficult to define with a simple function. This method enables the model to learn and adapt based on human-provided incentives, ultimately enhancing its capabilities.
- **Proximal policy optimization (PPO):** It is an iterative algorithm that updates the language model's policy to maximize the expected reward. The core idea of PPO is to take actions that improve the policy while ensuring the changes are not too drastic from the previous policy. This balance is achieved by introducing a constraint on the policy update that prevents harmful large updates while still allowing beneficial small updates. This constraint is enforced by introducing a surrogate objective function with a clipped probability ratio that serves as a constraint. This approach makes the algorithm more stable and efficient compared to other reinforcement learning methods.
- **Comparative ranking:** Comparative ranking is similar to reward modeling, but in comparative ranking, the model learns from relative rankings of multiple outputs provided by human evaluators, focusing more on the comparison between different outputs. In this approach, the model generates multiple outputs or actions, and human evaluators rank these outputs based on their quality or appropriateness. The model then learns to adjust its behavior to produce outputs that are ranked higher by the evaluators. By comparing and ranking multiple outputs rather than evaluating each output in isolation, comparative ranking provides more nuanced and relative feedback to the model. This method helps the model understand the task subtleties better, leading to improved results.
- **Preference learning:** It is also known as reinforcement learning with preference feedback and focuses on training models to learn from human feedback in the form of preferences between states, actions, or trajectories. In this approach, the model generates multiple outputs, and human evaluators indicate their preference between pairs of outputs. The model then learns to adjust its behavior to produce outputs that align with the human evaluators' preferences. This method is useful when it is difficult to quantify the output quality with a numerical reward but easier to express a preference between two outputs. Preference learning allows the model to learn complex tasks based on nuanced human judgment, making it an effective technique for fine-tuning the model on real-life applications.

3.3.1.5 Efficient Fine-Tuning

Fine-tuning an LLM can be a very computationally expensive process. To mitigate this cost there exist two kinds of major optimization: *optimizing the numbers of parameters trained* (parameter efficient fine-tuning) and *optimizing the memory footprint of the model* (quantization).

Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning (PEFT) is a technique used to improve the performance of pre-trained LLMs on specific downstream tasks while minimizing the number of trainable parameters. It offers a more efficient approach by updating only a minor fraction of the model parameters during fine-tuning.

PEFT selectively modifies only a small subset of the LLM's parameters, typically by adding new layers or modifying existing ones in a task-specific manner. This approach significantly reduces the computational and storage requirements while maintaining comparable performance to full fine-tuning.

Figure 3.6 depicts the set of PEFT techniques. Here we briefly focus on reparameterized PEFT and, more specifically, on LoRA [19] for its wide diffusion and implementation. Reparameterization stands for equivalently transforming a model's architecture from one to another via transforming its parameters. In the context of PEFT, this often means constructing a low-rank parameterization to achieve the goal of parameter efficiency during training. For inference, the model can be converted to its original weight parameterization, ensuring unchanged inference speed. Earlier research studies [1] have shown that common pre-trained models exhibit an exceptionally low intrinsic dimensionality. Intrinsic SAID [1] is the pioneering work in investigating the intrinsic dimension feature during the fine-tuning of LLMs. However, the most widely recognized reparameterization technique is LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the transformer architecture, greatly reducing the number of trainable parameters for downstream tasks.

Memory-Efficient Fine-Tuning—Quantization

Quantization refers to dividing a quantity into discrete parts, often as integral multiples of a common value. Initially studied as a rounding-off process, quantization has evolved to play a pivotal role in modern computer science and machine learning. It can influence data precision and entropy, as changes in precision affect the uncertainty quantified by Shannon Entropy.

A widely used technique in this domain is 8-bit quantization, which converts floating-point data into integer format. Although 32-bit single-precision floating point has traditionally dominated deep learning applications, lower-precision formats like 16-bit floating point (e.g., fp16 or bfloat16) have gained popularity due to their compatibility with most deep learning accelerators. For inference, even lower-precision formats, including floating-point, fixed-point, and integer representations, are employed.



Fig. 3.6 PEFT taxonomy [14]

Low-precision formats offer significant performance benefits. They leverage optimized hardware pipelines for reduced bit widths, accelerating computations like convolutions and matrix multiplications. Additionally, smaller word sizes mitigate memory bandwidth constraints and improve cache utilization, enhancing the overall efficiency of computation-intensive tasks.

3.3.2 *Retrieval-Augmented Generation (RAG)*

When considering external data access, retrieval-augmented generation (RAG) [25] is likely a superior option for applications needing to access external data sources. Fine-tuning, on the other hand, is more suitable if you require the model to adjust its behavior, and writing style, or incorporate domain-specific knowledge. In terms of suppressing hallucinations and ensuring accuracy, RAG systems tend to perform better as they are less prone to generating incorrect information. If you have large domain-specific, labeled training data, fine-tuning can result in a more tailored model behavior, whereas RAG systems are robust alternatives when such data is scarce. RAG systems provide an advantage with dynamic data retrieval capabilities for environments where data frequently updates or changes. Additionally, it is crucial to ensure the transparency and interpretability of the model's decision-making process. In that case, RAG systems offer insight that is typically not available in models that are solely fine-tuned.

Large pre-trained language models have been shown to store factual knowledge in their parameters and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability to access and precisely manipulate knowledge is still limited, and hence on knowledge-intensive tasks, their performance lags behind task-specific architectures. Additionally, providing provenance for their decisions and updating their world knowledge remain open research problems. Pre-trained models with a differentiable access mechanism to explicit nonparametric memory can overcome this issue but have so far been only investigated for extractive downstream tasks. Even though the RAG techniques have advanced since it was envisioned the first time, the base architecture stayed the same as in Fig. 3.7.

3.3.3 *Prompt Engineering*

Prompt engineering is a powerful technique for expanding the functionality of LLMs. By employing task-specific instructions, or prompts, this method enables models to perform a variety of tasks without altering their core parameters. Prompts, whether in the form of natural language instructions or learned vector representations, guide the model to produce desired behaviors, making them indispensable for tasks ranging from question-answering to commonsense reasoning. There are

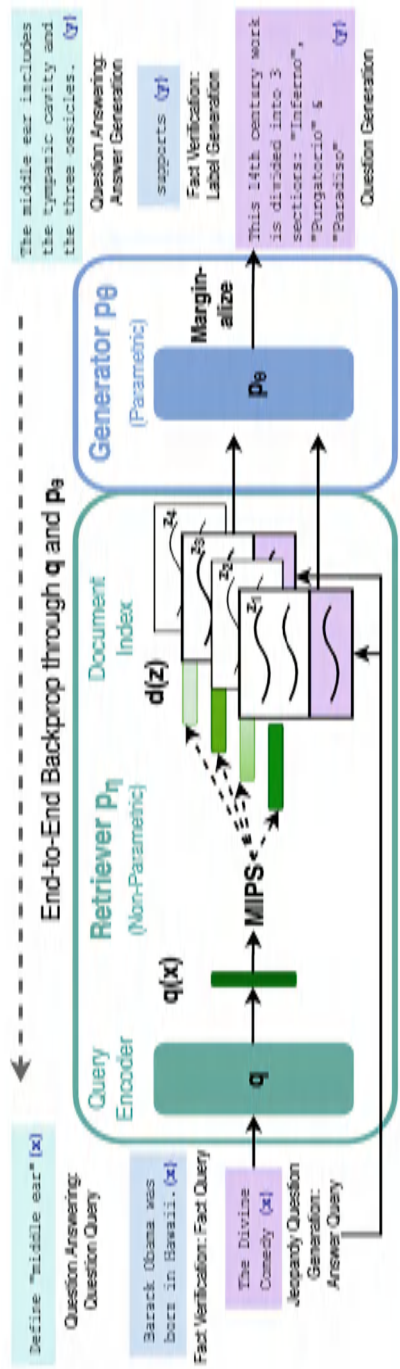


Fig. 3.7 RAG architecture [25]

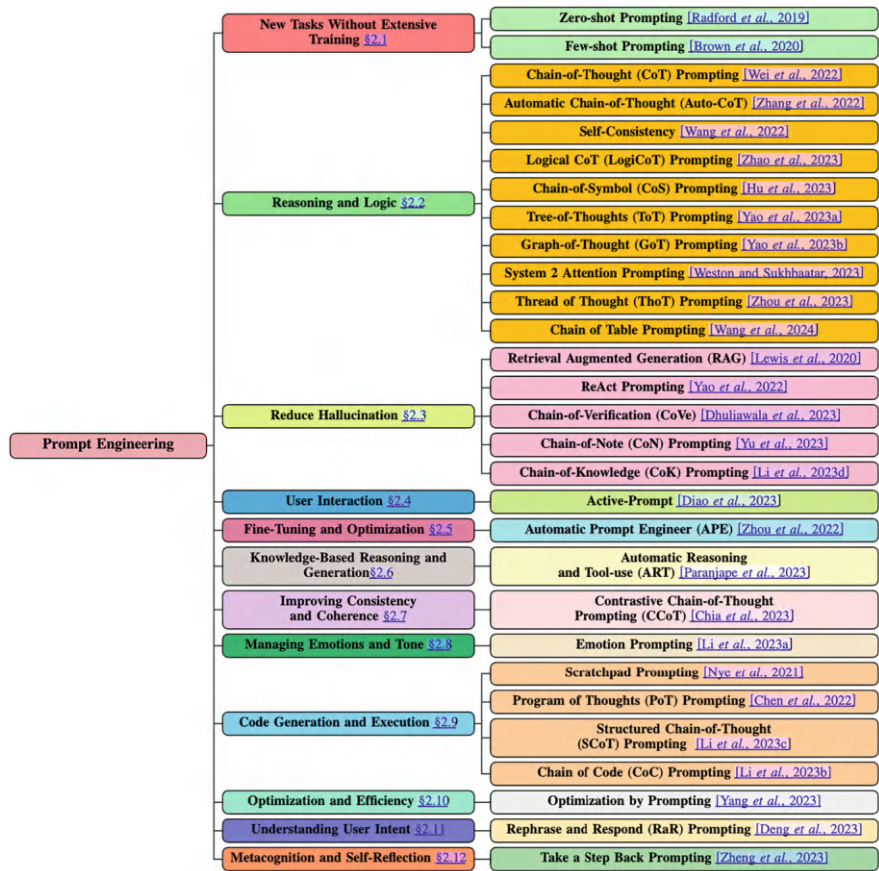


Fig. 3.8 Prompt engineering techniques [37]

several approaches to prompt engineering that are summarized in Fig. 3.8. The most relevant to us will be presented in detail in the recall of this section.

3.3.3.1 New Tasks Without Extensive Training

To present a new task to the LLM via prompting, we can leverage *in-context learning*. It is obtained by describing the new task directly in the prompt. Depending on whether some examples are presented or not, we have *zero-shot* or *few-shot prompting*.

Zero-Shot Prompting

Zero-shot prompting eliminates the need for extensive training data, instead relying on carefully designed prompts to guide the model in performing new tasks. The

model processes task descriptions within the prompt without using labeled data for training, leveraging its pre-existing knowledge to generate predictions. This approach significantly reduces the overhead of data preparation while enabling models to generalize effectively across various tasks.

Few-Shot Prompting

Few-shot prompting involves providing models with a limited number of input-output examples to help them understand a given task. Unlike zero-shot prompting, this technique offers concrete demonstrations, improving model performance on complex tasks. However, it requires additional tokens for examples, which may limit usability for longer inputs, and careful curation of examples is essential to mitigate biases and optimize results.

3.3.3.2 Reasoning and Logic

In the following paragraphs, we will present some techniques to enable some simulated reasoning and logical thinking by prompting.

Chain-of-Thought (CoT) Prompting

Chain-of-Thought (CoT) prompting enhances the reasoning capabilities of LLMs by guiding them to generate step-by-step solutions. This approach fosters structured and logical responses, especially for multi-step problems, by mimicking human problem-solving processes. CoT prompting has demonstrated state-of-the-art performance in reasoning benchmarks, such as math and commonsense tasks, showcasing its effectiveness for complex reasoning challenges.

Automatic Chain-of-Thought (Auto-CoT) Prompting

Auto-CoT addresses the labor-intensive nature of manual CoT example creation by automating the process. Using prompts like “Let’s think step by step,” the model generates diverse reasoning chains, enhancing robustness through sampling. This automated approach improves performance on reasoning tasks while reducing the need for manual effort, making it a practical enhancement over traditional CoT methods.

Self-Consistency

Self-consistency improves reasoning performance in CoT prompting by generating diverse reasoning chains and identifying the most consistent solution. This method acknowledges the diversity of valid reasoning paths for complex problems, marginalizing over sampled chains to determine the final answer. The combination of self-consistency and CoT prompting significantly boosts accuracy across benchmarks, making it an effective strategy for enhancing logical reasoning tasks.

3.3.4 LLM Agent

LLM agents are artificial entities capable of sensing their environment, making decisions, and taking actions. While significant progress has been made in advancing algorithms and training strategies to enhance specific capabilities, the community still lacks a general, adaptable model that can serve as a foundation for designing agents suitable for diverse scenarios.

LLMs, due to their broad capabilities, are increasingly seen as potential building blocks for artificial general intelligence (AGI). Many researchers have harnessed LLMs as a foundation for creating AI agents, achieving remarkable progress. A conceptual framework for LLM-based agents can be envisioned with three primary components: brain, perception, and action. This framework is flexible and can be tailored to different applications, enabling agents to adapt and excel in various environments.

In Fig. 3.9, it is presented the generic agent architecture that we will briefly discuss in the following paragraphs.

The brain module serves as the controller, handling tasks like memory, reasoning, and decision-making. The perception module processes multimodal information from the external environment, translating it into a format understandable by LLMs. The action module executes tasks using tools and interacts with the surroundings.

To illustrate how the whole agent workflow, consider an example where a human asks whether it will rain. The perception module interprets the query and transforms it into a representation the LLM can process. The brain module reasons using current weather data and forecasts, and the action module provides a response,

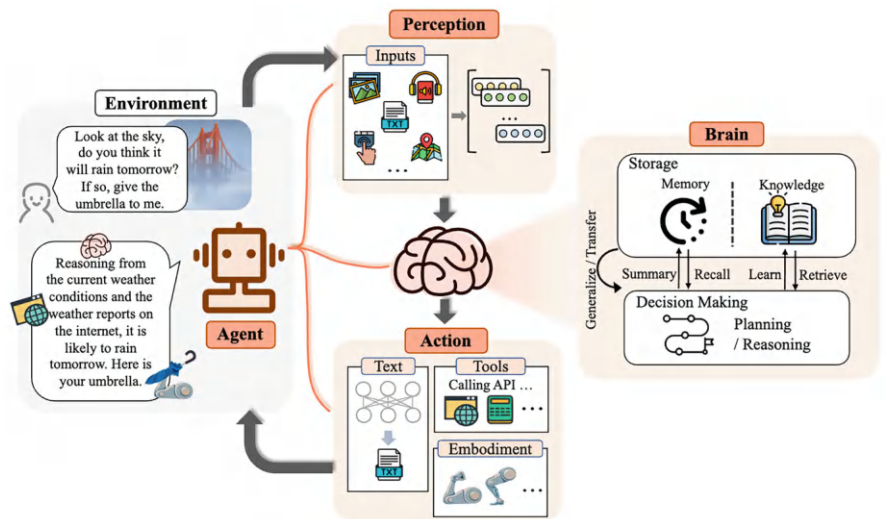


Fig. 3.9 LLM agent [46]

such as handing an umbrella to the human. Through continuous feedback and interaction, such agents can refine their functionality and adapt dynamically to their environment.

References

1. Aghajanyan, A., Zettlemoyer, L., Gupta, S.: Intrinsic dimensionality explains the effectiveness of language model fine-tuning. arXiv preprint arXiv:2012.13255 (2020)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization (2016). <https://arxiv.org/abs/1607.06450>
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR **abs/1409.0473** (2014). <https://api.semanticscholar.org/CorpusID:11212020>
4. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer (2020). <http://doi.org/10.48550/ARXIV.2004.05150>. <https://arxiv.org/abs/2004.05150>
5. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901. Curran Associates, Inc. (2020). https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf
6. Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H.P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., Zaremba, W.: Evaluating large language models trained on code (2021). <https://arxiv.org/abs/2107.03374>
7. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation (2014). <https://arxiv.org/abs/1406.1078>
8. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A.M., Pillai, T.S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., Fiedel, N.: Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.* **24**(1), 11324–11436 (2024)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: *North American Chapter of the Association for Computational Linguistics* (2019). <https://api.semanticscholar.org/CorpusID:52967399>
10. Du, N., Huang, Y., Dai, A.M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A.W., Firat, O., Zoph, B., Fedus, L., Bosma, M.P., Zhou, Z., Wang, T., Wang, E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q., Wu,

- Y., Chen, Z., Cui, C.: GLaM: Efficient scaling of language models with mixture-of-experts. In: K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, S. Sabato (eds.) *Proceedings of the 39th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 162, pp. 5547–5569. PMLR (2022). <https://proceedings.mlr.press/v162/du22c.html>
11. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**(null), 2121–2159 (2011)
 12. Gage, P.: A new algorithm for data compression. *C Users J.* **12**(2), 23–38 (1994)
 13. Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., Leahy, C.: The pile: An 800gb dataset of diverse text for language modeling (2020). <https://arxiv.org/abs/2101.00027>
 14. Han, Z., Gao, C., Liu, J., Zhang, J., Zhang, S.Q.: Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608* (2024)
 15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). <http://doi.org/10.1109/CVPR.2016.90>
 16. Hearst, M., Dumais, S., Osuna, E., Platt, J., Scholkopf, B.: Support vector machines. *IEEE Intelligent Systems and their Applications* **13**(4), 18–28 (1998). <http://doi.org/10.1109/5254.708428>
 17. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). *arXiv: Learning* (2016). <https://api.semanticscholar.org/CorpusID:125617073>
 18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Computation* **9**(8), 1735–1780 (1997). <http://doi.org/10.1162/neco.1997.9.8.1735>
 19. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021)
 20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *CoRR* **abs/1412.6980** (2014). <https://api.semanticscholar.org/CorpusID:6628106>
 21. Kudo, T.: Subword regularization: Improving neural network translation models with multiple subword candidates (2018). <https://arxiv.org/abs/1804.10959>
 22. La Rosa, B.: Explaining Deep Neural Networks by Leveraging Intrinsic Methods. *arXiv*. <http://doi.org/10.48550/ARXIV.2407.12243>. (2024)
 23. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **17**(39), 1–40 (2016). <http://jmlr.org/papers/v17/15-522.html>
 24. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Rahman Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: *Annual Meeting of the Association for Computational Linguistics* (2019). <https://api.semanticscholar.org/CorpusID:204960716>
 25. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **33**, 9459–9474 (2020)
 26. Lipinski, C.A., Lombardo, F., Dominy, B.W., Feeney, P.J.: Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Delivery Rev.* **64**, 4–17 (2012). <http://doi.org/10.1016/j.addr.2012.09.019>
 27. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.M.: Generating wikipedia by summarizing long sequences. *ArXiv* **abs/1801.10198** (2018). <https://api.semanticscholar.org/CorpusID:3608234>
 28. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. *ArXiv* **abs/1508.04025** (2015). <https://api.semanticscholar.org/CorpusID:1998416>
 29. Mnih, V., Kavcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). <https://api.semanticscholar.org/CorpusID:205242740>

30. Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., Xiong, C.: Codegen: An open large language model for code with multi-turn program synthesis. In: International Conference on Learning Representations (2022). <https://api.semanticscholar.org/CorpusID:252668917>
31. Ovadia, O., Brief, M., Mishaeli, M., Elisha, O.: Fine-tuning or retrieval? comparing knowledge injection in llms. arXiv preprint arXiv:2312.05934 (2023)
32. Parthasarathy, V.B., Zafar, A., Khan, A., Shahid, A.: The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. arXiv preprint arXiv:2408.13296 (2024)
33. Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training (2018). <https://api.semanticscholar.org/CorpusID:49313245>
34. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **21**(140), 1–67 (2020). <http://jmlr.org/papers/v21/20-074.html>
35. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 779–788. IEEE (2016). <http://doi.org/10.1109/cvpr.2016.91>
36. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**(6), 386–408 (1958). <https://api.semanticscholar.org/CorpusID:12781225>
37. Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A.: A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927 (2024)
38. Schuster, M., Nakajima, K.: Japanese and Korean voice search. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 5149–5152 (2012). <http://doi.org/10.1109/ICASSP.2012.6289079>
39. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units (2016). <https://arxiv.org/abs/1508.07909>
40. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. *Nature* **550**(7676), 354–359 (2017). <http://doi.org/10.1038/nature24270>
41. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (eds.) *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc. (2014). https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf
42. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2014). <http://doi.org/10.1109/cvpr.2014.220>
43. Tay, Y., Dehghani, M., Tran, V.Q., Garcia, X., Wei, J., Wang, X., Chung, H.W., Shakeri, S., Bahri, D., Schuster, T., Zheng, H.S., Zhou, D., Hounsby, N., Metzler, D.: U12: Unifying language learning paradigms (2023). <https://arxiv.org/abs/2205.05131>
44. Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., Stojnic, R.: Galactica: A large language model for science (2022). <https://arxiv.org/abs/2211.09085>
45. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. (2017). https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf
46. Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al.: The rise and potential of large language model based agents: A survey. arXiv preprint arXiv:2309.07864 (2023)

47. Xu, S., Cheng, Y., Gu, K., Yang, Y., Chang, S., Zhou, P.: Jointly attentive spatial-temporal pooling networks for video-based person re-identification. In: 2017 IEEE International Conference on Computer Vision (ICCV), p. 4743–4752. IEEE (2017). <http://doi.org/10.1109/iccv.2017.507>
48. Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., et al.: Instruction tuning for large language models: A survey. arXiv preprint arXiv:2308.10792 (2023)
49. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M.T., Li, X., Lin, X.V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P.S., Sridhar, A., Wang, T., Zettlemoyer, L.: Opt: Open pre-trained transformer language models. ArXiv **abs/2205.01068** (2022). <https://api.semanticscholar.org/CorpusID:248496292>

Chapter 4

A Reference Framework for Information Systems



Francesca De Luzi  and Massimo Mecella 

Abstract This chapter presents a comprehensive exploration of information systems (IS), analyzing their role, structure, and life cycle. First, we clarify key concepts such as data, information, and knowledge and explain their interconnections within organizations. The discussion thus defines IS from both a technological and an organizational point of view, underlining its importance in modern business environments. In this chapter, we describe the IS life cycle in detail, covering the planning, design, development, testing, and maintenance phases. We also introduce the information and knowledge management layer, highlighting data analysis and information extraction as critical components for turning raw data into useful information. Finally, the chapter discusses information systems engineering (ISE) as an interdisciplinary approach integrating technical and business considerations, addressing the challenges of big data, automation, and security. By examining methodologies, technologies, and best practices, this chapter provides a structured framework for understanding and optimizing IS in dynamic organizational contexts.

4.1 Introduction

In today's digital economy, businesses increasingly rely on information systems to drive efficiency, innovation, and competitiveness. Changes in strategy, rules, and business processes increasingly require updates to hardware, software, databases, communication networks, and user interfaces [22]. Information systems serve as a crucial link between business strategy and IT capabilities, enabling organizations to adapt to changing market conditions and operational demands. However, organization's capabilities are often limited by what its information system allows it to accomplish.

In this context, the Internet and the World Wide Web have become fundamental pillars upon which organizations build information systems to create

F. De Luzi (✉) · M. Mecella
Sapienza Università di Roma, Rome, Italy
e-mail: deluzi@diag.uniroma1.it; mecella@diag.uniroma1.it

an information-driven economy. Among the various approaches to address the challenges of a global economy is the need to acquire more information while also leveraging the value of existing data. Web-based platforms and applications are an effective response, as they allow the collection, correlated management, and distribution of large amounts of data from heterogeneous information sources. In such contexts, immediate and effective access to critical data extracted from vast datasets becomes both strategic and essential.

Information systems supported by information technology (IT), encompass software applications, data management systems, user interfaces, and communication networks. These systems support organizations in managing information and involve complexities in design, control, and management. These complexities stem from diverse information sources, databases differing in models and technologies, communication networks based on various protocols, and processes and corporate actors distributed and interconnected in cooperation.

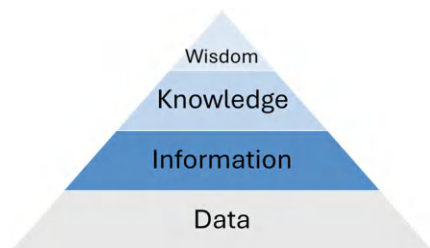
Therefore, design and implementing an information system is a complex discipline. On one hand, it must address future challenges such as the exponential growth of data (i.e., Big Data) and the increasing availability of low-cost, high-speed computational power (i.e., Cloud Computing). On the other hand, it must enable integration among heterogeneous system, and managing the resulting complexity is a crucial task. To address these challenges, various methodologies, techniques, and tools have been developed to support the design and implementation of information systems.

In the following sections, we will explore key concepts in detail to better understand how an information system works and the different perspectives from which it can be analyzed.

4.1.1 Data, Information, and Organization

Before defining an information system and its components, it is important to clarify key terms. The terms “data” and “information” are often used interchangeably. To illustrate their differences, we reference the so-called knowledge pyramid shown in Fig. 4.1, also known as DIKW (Data, Information, Knowledge, Wisdom) [13].

Fig. 4.1 The knowledge pyramid



At the base of the pyramid lies *data*, which represents facts or measurements—something considered true by definition and representing a portion of the reality to be depicted. Every piece of data is always associated with its type, which defines the range of possible values it can assume. Additionally, data may be characterized by its unit of measure. Given the vast amount of collected data, database management provides methods and tools to structure, store, and retrieve it efficiently.

Following the hierarchy suggested by the knowledge pyramid, above data lies *information*, which is constructed from data and can be defined as the interpretation of one or more data points. Indeed, a single data point, without any reference context, is often not useful. For data to become useful, it must be combined with other data to define a context that better characterizes the represented reality. Therefore, information can be seen as the output of queries applied to a set of data.

Ascending further up the pyramid, above information, we find *knowledge*, obtained by integrating information with experience. While information provides a comprehensive representation of reality, knowledge informs decision-making and influences outcomes. It is worth noting that knowledge is acquired over time through an individual's experience and generally cannot be transferred to others. In contrast, information—and, to a certain extent, experience—can be transferred, although formalizing the latter can be challenging.

Finally, we reach *wisdom*, which represents an extension of the traditional knowledge pyramid. Wisdom can be defined as the application of experience to knowledge, guiding an individual to take the most appropriate action in a given moment. This structure shows that at the bottom, data is highly numerous and fine-grained, while at the top, wisdom is represented by fewer elements, which are more synthetic and coarse-grained.

The use of information systems requires an understanding not only of data and information but also of the *business entity*, its management, and the role of IT within it. An information system creates value by providing organizational and managerial solutions to the challenges posed by the environment. Specifically, in the field of information systems, the key concepts are *resources* and *processes*. Therefore, the organization of a business entity defines the actions required to implement business processes, determines their distribution across different levels of the structure, and coordinates their execution to optimize resource utilization.

A resource refers to anything an organization operates with (both tangible and intangible) to pursue its objectives. Data and information are examples of resources and represent the fundamental elements of an information system. A process is the set of activities an organization carries out to manage the life cycle of a resource in order to achieve a defined and measurable result. Processes can be classified according to various classification models. Below we go into the hierarchical model of the organization given by Anthony's pyramid.

Fig. 4.2 Anthony pyramid

4.1.1.1 Anthony's Pyramid

Anthony's pyramid [3] is a foundational framework for understanding organizational processes and their relationship with decision-making levels. Robert Anthony's model categorizes organizational activities into three hierarchical levels, as shown in Fig. 4.2: *strategic planning*, *management control*, and *operational control*.

- The *strategic planning* level focuses on long-term goals and policies that shape the organization's direction. Decisions at this level—such as market expansion or launching significant initiatives—are based on aggregated data and high-level insights. The strategic layer considers both external and internal factors that influence organizational objectives.
- The *management control* level bridges strategic decisions with daily operations. It involves allocating resources, monitoring performance, and ensuring alignment with strategic goals. Examples include budget planning, resource scheduling, and performance evaluation, all crucial for maintaining efficiency and effectiveness.
- The *operational control* level handles day-to-day activities and tasks critical to the organization's functioning. It encompasses routine processes such as transaction recording, inventory management, and service delivery. The emphasis here is on precision and adherence to established procedures.

Anthony's pyramid illustrates the varying information needs and decision-making complexity at each level. Operational decisions are based on detailed, real-time data, while strategic decisions are based on synthesized insights and forecasts. Modern information systems—especially those that utilize business intelligence (BI) and data warehousing—support these diverse needs, ensuring seamless organizational function across all levels.

4.1.2 Information System

After clarifying what information is and how fundamental it is for an organization, we can accurately define an *information system*. An information system is, first and foremost, a *system*—an organized set of diverse elements that interact in a

coordinated manner. It highlights the fact that there is an organized set of elements of different natures, which interact in a coordinated manner.

Laudon and Laudon [17] define an information systems as a set of interrelated components that collect, process, store, and distribute information within an organization. Hirschheim et al. [14] emphasize the importance of the human factor in information systems and define them as:

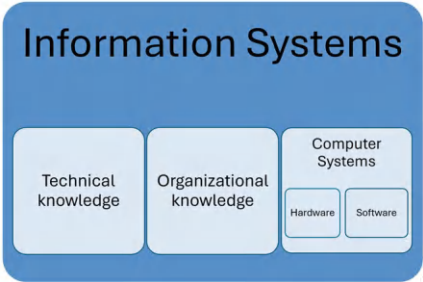
Traditionally, an information system has been defined in terms of two perspectives: one relating to its function; the other, to its structure. From a structural perspective, an information system consists of a collection of people, process, data, models, technology and partly formalized language, forming a cohesive structure, which serves some organizational purpose or function. From a functional perspective, an information system is a technologically implemented medium for the purpose of recorded, storing, and disseminating linguistic expressions as well as for the supporting of inference making. Through performing these elementary functions, IS facilitates the creation and the exchange of meanings that serve socially defined purposes such as control, sensemaking, and argumentation (i.e. the formulation and justification of claims). In either of these two perspectives on information systems, it should be noted that humans are included within its boundaries which means that the services provided by an IS in part depend upon human capabilities and contributions.

Information systems can be viewed from multiple perspectives, but two primary approaches emerge: a human- or information-centered perspective and a technology-centered perspective.

In general, information systems involve computerized as well as manual routines and include people involved as agents manipulating and conveying information that concern the things that are dealt with in the organization. The reduction in IT costs has widely expanded the possibility to automatically elaborate information, making procedures more accurate and faster while enhancing the ways information is processed. The term *automated information system* or *computer system* refers to that part of the information system implemented using IT: information is collected, processed, stored, and exchanged through the use of information and communication technologies.

There is a significant difference between an information system and a computer system: the former includes hardware, software, and the set technical and organizational knowledge of a business entity within which it operates, while the latter consists solely of hardware and software, as illustrated in Fig. 4.3.

Fig. 4.3 Representation of a business entity



4.1.2.1 Representation of an IS

An information system is a complex entity that cannot be fully captured by a single model and can be analyzed from multiple perspectives. The most widely used modeling in the literature involves three different levels of representation: *organizational model*, *functional model*, and *computer model*. In particular, the functional model will be examined.

In general, at the functional level, an information system performs three main activities [17]: input of information, i.e., a set of data entering the system, which can be collected in various ways (textual input, sensor input, etc.); data processing, where the data is aggregated, transformed, and stored to be output; and data output, where the data is distributed to system users, including individuals working within the organization, external individuals, or other information systems. The output generated by the information system indirectly produces *feedback*. Essentially, users will make decisions based on the output, and their actions will, in turn, generate additional input for the information system. Therefore, an information system not only monitors internal organizational activities but also interacts with and is influenced by external factors.

To be more specific, an information system is fed by *events*, represented as data, that occur within an organization as illustrated in Fig. 4.4. It transforms these data into *information*, which must be appropriately modeled and represented depending on who will use it. These events can trigger organizational processes, which form the foundation of an organization. Therefore, the key aspects of an information system revolve around managing the information resource according to *business rules* and objectives, utilizing technology for one or more organizations.

In line with these aspects, the design of information systems aims to understand and model, starting from organizational needs, how data and processes can be

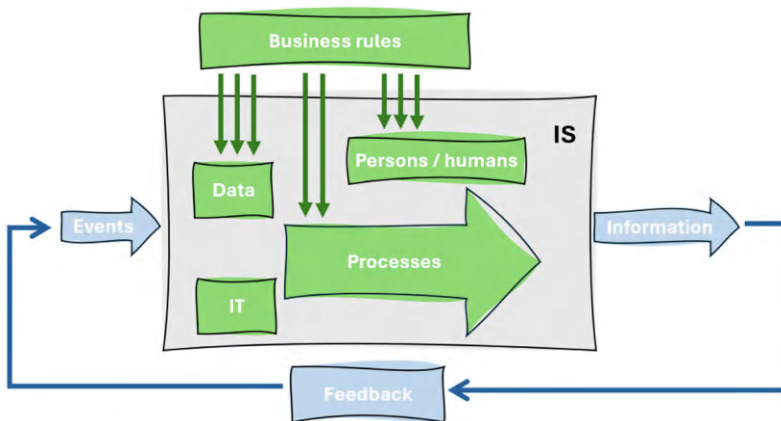


Fig. 4.4 Representation of an information system

supported by technology. The information system is often associated with the technological aspect alone, which is in fact a part, identified as *IT*, which processes, stores, and manages the exchange of information, and changes over time with the evolution of the business entity and technologies. Therefore, the key points of the information system become the management of the information resource according to rules and business objectives (business rules) with the use of technologies for one (or more) organizations.

The design and development of information systems involve multiple aspects, including data, processes, and user interaction. Structuring the project into key areas—data, business logic, and user interface—helps manage complexity effectively.

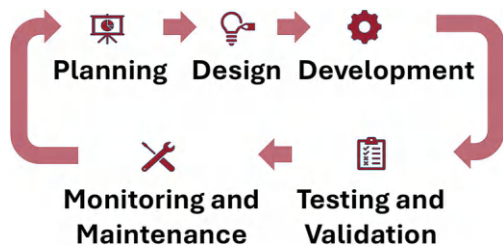
4.2 Information Systems Life Cycle

The life cycle of an information system is represented by its development plan, structured into various phases that form a complex process often spanning several years. Since the 1960s, researchers have investigated the ultimate methodology for developing information systems [20], and several research-based methodologies have been introduced [6]. The methodologies vary significantly, but in general, they give guidelines for how to control and coordinate the process of developing and implementing information systems. The life cycle, shown in Fig. 4.5, is iterative. The design process follows a planning and requirements analysis phase. After designing the system, we move on to its implementation and customer-side testing. The system is then deployed and monitored. New requirements may emerge during these phases, which is why the development of the system is iterative. In the following, we provide a description of each phase.

4.2.1 Planning

The first phase involves planning, during which strategic guidelines, organizational roles, and operational instructions for implementation are outlined. At this level,

Fig. 4.5 IS life cycle



once the objectives requiring IT solutions have been identified (strategic planning), the costs of various alternatives are assessed, and the priorities for implementing the different components of the information system are established (feasibility study).

Strategic planning helps assess the current state of the organization and identify areas and processes that require IT interventions, ensuring alignment between IT and business objectives. Key activities in strategic planning include (i) analyzing opportunities for information system development, including technologies, strategies, and processes that shape future system evolution, and (ii) assessing information needs, leading to the modeling and structuring information flows necessary for organizational operations.

The *feasibility study* leads to the selection of the most suitable solution by evaluating multiple viable options from both technical and organizational perspectives. Cost-effectiveness is also considered when comparing alternatives. Once a solution is chosen, system implementation begins.

During this phase, one or more system analysts collaborate with stakeholders to define the specific requirements of the new system. Procedures are documented, key users are interviewed, and data requirements are analyzed to gain a comprehensive understanding of the system's objectives. The outcome of this phase is a formal system requirements document.

4.2.2 Design

Once a solution is identified—whether it involves modifying the existing information system or implementing a new one—it requires a dedicated design phase. The feasibility study defines *what* the system needs to do, while the design phase focuses on *how* it will achieve this. There are three fundamental aspects to the design of an information system: data, which is the basis for conceptual modeling; process flows, which are the basis for process modeling; and user-system interactions, the basis for interaction modeling.

In the conceptual modeling phase, an abstract representation of the information system is defined (conceptual design), serving as a reference for the subsequent implementation phase. The objective of this phase is to identify types of information and classes and to describe how logical connections between them occur through relationships. Conceptual modeling is typically performed using a specialized modeling language that describes the analyzed reality and defines the system to be implemented. Some of the main languages used for the various design stages include:

- E/R Diagrams (Entity/Relationship): a graphical formalism used to document data structures based on the relational model.
- IDEF (Integration Definition) Languages: a family of graphical languages that emerged in the 1970s, with IDEF1X being the most well-known for data documentation.

- UML (Unified Modeling Language): a set of standard diagrams and graphical formalisms for describing various aspects of an information system. The functionalities are modelable in UML through *use case diagrams*, representing the functional requirements of a system from the user's point of view, identifying the actors involved in the system and the functions they are authorized to perform.
- BPMN (Business Process Model and Notation): a graphical language for describing and defining business processes.

The main output of the design phase provides a detailed description of system outputs, inputs, controls, and user interfaces. It also outlines hardware, software, databases, telecommunications, personnel, and procedures, illustrating how these components interrelate. A project specification document is then prepared, containing formal descriptions of the system and software architecture. Databases are designed based on specific models (schemas). The duration of the design phase varies depending on the solution: modifying an existing system may take several months, whereas designing a new system could take several years.

4.2.3 Development

In the development phase, the process of creating a system produces an operational solution through programming, that is, the creation of a set of logical instructions that a digital device must follow using a programming language. Developers write the code and build the app based on final requirements and specification documents. In particular, the development process consists of the actual implementation of the information system design in its various components through the acquisition of hardware and communication platforms and the development and installation of software applications. Developing good software is not easy; it requires a lot of resources and involves different groups of people in an organization. To do this effectively, the groups agree to follow a specific software development methodology.

Today, iterative and incremental software development methods, influenced by the principles outlined in the Agile Manifesto [4], referred to as *Agile* software development methods, are widely adopted due to their flexibility and adaptability. These approaches contrast with more traditional methodologies, such as the waterfall model [5], which emphasize extensive planning and structured phases. While traditional methods remain suitable for certain projects requiring strict documentation and predictability, Agile methodologies have gained prominence for their ability to accommodate changing requirements and foster continuous collaboration.

The design, build-up, and population of the database, which will be the essential part of the computer system, is of fundamental importance. The result of this phase is an initial working program that meets the requirements specified in the system planning phase and the design developed in the system design phase.

4.2.4 *Testing and Validation*

In the testing phase, the software program developed in the previous phase is put through a series of structured tests. The first is a unit test, which evaluates individual parts of the code for errors or bugs. This is followed by a system test in which the different components of the system are tested to ensure that they work together properly. Finally, the user acceptance test allows those that will be using the software to test the system to ensure that it meets their standards. Any bugs, errors, or problems found during testing are resolved, and then the software is tested again.

Typically, design and testing are typically conducted concurrently. For each change, testing must be done to ensure the system works as anticipated. There are different levels of testing that occur. IT staff will do the initial testing for bugs (coding errors) and glitches (sudden malfunction). Users may be asked to test for accuracy of the content, ease of use, or satisfaction with the feature, workflow, or other tasks.

4.2.5 *Monitoring and Maintenance*

After the implementation of a new system, it enters the monitoring and maintenance phase, where it remains in production and is actively used by the organization. Although no longer in development, continuous improvements and adjustments are necessary to fix bugs, accommodate new business requirements, and ensure long-term functionality. Maintenance includes [9] *corrective* maintenance (resolving errors not detected during implementation), *perfective* maintenance (adapting to changing user needs and regulations), and *adaptive* maintenance (aligning with technological advancements).

A crucial aspect of this phase is system monitoring, which ensures the system operates efficiently, securely, and in alignment with business needs. Monitoring involves tracking system performance, detecting anomalies, analyzing usage data, and identifying potential risks or failures before they impact operations. IT teams use automated tools and real-time analytics to oversee system health, optimize resources, and implement proactive measures to prevent downtime.

Regular system updates, backups, and structured support processes help maintain efficiency and reliability. Additionally, IT management must ensure that the system continues to align with business priorities while optimizing operational costs. Over time, significant upgrades or even a complete system overhaul may be required, particularly when integrating with newer technologies. A well-maintained and well-monitored system remains scalable, secure, and capable of supporting evolving business needs.

4.3 The Information and Knowledge Management Layer

As discussed in the introductory section, information systems are complex systems through which an organization can achieve its goals and improve its products or services, making them competitive in the market. However, an information system is not isolated from what happens outside the organization, and the way it is designed, as well as its capabilities, is influenced by the technologies used to develop it. The emergence of new technologies and their adoption within an organization's information system bring numerous advantages and can significantly impact the company's objectives.

In this modern data-driven environment, effective management of information and knowledge becomes essential. The relationship between Knowledge Management (KM) and Information Management (IM) plays a crucial role in this landscape: while IM focuses on ensuring the efficient collection, processing, storage, and distribution of information, KM leverages this structured information to generate insights and knowledge. A multi-layered approach, such as the one depicted in Fig. 4.6, illustrates this relationship effectively. The key components are described below.

- Data sources [12] represent different types of data according to a classification proposed by UNECE [26] (United Nations Economic Commission for Europe). There are three main types of data sources that can be viewed as Big Data: human sourced (e.g., blog comments), process mediated (e.g., banking records), and machine generated (e.g., sensor measurements).
- Transactional databases form the foundation by storing raw operational data. These interactions, or transactions, are named as such due to the type of data they host (e.g., data derived from an interaction between two or more parties) rather than the type of database they are. There are various types of databases that can function as transactional databases, including relational databases and NoSQL databases.
- ETL [10] (extraction, transformation, loading) processes ensure data quality and standardization before integration into structured repositories. In particular, this ETL process helps the data warehouse achieve enhanced system performance and business intelligence by making data more reliable, accurate, and easily accessible. This enables timely data access and a higher return on investment, as it allows for more efficient analysis and reporting.
 - Extraction: involves connecting systems and collecting the data needed for analytical processing.
 - Transformation: extracts data and converts it into a standard format.
 - Loading: transformed data are imported into a large data warehouse.
- Data warehouses [16] and data lakes [11] act as centralized storage solutions, with the former providing structured data for analytical purposes and the latter offering flexible storage for all types of data (structured, semi-structured, and unstructured data). Data lakehouse [21] combines the best elements of data lakes

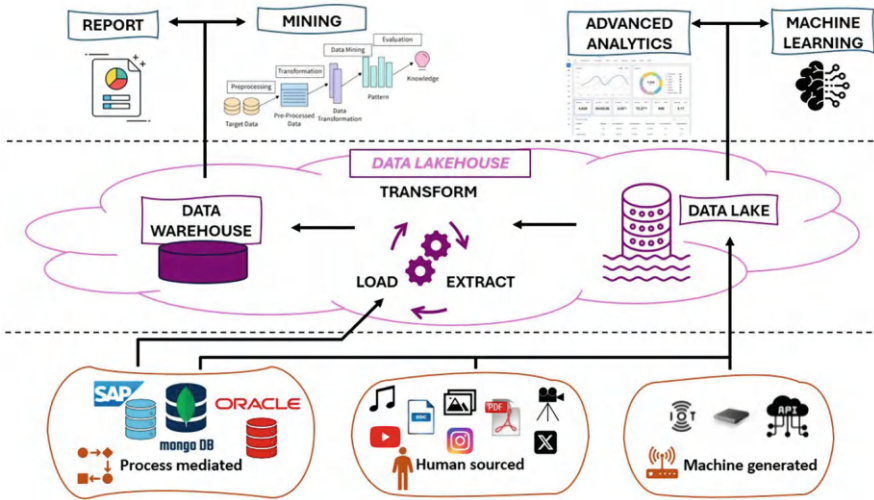


Fig. 4.6 The information and knowledge management layer

and data warehouses. It is a novel architecture based on low-cost and directly accessible storage that provides traditional analytical DBMS management and performance features. Data can be stored in a single location and supports ML, BI, and data streaming. In data lakehouses, data warehouse-like structures and schemas can be used for unstructured data like in a data lake.

- Analytics and dashboarding represent the final stage, where processed data is transformed into meaningful insights, supporting decision-making and strategic planning (c.f., Sect. 4.3.1).

By integrating KM and IM principles, organizations bridge the gap between raw data and actionable knowledge, facilitating a seamless transition from data management to strategic insight generation. Therefore, once data has been collected, processed, and stored through the aforementioned layers, the next crucial step is to extract valuable insights and transform raw data into actionable knowledge. This phase is essential for organizations aiming to leverage their information assets for strategic decision-making.

In the current landscape, two processes play a key role: *data analytics*, which focuses on analyzing and visualizing data to support business intelligence, and *information extraction*, which involves retrieving relevant knowledge from structured and unstructured data sources. The following sections provide an in-depth discussion of these two critical components.

4.3.1 Data Analytics

Data analytics serves as a critical component within the information and knowledge management layer, enabling organizations to process, analyze, and interpret large-scale datasets. In recent years, advancements in computational power and algorithmic efficiency have facilitated the application of sophisticated analytical techniques to extract valuable insights from data.

There are four essential types of analysis [24]: exploratory, descriptive, sentiment, and predictive. *Exploratory analysis* investigates data for patterns, relationships, and anomalies without making initial assumptions. It helps in understanding the structure of the data and generating hypotheses for further examination. *Descriptive analysis* summarizes historical data to provide insights into past trends and performance. For example, it transforms raw data into reports that monitor an organization's operations in real time. *Sentiment analysis* examines subjective data, measuring opinions or emotions expressed in textual content. It is often used to understand customer feedback. *Predictive analytics* leverages historical data and statistical models to forecast future outcomes. Therefore, mining and extracting meaningful patterns from massive input data for decision-making, prediction, and other inferencing is at the core of Big Data analytics [19].

Moreover, in the context of KM, data analytics plays a pivotal role in transforming unstructured textual data into structured knowledge [15]. By leveraging techniques such as sentiment analysis, topic modeling, and entity recognition, organizations can gain deeper insights into textual information, enabling better information utilization. The application of real-time analytics also enhances responsiveness, allowing organizations to adapt swiftly to emerging trends and dynamic market conditions.

4.3.2 Information Extraction

Information extraction (IE) refers to the automatic extraction of structured information, such as entities, their attributes, relationships, and events, from unstructured text [2, 18, 23]. It is a fundamental task in natural language understanding and plays a crucial role in transforming raw text into structured knowledge.

IE encompasses a wide range of downstream tasks, such as named entity recognition (NER) [30], which identifies and classifies entities; entity typing (ET) [8], which assigns specific categories to entities; relation extraction (RE) and event extraction (EE) [27], which respectively determine the connections between entities and identify events and their arguments; and knowledge graph construction [29]. In addition, Aspect-Based Sentiment Analysis (ABSA) [7] is often considered part of IE, as it extracts opinions and sentiments related to specific aspects of entities.

Given its role in structuring raw data, IE is deeply connected to KM. By efficiently extracting and organizing knowledge, IE systems help enhance the

efficiency and effectiveness of data analysis, particularly in context-aware analytical systems [1]. This improves data-driven decision-making and knowledge management, ultimately facilitating the development of intelligent systems.

4.4 Information Systems Engineering

In the introduction, we clarified key terms to understand the concept of information systems. In particular, we have come to define ISs by distinguishing the concepts of data, information, knowledge, and wisdom. To complete our discussion and define information systems engineering (ISE), we still need to analyze the concept of engineering, both in absolute terms and in relation to a system. The approach taken to define ISE has followed the perspective outlined in [28], giving the fundamental terms that constitute ISE and finally its definition.

The field of *engineering* has been defined by the Engineers Council for Professional Development as the creative application of *scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or work utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behaviour under specific operating conditions; all as respects and intended function, economics of operation and safety to life and property.*

Therefore, engineering involves systematic analysis, problem-solving, and innovation to create functional and optimized solutions across various disciplines, from mechanical and civil engineering to software and information systems engineering. Notably, in the context of ISs, this definition highlights that engineering encompasses both the creative application of manufacturing processes and the operation of systems [28].

Systems engineering [25] is an interdisciplinary approach that focuses on designing, integrating, and managing complex systems over their life cycles. It ensures that all components—technical, operational, and human—function together cohesively to meet user needs and system requirements. Systems engineering emphasizes a holistic view, addressing aspects such as performance, reliability, cost, scalability, and sustainability while considering the interactions between system elements and their environment. However, there is much more to systems engineering than this definition alone suggests. Its scope is broad and can be described from a variety of perspectives, including participants, total life cycle impact, and the roles played by both industry and researchers.

As a result, the field of *information systems engineering* (ISE) should focus not only on implementation technologies and development methodologies but also on the engineering process itself and the study of system usage and maintenance in a broader sense. Both engineering and systems engineering are at the core of ISE, an interdisciplinary approach that integrates technical and business considerations throughout the life cycle of an information system. ISE applies formal analytical

methods to design, develop, and maintain complex systems that efficiently process, analyze, and protect large amounts of data.

In particular, with the advent of the widely discussed Big Data and the need for fast and reliable decisions, modern ISs must support operational and informational functions while facing challenges such as scalability, automation, and security. In such a context, advanced methodologies are emerging, such as agile software development, cognitive engineering, and automated decision-making systems, which promise to improve system adaptability, user experience, and resilience. In addition, the integration of user-centered and multimedia design principles ensures that ISs are not only functionally robust but also intuitive and engaging, facilitating seamless interaction between man and computer. By managing complexity and promoting innovation, ISE enables the development of high-performance systems that align with technological and organizational needs in an increasingly fast-evolving landscape.

In this chapter, we have detailed each phase of the IS life cycle to establish a sound conceptual basis. By defining ISE and its interdisciplinary nature, we provide the necessary context for understanding the methodologies, challenges, and principles that guide the development and management of ISs.

This basic knowledge ensures that the subsequent investigation can be approached with a clear understanding of the engineering processes and systemic considerations that influence each stage of the life cycle.

References

1. Adnan, K., Akbar, R.: Limitations of information extraction methods and techniques for heterogeneous unstructured big data. *Int. J. Eng. Bus. Manag.* **11**, 1847979019890771 (2019)
2. Andersen, P.M., Hayes, P.J., Weinstein, S.P., Huettner, A.K., Schmandt, L.M., Nirenburg, I.: Automatic extraction of facts from press releases to generate news stories. In: *Third Conference on Applied Natural Language Processing*, pp. 170–177 (1992)
3. Anthony, R., Planning, N.: *Control Systems: A Framework for Analysis*. Graduate School of Business Administration, Division of Research, Harvard University, Boston (1965)
4. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al.: *Manifesto for agile software development* (2001)
5. Benington, H.D.: Production of large computer programs. *Ann. Hist. Comput.* **5**(4), 350–361 (1983)
6. Bubenko, J.: On the evolution of information systems modelling: A Scandinavian perspective. *DSV* (1992)
7. Chen, Z., Qian, T.: Relation-aware collaborative learning for unified aspect-based sentiment analysis. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3685–3694 (2020)
8. Choi, E., Levy, O., Choi, Y., Zettlemoyer, L.: Ultra-fine entity typing. *arXiv preprint arXiv:1807.04905* (2018)
9. Edwards, C.: Information systems maintenance: an integrated perspective. *Mis. Quart.*, 237–256 (1984)
10. Esmail, F.S.: A survey of real-time data warehouse and etl. *Management* **9**(3), 3–9 (2014)

11. Fang, H.: Managing data lakes in big data era: What's a data lake and why has it become popular in data management ecosystem. In: 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), pp. 820–824 (2015). <http://doi.org/10.1109/CYBER.2015.7288049>
12. Firmani, D., Mecella, M., Scannapieco, M., Batini, C.: On the meaningfulness of “big data quality”. *Data Sci. Eng.* **1**, 6–20 (2016)
13. Frické, M.: The knowledge pyramid: a critique of the DIKW hierarchy. *J. Inf. Sci.* **35**(2), 131–142 (2009)
14. Hirschheim, R., Klein, H.K., Lyytinen, K.: Information systems development and data modeling: conceptual and philosophical foundations, vol. 9. Cambridge University Press, Cambridge (1995)
15. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. *J. Parallel Distrib. Comput.* **74**(7), 2561–2573 (2014)
16. Kimball, R., Ross, M.: The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley, New York (2013)
17. Laudon, K.C., Laudon, J.P.: Management Information Systems: New Approaches to Organization and Technology. Prentice Hall PTR, Hoboken (1997)
18. Lu, Y., Liu, Q., Dai, D., Xiao, X., Lin, H., Han, X., Sun, L., Wu, H.: Unified structure generation for universal information extraction. arXiv preprint arXiv:2203.12277 (2022)
19. Najafabadi, M.M., Villanustre, F., Khoshgoftaar, T.M., Seliya, N., Wald, R., Muharemagic, E.: Deep learning applications and challenges in big data analytics. *J. Big Data* **2**, 1–21 (2015)
20. Nilsson, A.G.: Evolution of methodologies for information systems work : A historical perspective (1995). <https://api.semanticscholar.org/CorpusID:61002903>
21. Oreščanin, D., Hlupić, T.: Data lakehouse-a novel step in analytics architecture. In: 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO), pp. 1242–1246. IEEE (2021)
22. Pernici, B., Cappiello, C., Fugini, M., Grefen, P., Plebani, P., Vitali, M.: Fondamenti di Sistemi informativi: per il Settore dell'Informazione. In: Politecnico di Milano (2025)
23. Sarawagi, S., et al.: Information extraction. *Found. Trends® Databases* **1**(3), 261–377 (2008)
24. Siddiqui, M.R.: Empowering decision-making through big data analytics insights from exploratory, descriptive, sentiment, and predictive frameworks. *Authorea Preprints* (2025)
25. Stevens, R.: Systems Engineering: Coping with Complexity. Pearson Education, London (1998)
26. UNECE: Classification of types of big data (2015). <https://unstats.un.org/unsd/classifications/expertgroup/egm2015/ac289-26.PDF>. Accessed: Jan 2025
27. Wang, X., Zhou, W., Zu, C., Xia, H., Chen, T., Zhang, Y., Zheng, R., Ye, J., Zhang, Q., Gui, T., et al.: Instructuie: Multi-task instruction tuning for unified information extraction. arXiv preprint arXiv:2304.08085 (2023)
28. Wangler, B., Backlund, A.: Information systems engineering: What is it? In: CAiSE Workshops (2), pp. 427–437 (2005)
29. Zhong, L., Wu, J., Li, Q., Peng, H., Wu, X.: A comprehensive survey on automatic knowledge graph construction. *ACM Comput. Surv.* **56**(4), 1–62 (2023)
30. Žukov-Gregorič, A., Bachrach, Y., Coope, S.: Named entity recognition with parallel recurrent neural networks. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 69–74 (2018)

Chapter 5

Exploring Large Language Models in Information Systems: A Survey



Filippo Bianchini  and Matteo Marinacci 

Abstract The integration of large language models (LLMs) into the life cycle of information systems (ISs) is transforming traditional methodologies across multiple phases, from planning and design to monitoring and maintenance. This chapter provides a comprehensive survey of the most impactful research on LLMs in key IS life cycle stages, analyzing their applications, benefits, and limitations. Through a literature review, we examine how LLMs are utilized and can be leveraged across different phases of the IS life cycle: planning, design, development, testing and validation, and monitoring and maintenance. Additionally, we analyze their role in tasks that, while not strictly part of the IS life cycle, are increasingly relevant, such as information extraction and data analytics. This chapter provides an in-depth analysis of recent LLMs applications in each phase, evaluating their impact on traditional methodologies and assessing their effectiveness in automating processes, improving decision-making, and enhancing overall system efficiency.

5.1 Introduction

This chapter serves as a comprehensive survey of the most impactful studies on the application of large language models (LLMs) within the realm of information systems (ISs), specifically on how LLMs are leveraged to address challenges across the entire life cycle of an IS, highlighting state-of-the-art methodologies, innovations, and practical applications. The life cycle of an IS¹ encompasses several phases: planning, design, development, testing and validation, and monitoring and maintenance. For each of these phases, this survey identifies and evaluates

¹ For a detailed overview of the phases of an IS life cycle, see Chap. 4.

F. Bianchini (✉) · M. Marinacci
Sapienza Università di Roma, Rome, Italy
e-mail: bianchini@diag.uniroma1.it; marinacci@diag.uniroma1.it

Table 5.1 Survey summary of IS life cycle phases

| Phase | Studies | Description |
|----------------------------|---------|---|
| Planning | 4 | Focuses on strategic alignment, requirements elicitation, and techniques such as SWOT analysis to define objectives and align IS goals with organizational needs |
| Design | 5 | Involves creating abstract representations of systems, such as UML diagrams, Entity-Relationship diagrams, and process flows. Topics include quantitative and qualitative evaluation of LLM performance for accuracy, scalability, and adherence to constraints |
| Development | 4 | Defines the technical framework of the system, including components, interactions, and platforms. Key topics include scalability, security, and technological standardization |
| Testing and Validation | 6 | Ensures the system meets functional and non-functional requirements. Topics include iterative evaluation, correctness verification, and performance optimization |
| Monitoring and Maintenance | 7 | Focuses on continuous oversight of system performance, security, and compliance. Includes techniques such as log analysis, anomaly detection, and performance tracking |
| Other areas | | |
| Data Analytics | 6 | Leverages LLMs for predictive modeling, trend analysis, and multi-source data integration to extract actionable insights |
| Information Extraction | 6 | Focuses on deriving structured information from unstructured data sources, including text and multimedia, to enrich the system’s knowledge base |

significant research contributions, offering a detailed view of how LLMs are reshaping IS workflows.

To identify relevant studies, this survey conducts a literature review using advanced academic article retrieval tools such as Publish or Perish² and by referring to the main guidelines proposed in the literature [33, 66]. Queries were designed to target highly specific intersections of IS and LLM applications. In cases where the queries yielded zero or very few results—reflecting the scarcity of direct applications of LLMs in certain IS life cycle phases—the selected studies focused on frameworks, techniques, or methodologies relevant to the respective phase, where LLMs could plausibly be utilized. This approach ensured comprehensive coverage of the literature, capturing both foundational studies and innovative applications.

² <https://harzing.com/resources/publish-or-perish>

Each study underwent a detailed analysis, assessing both its specific contributions and its placement within the broader state-of-the-art landscape. Particular attention was given to the LLM models employed during the testing and validation phases, as these models are critical to understanding the practical implications and limitations of the research. Evaluation methodologies were also analyzed, highlighting how researchers assessed the performance and effectiveness of their proposed approaches. This dual focus on LLM models and evaluation methodologies ensures that the survey not only captures theoretical advancements but also provides insights into their practical applicability.

5.2 Results of the Literature Review

The life cycle of an IS encompasses multiple phases, each addressing specific objectives and challenges in the development and management of the system.

- Planning phase: focuses on defining strategic objectives and gathering requirements to ensure alignment with organizational goals and stakeholder expectations (Table 5.2).
- Design phase: an abstract representation of the system is then provided through tools such as Entity-Relationship diagrams, UML models, or process flows, which serve as blueprints for structuring and refining system requirements (Table 5.3).
- Development phase: establishes the technical framework of the system, including the definition of components, their interactions, and the underlying technological platforms. This phase ensures the scalability, security, and standardization of the system (Table 5.4).
- Testing and validation phase: verifies that the system meets its functional and non-functional requirements, employing rigorous evaluation to ensure reliability and correctness (Table 5.5).
- Monitoring and maintenance phase: addresses the continuous oversight of the system's performance, security, and compliance through techniques such as log analysis and anomaly detection (Table 5.6).

Although the deployment phase is not addressed separately, it is considered an integral part of the transition process between development, testing, and maintenance. Deployment activities, including system configuration and release, are closely linked to the testing and validation phases, as well as monitoring and maintenance operations, to ensure the system's stability and effectiveness in production. Any studies addressing deployment-related topics will be discussed within the relevant phases of the IS life cycle, depending on the specific context in which they are analyzed. In addition to these life cycle phases, LLMs also play a significant role in areas closely connected to information systems, such as data analytics and information extraction (c.f. Sect. 5.2.6).

- Data analytics helps in extracting actionable insights from the data generated by the system, enabling predictive modeling and trend analysis (Table 5.7).
- Information extraction focuses on structuring unstructured data from diverse sources, such as text or multimedia, to enrich the system’s knowledge base and support decision-making processes (Table 5.8).

For each of these phases, a summary table is provided to present key findings, including the LLMs used for testing and validation, as well as the evaluation methodologies applied. In Table 5.1, the analyzed phases are reported along with number of selected papers and a brief description for each phase.

Table 5.2 Planning for information systems

| Source | LLM | Evaluation methodology |
|---------------------|--|--|
| Ataei et al. [6] | GPT-4-turbo text-embedding-ada-002 | User agent diversity, evaluated using computational metrics and clustering, showing serial method as most diverse; <i>Latent Need Identification</i> , simulated user scenarios with LLM agents, assessed with F1-scores and predefined criteria; <i>Latent Need Detection</i> , different detection methods were compared, including zero-shot and criteria-based approaches (evaluated on precision, recall, and F1 score) |
| Ronanki et al. [53] | GPT-3.5 | Expert-based comparative evaluation, attribute scoring methodology (scale on 0 to 10 for selected quality attributes), controlled context testing using standardized questions on Trustworthy AI principles |
| Kenyon et al. [46] | all-MiniLM-L12-v2 all-MiniLM-L6-v2 all-mpnet-base-v2 msmarco-bert-base-dot-v5 msmarco-MiniLM-L6-cos-v5 multi-qa-MiniLM-L6-co-v1 multi-qa-mpnet-base-dot-v1 | Semantic validation, semantic similarity analysis using cosine distance between text embeddings, accuracy analysis (spot checking with <i>Damerau-Levenshtein</i> distance metric), quantitative evaluation, out-of-sample return analysis to evaluate the effectiveness of the SWOT-based portfolios against traditional methods |
| Arora et al. [4] | GPT-3.5 | Comparative analysis, customized metric evaluation (full match, partial match, or no match), automatic metric evaluation (precision and recall), human evaluation |

Table 5.3 Design for information systems

| Source | LLM | Evaluation methodology |
|------------------------|--|---|
| Klievtsova et al. [34] | GPT-1 (text-davinci-001), GPT-2 (text-davinci-002), GPT-3 (text-davinci-003), GPT-3.5 (gpt-3.5-turbo), GPT-4 | Quantitative Evaluation (Jaccard Index, Precision and Recall, Key Performance Indicators (KPIs)), Qualitative Evaluation (human experts), surveys |
| Kourani et al. [35] | GPT-4, Gemini | LLMs effectiveness, comparison with existing frameworks (e.g.,) |
| Cámara et al. [16] | GPT (OpenAI, February 2023 release) | Syntactic and semantic accuracy, scalability, concepts modeling, prompt sensitivity |
| Fill et al. [22] | GPT-3, GPT-4 | correctness, completeness, adherence to constraints |
| Conrardy et al. [15] | GPT-4V, Gemini Pro, Gemini Ultra, CogVLM | Accuracy, Completeness, Syntax Errors, Impact of Complexity, Effect of Prompt Detail, Semantic Understanding |

Table 5.4 Development for information systems

| Source | LLM | Evaluation Methodology |
|--------------------|--|---|
| Rahman et al. [52] | GPT-4.0 | Effectiveness tested on seven RE documents, user evaluation based on RUST framework, quantitative and qualitative evaluation |
| Dong et al. [20] | GPT-3.5 | Comparison against other models on different benchmarks using two experimental settings (i.e., one combining NL descriptions, function signatures, and public test cases and another relying only on NL descriptions) using Pass@1 metric |
| Jiang et al. [28] | code-davinci-002 | Comparison against other models on different benchmarks using different metrics (i.e., Pass@1, Pass@k, AvgPassRatio, CodeBLEU), user evaluation |
| Li et al. [41] | gpt-3.5-turbo-0301 code-davinci-002 | Comparison against other standard prompting methods on different benchmarks using Pass@k metric, user evaluation, ablation studies |

Table 5.5 Testing and validation for information systems

| Source | LLM | Evaluation methodology |
|---------------------|--|---|
| Liu et al. [42] | GPT-3 | Comparison with existing state-of-the-art approaches, user evaluation, usefulness |
| Feng et al. [21] | GPT-3.5 | Comparison against existing approaches (e.g., and ablation models) on performances reproducibility and computational time, user evaluation |
| Chen et al. [14] | Codex (code-davinci-002) GPT-3.5-turbo GPT-4 StarCoder | Evaluation against benchmark datasets, comparison with existing approaches (e.g., Coder-Reviewer) |
| Wei et al. [67] | CodeT5-large InCoder-6.7B | Comparison against existing traditional NMT-based and LLM-based approaches, bug-fixing capabilities, compilation rate, generalizability |
| Schäfer et al. [58] | GPT-3.5-turbo Code-Cushman-002 StarCoder | Comparison against employed LLMs and state-of-the-art methods, quality of generated tests, coverage, similarity with already existing results |
| Joshi et al. [30] | Codex | Comparison against other repair engines, repair accuracy, multilingual capabilities, error localization |

Table 5.6 Monitoring and maintenance for information systems

| Source | LLM | Evaluation methodology |
|------------------|---|--|
| Gu et al. [29] | gpt-3.5-turbo | Evaluation against benchmark datasets, template accuracy (F1 score), performance comparison with state-of-the-art parsers, efficiency assessment |
| Xu et al. [72] | Codex, GPT-3 | Comparison against state-of-the-art accuracy benchmarks in all logging tasks, evaluated on parameter tuning time (UniLog required < 4% of the time compared to SOTA methods) |
| Ma et al. [44] | Flan-T5-small Flan-T5-base LLaMA-7B ChatGLM-6B | Evaluations on a benchmark dataset of logs, parsing accuracy (PA), Grouping accuracy (GA), analysis of training data size effects, pre-training effect analysis, efficiency testing |
| Jia et al. [25] | GPT-3.5 BERT | Precision, recall, and F1-score, Comparison across multiple datasets (HDFS, BGL, and Thunderbird), few-shot learning evaluation with varying proportions of labeled training data (5%, 10%, 20%, and 30%), generalization scenario testing, evaluation against state-of-the-art unsupervised, semi-supervised, and supervised models |
| Ali et al. [2] | gpt-3.5-turbo | Comparison against traditional machine learning models, analysis of privacy and computational efficiency challenges, assessment of known attack type identification accuracy, evaluation of effectiveness against zero-day attacks, investigation of LLM adaptability and generalization capabilities |
| Wu et al. [70] | GPT-4 Llama | Simulating scenario to evaluate the tool's reasoning capabilities, use of metrics such as precision, recall, and F1-score, to quantify the accuracy while metrics as time to detect vulnerabilities and time to execute countermeasures to perform benchmark tests, real-world attack simulation |
| Deng et al. [18] | GPT-4 GPT-3.5 BARD | Automated reasoning tests, generation and parsing accuracy (syntactic correctness, logical coherence), context retention evaluation, performance benchmarking (<i>time-to-detection</i> and <i>time-to-mitigation</i>), real-world attack simulation |

Table 5.7 Data analytics for information systems

| Source | LLM | Evaluation Methodology |
|--------------------|--|---|
| Wang et al. [65] | PaLM 2 GPT-3.5 | Benchmark evaluation across several datasets (<i>WikiTQ</i> for table QA, <i>TabFact</i> for table fact verification, and <i>FeTaQA</i> for free-form-table-based QA), comparison existing reasoning frameworks (chain-of-thought and text-to-SQL), evaluation of tabular operations (sorting, filtering) against gold standard |
| Nguyen et al. [11] | GPT-3.5 GPT-4 | Accuracy and interpretability evaluation with datasets (TabFact, WikiTQ, and FeTaQA), comparison with other frameworks (text-to-SQL, Chain-of-Table, and DATER) for both accuracy and user interpretability, human evaluation coherence and clarity, efficiency analysis by several metrics (query reduction rate and computational cost savings) |
| Parisi et al. [49] | T5 BM25-Based | Knowledge-oriented QA evaluation (F1 scores), reasoning-oriented evaluation (Math QA), iterative self-play validation, comparison against human-curated answers and baseline LLM predictions |
| Liu et al. [26] | GPT-4o GPT-3.5-Turbo-0613 GPT-4-0613 | Benchmark evaluation (InfiAgent-DABench, ML-Benchmark, MATH Dataset), metrics evaluation (accuracy; Normalized Performance Score, NPS; and Comprehensive Score, CS), comparative baselines against other framework (AutoGen, OpenInterpreter, and OpenDevin) |
| Qiao et al. [51] | GPT-4 GPT-3.5 | Benchmark evaluation (DS-100, InfiAgent-DABench, DSEval), metrics evaluation (overall task success rates, pass rate against ground-truth code), comparison with existing framework (LangChain, AutoGPT) through accuracy, efficiency, and stateful execution |
| Wang et al. [69] | GPT-4 GPT-3.5-Turbo | Benchmark evaluation (MATH, AFLWord, Natural Questions), metrics evaluation (success ratio, F1 score, recall), baselines comparison against AutoGPT, ChatGPT, Langchain ReAct, Multi-Agent Debate |

Table 5.8 Information extraction for information systems

| Source | LLM | Evaluation methodology |
|-------------------|---|---|
| Zhang et al. [78] | GPT-3 GPT-4 BERT | Comparison against baseline methods |
| Li et al. [38] | GPT-3.5-turbo GPT-4 T5-XXL | Comparison against state-of-the-art DocRE models and under different configurations (i.e., zero-shot document-level RE, training with the augmented dataset) considering metrics like precision, recall, F1 score |
| Guo et al. [24] | OPT (Open Pre-trained Transformer) BLOOM GPT-J GPT-NEO | Comparison against VQA methods and different LLMs, generalization and ablation studies |
| Li et al. [40] | OPT Flan-T5 CLIP EVA-CLIP Florence Flamingo | Comparison against state-of-the-art models on different visual-language tasks |
| Liu et al. [43] | LLaMA 2 MU-LLaMA MERT AudioLDM 2 MusicGen ViT (Vision Transformer) ViViT (Video Vision Transformer) | Comparison of music generation and understanding tasks against state-of-the-art models on different benchmarks, user evaluation on music generation task |
| Zhang et al. [77] | LLaMA/Vicuna ViT-G/14 ImageBind | Comparison against existing models, empirical evaluation |

5.2.1 Planning

The requirements elicitation phase is a critical component of the planning stage in the life cycle of an information system. This phase lays the foundation for the entire development process by identifying user needs, system constraints, and functional requirements [80]. Effective requirements elicitation ensures that the resulting system aligns with organizational goals and user expectations while mitigating potential risks related to misunderstandings or incomplete specifications. Several relevant studies on the use of LLMs in the requirements elicitation process are presented below.

Authors in [6] propose *Elicitron*, a framework designed to leverage LLMs for simulating diverse user agents and uncovering both explicit and latent needs during the requirements elicitation process. The framework employs a multi-stage approach, starting with the generation of user agents using GPT-based models, where agents are created either serially to maintain contextual diversity or in parallel with clustering techniques to ensure variability. These agents then simulate interactions with a hypothetical product, providing structured feedback in terms of actions, observations, and challenges. Simulated interviews are conducted to gather detailed insights, which are subsequently analyzed by the LLM to identify latent needs requiring innovative design solutions. The framework is validated through experiments that demonstrate its ability to identify more diverse and latent needs

compared to traditional methods, showcasing the potential of LLMs like GPT-4-turbo to improve the scalability and depth of requirements elicitation processes.

The second study [53] focuses on exploring ChatGPT's capabilities in eliciting requirements for Trustworthy AI systems. Using the GPT-3.5 model, the authors developed a controlled methodology to compare ChatGPT-generated requirements with those formulated by human experts. The study employs six carefully crafted questions aimed at eliciting requirements related to key attributes of Trustworthy AI, such as accuracy, transparency, and privacy. ChatGPT's responses were evaluated alongside 30 requirements collected from domain experts by a separate group of evaluators, who assessed them across seven quality attributes: abstraction, atomicity, consistency, correctness, unambiguity, understandability, and feasibility. The findings reveal that ChatGPT excels in abstraction, consistency, and understandability, often outperforming human experts in these dimensions. However, the model showed limitations in producing unambiguous and feasible requirements, highlighting challenges in its ability to address domain-specific complexity and practicality. Despite these limitations, the study underscores the potential of LLMs to support and enhance the requirements elicitation process by reducing time and effort while improving quality in key areas.

Moving on to other analysis techniques for the initial phases of planning ISs, the authors in [46] propose an interesting approach to automate and validate SWOT analysis,³ using LLMs. This approach focuses on portfolio construction, extracting SWOT elements from company reports (10-K filings) using tools like ChatPDF. A key component of the methodology is prompt engineering, avoiding reliance on general knowledge. This iterative process improves the extraction quality, ensuring that each SWOT element is supported by specific evidence. Each feature is assigned a weighted score (high, medium, or low), allowing for an assessment of its significance. For validation, the authors use LLMs and embedding models to verify the extracted features. The semantic alignment of the extracted SWOT elements is assessed through cosine distance between text embeddings, which measures similarity and ensures the accuracy of the extracted data. Additional spot-checking using metrics like Damerau-Levenshtein⁴ distance further ensures consistency and accuracy, addressing potential hallucination issues in MLM outputs. The authors demonstrate the methodology by constructing and validating portfolios for 471 S&P 500 companies. These portfolios, based on SWOT analysis, are compared with those constructed using traditional Sharpe ratio metrics.

Finally, in [4], authors propose a study that highlights how LLMs, particularly ChatGPT, can streamline requirements elicitation while using SWOT analysis as a structured approach to assess their impact across different stages of requirements

³ A SWOT analysis is a strategic planning tool used to assess the internal and external factors that can impact a business, project, or organization. It consists of four key elements, Strengths, Weaknesses, Opportunities, and Threats [37, 56].

⁴ Damerau-Levenshtein metric calculates the minimum number of edits (insertions, deletions, substitutions, or transpositions) required to transform one string into another.

engineering (RE). They focus on using LLMs like ChatGPT (GPT-3.5) to address challenges in four key RE stages: requirements elicitation, specification, analysis, and validation. The authors emphasize the potential of LLMs to enhance efficiency, bridge communication gaps, and improve the accuracy of requirements-related tasks through automated and interactive capabilities. The study outlines a vision for an LLM-driven RE process that integrates LLM agents for various RE tasks. These agents use advanced prompting techniques, such as zero-shot, few-shot, and chain-of-thought prompting, to assist stakeholders in generating, refining, and validating requirements. The methodology is tested on a real-world project (ActApp), designed to assist patients with type 2 diabetes, to evaluate the feasibility of LLMs in requirements elicitation. The evaluation involves analyzing how ChatGPT generates user-story-style requirements and comparing them against expert-identified requirements. For each RE stage, SWOT analysis is performed, identifying strengths (e.g., efficient data processing, domain knowledge assimilation), weaknesses (e.g., lack of empathy, risk of misinterpretation), opportunities (e.g., real-time feedback, human-centric elicitation), and threats (e.g., data privacy concerns, potential biases).

The use of LLMs in the IS planning phase is expanding to cover various tasks, in particular requirements elicitation and SWOT analysis support. Their use relies on identifying explicit and latent needs by simulating user interactions and refining requirements through iterative analysis. Their capabilities are further enhanced through advanced prompting techniques, embedding models, and validation methods that improve accuracy and reduce errors and inconsistencies. LLMs also facilitate automated SWOT analysis, extracting and verifying strategic insights from official company documents.

5.2.2 Design

The concept of *Conversational Process Modeling* (*ConverMod*) was introduced in [34] to transform traditional business process modeling practices by leveraging the potential of conversational AI. This interactive and iterative approach supports domain experts in extracting, modeling, and refining business process descriptions with the assistance of LLMs. Cooperation between domain experts and process analysts is essential to bridge the gap between these roles and facilitate the generation of Business Process Model and Notation (BPMN) diagrams. Starting from prompts designed with textual process description and additional context (e.g., a list of tasks extracted from it, a set of custom rules for textual graph representation), the LLMs are guided to generate the desired output. To ensure compatibility with the limited token context of LLMs, ConverMod exploits intermediary textual notation (TN) of BPMN such as Mermaid.js (MER) to represent BPMN diagrams in their textual definition. The result of MER is then converted into the graphical representation (GR) with the BPMN2Constraints tool. The final model generated by LLMs is then reviewed and corrected to align with BPMN standards. The textual description from which the model is generated is contained in the PET dataset that is used as

standard. Using the PET dataset as the ground truth, the performances of LLMs in task extraction and process model generation are evaluated both quantitatively and qualitatively.

In [35] has been proposed a framework to generate process models exploiting LLMs. This framework transforms textual descriptions into structured, standard formats such as BPMN and Petri nets. A key component is the iterative feedback loop, which is based on user's feedback to refine the generated process model progressively. The framework starts with prompt engineering techniques, where textual descriptions are enhanced with additional information to guide the LLM in the most effective way. These techniques include role prompting, knowledge injection, few-shot learning, and negative prompting. After querying the LLM, it generates an intermediate process representation in Partially Ordered Workflow Language (POWL). The model in POWL goes under the error-handling and refinement step that categorizes the errors as critical (i.e., which disrupts the system's functionality or compromises security) and adjustable (i.e., which affects qualitative aspects). This step incorporates iterative refinements to be aligned with user expectations. Generated models in POWL are then converted into BPMN or Petri nets for display and export purposes. The framework's evaluation focuses on the effectiveness of integrating LLMs and its performances against existing approaches.

The authors in [16] explored the application of LLMs in software modeling tasks, specifically focusing on realizing UML class diagrams enriched with Object Constraint Language (OCL) constraints. The implemented approach unfolds in two different phases: (i) exploration and (ii) focused experiments. In the former, a conversational prompt-based approach was performed to identify modeling capabilities and limitations. During this phase, there were no constraints on the content of the prompts. In the latter, a more specific modeling task was performed in order to highlight concepts like inheritance, enumerations, determinism, abstraction, and OCL constraints. In this phase, the prompts were limited only to a set of predefined model arguments. Each model, which has been generated starting from user intent used as initial prompt, went through a refinement stage to test the adaptability of LLMs to corrections. The evaluation of LLMs applications in software modeling tasks spanned on their syntactic and semantic accuracy, scalability, handling of modeling concepts, and prompt sensitivity.

The study conducted in [22] explores the capabilities of LLMs for conceptual modeling tasks. The modeling languages analyzed are Entity-Relationship (ER) diagrams, BPMN, UML class diagrams, and Heraklit models. The approach employed relies on the use of different prompts carefully designed to generate models in the desired language format. This process exploits a combination of few-shot learning (i.e., providing some examples in the prompt) and zero-shot learning (i.e., relying on only instruction contained in the prompt). The prompts also include specifics of the output's format such as JSON for ER and BPM, PlantUML for UML class diagrams, or custom JSON-based representation for Heraklit models. Moreover, the proposed approach also shows the interpretation capabilities of LLMs by extracting textual descriptions from pre-defined diagrams in a simplified custom

textual notation. The generated results of this approach were assessed based on correctness, completeness, and adherence to the format and prompt constraints.

Authors in [15] demonstrate the capabilities of several multimodal LLMs to transform images of hand-drawn UML class diagrams into corresponding computer-readable PlantUML notation. The implemented approach foresees the use of four hand-drawn UML diagrams and three different prompts. The first three UML diagrams are of increasing complexity (e.g., specifying relationships, attributes, and cardinalities), and the last one is a diagram with correct syntax but semantically unrealistic. Each prompt has been designed to increase the level of detail used to describe the task. Each LLM was fed with each diagram using all three prompts, and each prompt-model combination was tested three times to assess consistency. The final results were evaluated on accuracy and completeness with respect to the original hand-drawn UML diagrams.

LLMs are increasingly used in the design phase of software and process modeling to extract structured representations such as UML diagrams, BPMN models, and Petri nets from textual descriptions or images. Their capability to interpret and refine models is enhanced through prompt engineering techniques, iterative feedback loops, and intermediate notations like POWL or MER. These models demonstrate adaptability in handling errors, improving semantic and syntactic accuracy through user interaction and refinement processes. Furthermore, LLMs exhibit multimodal capabilities, transforming hand-drawn UML diagrams into structured digital representations, enabling seamless integration with standard modeling tools.

5.2.3 Development

In [52], the authors proposed a framework called *GeneUS*, designed to automate the generation of user stories and associated test case specifications directly from software requirements documents—the output of the requirements engineering (RE) phase. The framework leverages the natural language understanding and generation capabilities of LLMs. The outputs generated by *GeneUS* are in JSON format to facilitate the integration with popular project management tools such as Jira and Azure DevOps. The core of this framework is Refine and Thought (RaT), an optimized version of Chain of Thought (CoT) prompting. RaT operates in two stages: a *Refinement Step*, where redundant or non-textual information (e.g., diagrams, symbols) is filtered to clean the input text, and *Thought Step*, where the refined input text is used to generate structured outputs (i.e., user stories and associated test cases). *GeneUS* functionalities can be summarized in three main steps, requirement extraction in which functional and non-functional requirements are extracted from the input document, user story generation in which these extracted requirements are converted into structured user stories with clear objectives and deliverables, and test case generation in which for each user story associated test case are generated. The framework functionalities are assisted with sequential RaT prompts to ensure

accuracy and completeness. The effectiveness of the framework has been tested on seven mid-sized RE documents, and a survey based on the RUST framework has also been conducted.

The authors of [20] proposed a self-collaboration framework for code generation, which utilizes multiple specialized LLMs-based agents to simulate a collaborative software development team. The aims of this framework is to improve the quality and correctness of the generated code by assigning specific software engineering roles (e.g., *Analyst*, *Coder*, and *Tester*), with each role assigned to a separate LLM agent. To ensure effective role execution, each LLM agent operates under specifically designed role instructions, guiding task execution in their domain. The framework is structured around two key components, (i) Division of Labor (DOL) and (ii) Collaboration. In the DOL the division of the software development tasks happens—in particular, the Analyst decomposes high-level requirements into detailed subtasks and formulates a high-level plan for coding, the Coder implements code according to the analyst’s plan and refines the code based on feedback, and the Tester evaluates the generated code by identifying bugs and providing structured feedback through test reports. In the Collaboration phase, agents interact through a shared blackboard mechanism in which outputs are shared and iteratively refined, first from Analyst to Coder and then in iterative cycles between Coder to Tester until the Tester’s feedback is fully addressed by the Coder. The framework adopts real-world development methodologies, specifically the waterfall model, where the process is sequential but allows for feedback loops to enable refinement. Its performance has been evaluated on benchmark datasets, such as HumanEval [13], HumanEval-ET [19], MBPP [8], and MBPP-ET, and compared against other LLMs using the Pass@1 metric, which measures the correctness of the top-1 generated code after execution. Two experimental settings were considered, one combining natural language descriptions, function signatures, and public test cases and another relying solely on natural language, simulating real-world scenarios with only textual requirements.

The study presented in [28] introduces a self-planning code generation framework designed to enhance the performances of LLMs in handling complex programming tasks. The framework is composed of a two-phase approach that guides LLMs through the code generation process. The first phase is the Planning phase, in which the LLM generates a high-level plan containing a sequence of steps to solve the given programming task. This generated plan simplifies the problem into manageable subproblems used to guide the LLM during the code generation phase. Through few-shot prompting, the model is fed with examples of how to convert problem descriptions into solution steps. The second step is the Implementation phase, where the model generates code step by step according to the plan produced in the previous phase. The high-level plan is appended to the original problem statement and used as input to the model to guide the code generation process. The framework was evaluated on multiple code generation benchmarks (e.g., HumanEval, MBPP-ET) against different LLMs. Evaluation metrics used for the evaluation include Pass@1, Pass@k, AvgPassRatio, and CodeBLEU. Moreover, a

human evaluation on correctness, readability, and robustness criteria was performed against direct generation and CoT prompting.

In [41] has been proposed a prompting technique called *structured chain-of-thought (SCoT)* to increase the performance of LLMs in code generation tasks. SCoT aims to improve code generation by aligning intermediate reasoning with program structures (i.e., sequence, branch, and loop structures) to guide LLMs more effectively during code generation. These structures mirror how human developers logically approach coding tasks. This technique uses a two-step generation pipeline. In the first step, the LLM generates a structured reasoning process (i.e., SCoT) that decomposes the problem using program structures and defines input-output parameters. In the second step, the LLM uses the generated SCoT as a soft template to produce the final code. This two-step approach helps mitigate errors in SCoT generation and allows for refinements. Moreover, to further assist the LLM in code generation, SCoT includes explicit input-output (I/O) specifications. To evaluate the performances of the proposed technique, experiments have been conducted on three code generation benchmarks (i.e., HumanEval, MBPP, MBCPP [7]) against different standard prompting methods (i.e., zero-shot, few-shot, CoT) using as evaluation metric Pass@k. Human evaluation and ablation studies⁵ have also been performed.

The use of LLMs in the development phase of IS has rapidly increased. They are employed in various tasks, in particular to automate code generation, requirement analysis, and test case specification, improving efficiency and reducing manual effort. Their performance is enhanced through structured reasoning techniques and multi-step prompting strategies that refine outputs iteratively, to improve code correctness, readability, and robustness. Furthermore, they facilitate seamless integration with project management tools by structuring outputs in standardized formats. Through iterative refinement and self-planning methodologies, LLMs continue to bridge the gap between natural language understanding and high-quality software development.

5.2.4 Testing and Validation

In [42], *QTypist* was introduced as a novel approach based on LLMs to automate text input generation according to the GUI context. QTypist requires two types of information to generate a valid text response: one extracted directly from the UI page with the text input and another from pre-existing data sources. At first, QTypist extracts context information from the view hierarchy file of a GUI page. The extracted context information is composed of three types: input widget, local context, and global context. Based on this extracted context, 14 linguistic patterns

⁵ The contribution of a component in an AI system is determined by removing it and then analyzing the performance of the system after the removal.

are designed to construct precise prompts for the LLM. Furthermore, exploiting the Rico dataset [17] and other sources, three types of input widgets are identified, which include search list, popup menu, and filled content. These types of input widgets and their associated candidate input contents are used to generate pairs of prompts and answers for model tuning. Finally, the prompt-tuned LLM, using the prompts generated from the GUI page’s context information, generates responses to enhance automated GUI testing tasks. The QTypist approach was evaluated considering the effectiveness of text input generation and the quality of generated inputs.

The authors of [21] proposed *AdbGPT*, a framework for automating Android bug replay tasks from bug reports. This framework employs LLMs through prompt engineering and techniques such as few-shot learning and chain-of-thought reasoning to reproduce bugs. AdbGPT is splitted in two main phases, Steps-to-Reproduce (S2R) extraction and Guided Replay, both of which exploit LLMs to accomplish their task. The former has the task of extracting S2R entities from bug reports, such as action types, target components, input values, and scroll directions. The extraction process occurs by exploiting a prompt composed of available actions, action primitives, input examples, chain-of-thought reasoning, and output examples. The latter has the task of matching the extracted S2R entities with a sequence of GUI events needed to dynamically replicate the bug steps. The first step is to encode the GUI into a custom HTML-like syntax, which is derived from the Android view hierarchy, to be processed by the LLM. Finally, the LLM is prompted with the encoded GUI, input example, chain-of-thought reasoning, and output example in order to reproduce the step. The performance of AdbGPT was evaluated in terms of accuracy, efficiency, and practical usefulness.

The study made in [14] highlights the capabilities of LLMs in code generation and debugging tasks. They developed *SELF-DEBUGGING* framework, which is designed to enhance pre-trained LLMs capabilities without any further fine-tuning relying on prompt-based instructions. This framework enables LLMs to debug their own generated code through an iterative approach. One single execution of the framework consists of three steps, Generation, in which LLM generates candidate programs relying solely on the problem description; Explanation, in which the generated code is described in detail via a line-by-line approach (i.e., rubber duck debugging), and in some cases, this explanation also includes execution traces; and Feedback, in which the model generates a feedback message indicating whether the code is correct or needs revision. The debugging process terminates when the code is correct or a maximum iteration limit is reached. The framework capabilities are evaluated across multiple benchmarks such as Spider for text-to-SQL generation [75], TransCoder for code translation [54], and MBPP for text-to-python generation [8]. Furthermore, SELF-DEBUGGING is compared also against other baseline approaches like MBR-Exec [59] and Coder-Reviewer [76].

In [67] has been proposed *Replit*, a novel framework that combines LLMs and Completion Engines for Automated Program Repair (APR). This approach aims to remove tokens generated by the LLM under the guidance of a Completion Engine in order to produce more valid and accurate patches during the repair

process. In detail, the LLM provides probabilistic predictions for the next token during patch generation, and the Completion Engine verifies the feasibility of the suggested tokens. Authors consider the repair problem as a cloze-style task [71] in which the buggy code is replaced with a masked token (``), and the LLM is used to generate the patch based on the surrounding context. Repilot iteratively generates patches through a feedback loop in which the LLM produces candidate tokens and probabilities. Infeasible tokens are removed from the search space and rejected by the Completion Engine. Instead, accepted tokens are appended to the patch, allowing the loops to continue until the patch is complete. To store previously rejected or accepted tokens, a prefix tree data structure (Trie) is used in order to reduce the overhead to the Completion Engine. Repilot's effectiveness has been compared against 19 APR tools across traditional, Neural Machine Translation-based (NMT), and LLM-based, using the Defects4J dataset (versions 1.2 and 2.0) [31] containing buggy Java programs. Evaluation metrics of the results of the proposed approach cover bug-fixing capabilities, compilation rate, and generalizability.

An interesting study proposed in [58] evaluates the use of the tool TESTPILOT, which incorporates LLMs for automated unit test generation for JavaScript projects. This tool relies on five distinct components (i.e., *API explorer*, *documentation miner*, *prompt generator*, *test validator*, *prompt refiner*) organized in four phases. TESTPILOT starts by using dynamic analysis to explore the API of the Package Under Test (PUT) in order to identify elements such as methods, constants, and signatures. Moreover, in this first phase, it also records the API's structure and gathers, from documentation, metadata like comments and usage examples if available. In the second phase, the prompt is constructed using up to five elements, the function signature, the function definition if available, the documentation comments, the usage examples if available, and failing tests and error messages if generated in a successive phase of this approach. The initial results that are obtained from the LLM using this prompt are validated for syntax correctness, and if they are syntactically valid, they are executed to identify errors or assertion failures. If a test fails, the fourth and last phase is in charge of refining the prompt by including the failing test and error response to assist the LLM in correcting the test. TESTPILOT's effectiveness has been compared using a benchmark of 25 npm packages with Nessie [5], a state-of-the-art feedback-directed test generator. The involved metrics are the quality of generated tests in terms of contained assertions, coverage of tests that include non-trivial assertions, and similarity with generated tests and already existing ones. Furthermore, a study has also been conducted to compare employed LLMs.

The study presented in [30] introduces *RING*, a multilingual repair engine that is powered by LLMs. The LLM chosen by the authors is used for its ability to perform few-shot learning and adapt to multilingual program repair tasks. *RING* program repair operation is divided into three key stages, fault localization, code transformation, and candidate ranking. The first step exploits compiler diagnostics, static analyzers, and linters to locate syntactic and semantic errors. Error messages are normalized to enforce consistency across languages. The middle step, leveraging

few-shot learning, uses a collection of buggy-fixed code pairs to guide the repair process by selecting programs with similar mistakes. Similar examples are selected based on error similarity using *error vector selection* and *message embedding selection* according to the selected language. The selected pairs are embedded into the prompt to show potential fixes for similar errors, allowing the LLM to generate more accurate candidate repairs. The last phase is in charge of ranking LLM's output candidates based on the average log probabilities of the tokens generated. Finally, this study compares RING against language-specific repair engines (e.g., LaMirage [9], TFix [10], and BIFI [74]) across different programming languages (i.e., Excel, Power Fx, Python, JavaScript, C, and PowerShell).

The use of LLMs in the testing and validation phase of the IS is growing, particularly in automated debugging, bug reproduction, and unit test generation. The accuracy and efficiency of these tasks are enhanced by leveraging prompt engineering, few-shot learning, and iterative feedback mechanisms to refine the outputs. LLMs also assist in program repair by integrating probabilistic token prediction, multilingual fault localization, and error-driven correction techniques. Moreover, they are also used in GUI testing and automated test case generation to improve coverage and validation, reducing manual effort.

5.2.5 Monitoring and Maintenance

Monitoring is a critical phase in the life cycle of information systems, ensuring their reliability, security, and performance over time. We identified three key areas of monitoring: log parsing, anomaly detection, and Intrusion Detection Systems (IDS). Log parsing focuses on extracting meaningful information from unstructured log files, a task essential for diagnosing system issues, tracking performance, and ensuring compliance. Anomaly detection involves identifying deviations from expected system behavior, which could indicate issues ranging from performance bottlenecks to security breaches. Finally, IDS safeguards information systems by identifying and mitigating potential security threats.

In [72], authors propose *UniLog*, an innovative approach that leverages LLMs and in-context learning (ICL)⁶ to enhance the automatic generation of logging statements in software systems. They identify the limitations of existing logging tools, which often address only specific sub-problems related to logging, such as determining verbosity levels or suggesting log messages, without providing a comprehensive solution that simultaneously tackles the determination of logging positions, generation of log messages, and setting of verbosity levels. By utilizing the ICL paradigm, UniLog circumvents the need for extensive training on large datasets, instead relying on the LLM's ability to infer appropriate logging statements

⁶ ICL is a paradigm that allows language models to learn tasks given only a few examples in the form of demonstration [2].

from provided code examples. This method not only improves the practicality of logging in real-world scenarios but also aligns with the fast deployment and interpretability needs of software engineering, ultimately achieving state-of-the-art performance in logging tasks. The evaluation of UniLog involved testing its automated logging capabilities on a dataset comprising 12,012 code snippets extracted from 1,465 GitHub repositories.

Gu et al. [29] and Ma et al. [44] focus on the use of LLMs for log parsing, aiming to improve the extraction of log templates and parameters from log messages.

In the first study, the authors propose *LILAC*, a log parsing framework designed to address the challenges of current log parsing techniques, such as dependency on handcrafted rules and limited training data. *LILAC* leverages an in-context learning (ICL) approach and enhances LLM capabilities through a hierarchical candidate sampling algorithm to select high-quality log message demonstrations. Additionally, the authors introduce an adaptive parsing cache that stores and refines generated log templates, enabling faster and more consistent parsing. *LILAC* is extensively evaluated on large-scale public datasets, achieving a 69.5% improvement in template accuracy F1 score over state-of-the-art methods while maintaining comparable efficiency to traditional parsers. The second study introduces *LLMParser*, a framework that approaches log parsing as a text-to-text or text generation task, utilizing LLMs such as FLAN-T5-SMALL, FLAN-T5-BASE, LLAMA-7B, and CHATGLM-6B. The authors employ few-shot tuning to adapt the models for translating raw logs into structured log templates. Evaluations on a benchmark dataset of logs from 16 open-source systems reveal that *LLMParser* achieves a parsing accuracy of up to 96%, outperforming traditional parsers like Drain, Logram, and LogPPT. The study also examines the effects of training size, model architecture, and pre-training on performance, finding that smaller models like Flan-T5-base provide similar accuracy to larger models with reduced computational overhead. Additionally, the authors explore limitations and propose future directions for enhancing the adaptability and efficiency of LLM-based log parsers.

Authors in [25] propose *LLMeLog*, a novel approach for anomaly detection that utilizes LLMs to enhance log event representations by addressing three identified anti-patterns: content missing, semantic deflection, and tendency lacks. *LLMeLog* employs a structured prompt designed for in-context learning to enrich log events with domain knowledge, including anomaly tendencies. Enriched events are then used to fine-tune a pre-trained BERT model through a hierarchical semantic fine-tuning process, which incorporates a novel hierarchical metric loss to jointly encode semantic information and anomaly tendencies into event embeddings. Finally, these embeddings are fed into a transformer-based anomaly detection model that captures temporal patterns for effective anomaly detection. Experimental evaluations on three public datasets (HDFS, BGL, and Thunderbird) demonstrate *LLMeLog*'s superior performance, achieving F1-scores consistently exceeding 99%, even when trained on as little as 10% of labeled data.

Focusing on another critical aspect of IS monitoring, specifically *Intrusion Detection Systems (IDS)*, the authors in [32] investigate the potential of large language models (LLMs) to enhance cybersecurity through advanced log analysis

and intrusion detection. The study compares the performance of general-purpose LLMs, fine-tuned task-specific LLMs, and traditional machine learning models using real-world commercial deployment data. Key strategies explored include few-shot prompting, retrieval-augmented generation, and a decoupled framework where LLMs are employed to classify attack types and generate detailed incident reports. The findings reveal that while general-purpose LLMs face challenges related to privacy concerns and computational efficiency, fine-tuned LLMs demonstrate strong performance in identifying known attack patterns. However, their limited effectiveness against zero-day threats underscores the need for further research to improve LLM adaptability and robustness in dynamic cybersecurity environments. Several recent works have proposed innovative approaches that integrate LLMs into cybersecurity frameworks to improve intrusion detection, penetration testing, and threat analysis. *HuntGPT* [2] integrates explainable AI techniques with LLM, using a Random Forest classifier alongside SHAP and Lime frameworks to improve interpretability. GPT-3.5 Turbo is used to generate comprehensible threat insights, which assists cybersecurity analysts in decision-making. Moreover, *SecGPT* [70] introduces an interactive LLM agent designed to streamline penetration testing. This system leverages a Generative Adversarial Network (GAN) architecture to synthesize data for model retraining, enabling the generation of new attack profiles and enhancing adaptability against evolving threats. Finally, *PentestGPT* [18] is a penetration testing tool that uses LLMs to automate reasoning, generation, and parsing tasks. It addresses key challenges such as context loss and inaccurate result generation, providing a more efficient and intelligent penetration testing process.

The use of LLMs in the monitoring and maintenance phase of IS is increasing in tasks such as log parsing, anomaly detection, and intrusion detection. The ability of LLMs to extract meaningful insights from unstructured logs enhances system diagnostics, performance tracking, and compliance monitoring. Log-analysis accuracy and anomaly detection are improved through in-context learning, hierarchical fine-tuning, and structured prompting. Through continuous advancements, LLMs are becoming a fundamental component in system monitoring, strengthening reliability and security in dynamic IT environments.

5.2.6 Beyond the IS Life Cycle: Other Areas Enhanced by LLMs

Tasks like information extraction and data analytics are not traditionally considered part of the IS life cycle, even though the use of LLMs has shown many advantages in these kinds of areas, introducing many benefits beyond conventional applications. Their ability to process vast amounts of structured and unstructured data allows them to extract meaningful insights, automate complex reasoning tasks, and improve decision-making processes. In detail, in data analytics tasks, LLMs facilitate seamless interaction with structured repositories through natural language interfaces, improving accessibility to large-scale data sources while

ensuring interpretability and efficiency. In information extraction tasks, LLMs enhance annotation, relation extraction, and multi-modal understanding by leveraging advanced prompting strategies and alignment techniques, reducing manual effort and improving accuracy. The possibility to integrate LLMs with external tools, domain-specific frameworks, and multi-modal architectures further improves their versatility across different tasks and domains, spacing from tabular reasoning to video comprehension, bridging the gaps in data-driven workflows and enhancing automation, augmentation, and analysis capabilities.

5.2.6.1 Data Analytics

In the context of data analytics for information systems, large language models have shown several capabilities that can be categorized into three main areas: first, data analytics on sample data, which includes tasks like tabular question answering (Tabular QA) to extract insights from structured datasets; second, data analytics on external data sources, where LLMs are used with relational databases through text-to-SQL models or with knowledge graphs via text-to-SPARQL interfaces, enabling seamless querying and integration with large-scale, structured repositories; and finally, data analytics on multiple external data sources involving the use of LLM agents and sophisticated frameworks that leverage LLMs to interact with diverse datasets, perform reasoning across different domains, and synthesize actionable insights. These categories represent the versatility of LLMs in addressing a wide range of data-driven challenges within the monitoring phase of information systems.

In [65] authors propose a novel framework, *Chain-of-Table*, designed to improve table-based reasoning tasks, such as question answering and fact verification, by integrating evolving tabular transformations into the reasoning process of large language models. The framework extends the concept of chain-of-thought (CoT) prompting [55], transforming tables through iterative, structured operations to better align intermediate reasoning steps with the final answer. The approach introduces a dynamic mechanism where LLMs iteratively apply pre-defined tabular operations, including adding columns, selecting rows, grouping, and sorting. At each step, the model dynamically plans the next operation based on the current state of the table, the reasoning history, and the question or task. This sequence forms a *chain of tables*, representing the logical steps of reasoning through intermediate transformations. By explicitly encoding these operations, the framework provides structured intermediate results that improve prediction accuracy and transparency compared to free-form reasoning approaches. To validate the proposed methodology, the authors conduct experiments on three established benchmarks for table understanding: WikiTQ [50] (table-based question answering), TabFact [12] (table-based fact verification), and FeTaQA [48] (free-form table-based question answering). The framework demonstrates state-of-the-art performance across these datasets, surpassing existing generic reasoning methods (e.g., chain-of-thought) and program-aided reasoning approaches (e.g., text-to-SQL). Notably, it achieves superior results with fewer steps required to arrive at the correct answer, showcasing its efficiency. The

study also analyzes the robustness of the method across different table sizes and the complexity of the required operations. The results highlight the framework’s ability to handle large tables and multi-step reasoning tasks more gracefully than competing methods. The proposed approach uses LLMs such as PaLM 2 and GPT-3.5, leveraging their in-context learning capabilities to dynamically plan and execute operations while reducing query inefficiencies. The authors conclude that Chain-of-Table provides a significant advancement in LLM-driven table understanding, enabling more reliable and interpretable results by explicitly encoding intermediate reasoning steps into structured tabular transformations.

Moreover, in [11], *Plan-of-SQLs (POS)* is proposed, a novel framework for Table Question Answering (Table QA) that emphasizes interpretability, efficiency, and competitive accuracy. The study addresses a critical gap in Table QA literature: while recent advances using large language models (LLMs) have significantly improved performance, their decision-making processes often remain opaque, particularly in high-stakes industries like finance and healthcare. POS aims to bridge this gap by decomposing complex queries into a sequence of simple, atomic SQL operations, enabling transparency in reasoning and facilitating user trust. The POS framework operates in three primary stages. First, a Natural Language Atomic Planner takes an input query and generates a sequence of sub-queries in natural language, breaking down complex tasks into atomic steps. Each step corresponds to a minimal SQL operation, such as filtering rows or sorting columns, ensuring traceability and interpretability. Second, the Step-to-SQL Module converts each atomic step into an SQL command, executing the operation on the input table. The resulting intermediate tables are then used as inputs for subsequent steps, forming a sequential chain of transformations. Finally, the output of the last SQL operation provides the answer to the query, along with detailed attribution maps that highlight the specific rows and columns contributing to the answer. To evaluate POS, the authors conduct experiments on three benchmark datasets: TabFact, WikiTQ, and FeTaQA. POS demonstrates competitive or superior accuracy compared to existing Table QA methods, including text-to-SQL, Chain-of-Table, and DATER. Notably, it achieves higher interpretability scores in human and LLM-based evaluations, with users consistently preferring POS explanations due to their clarity and coherence. The framework also outperforms baseline methods in efficiency, requiring significantly fewer LLM calls and database queries per task, thereby reducing computational costs. The study highlights POS’s robustness in handling complex queries and its alignment with human reasoning. The findings suggest that POS not only enhances the accuracy and efficiency of Table QA but also addresses key interpretability challenges, making it particularly valuable for applications in domains where accountability and transparency are critical.

Another notable study is *TALM: Tool Augmented Language Models* [49]. The authors propose a framework for enhancing LLMs with external tools to address limitations in handling dynamic, private, or computationally intensive data sources. This approach integrates LLMs with external tools through a text-to-text API interface, enabling the models to generate tool queries, process tool outputs, and incorporate the results into final answers. The study highlights the frame-

work’s application to two distinct domains: knowledge-oriented question answering using the Natural Questions (NQ) dataset and reasoning tasks using the MathQA dataset [3]. TALM operates in two stages. First, the model generates a tool input based on the task’s context and calls the relevant tool via an API. Once the tool outputs the results, these are appended to the input sequence, allowing the model to generate an informed response. To improve performance on tasks with limited tool-use demonstrations, the authors employ an iterative self-play mechanism. This involves bootstrapping tool-use datasets from a few labeled examples and iteratively refining the dataset by validating generated outputs against task-specific metrics. The iterative self-play process improves TALM’s ability to generalize and handle multi-step reasoning tasks. The evaluation demonstrates TALM’s effectiveness across domains. For Natural Questions (NQ)[36], TALM accesses a BM25-based retriever tool to extract answer spans from indexed documents, outperforming non-augmented LLMs in knowledge-heavy QA tasks. In MathQA, TALM interacts with a custom math solver API to evaluate formulas and generate correct solutions for math word problems, achieving significant performance improvements over baseline LLMs. TALM also showcases robustness in handling out-of-distribution tasks, such as changing world knowledge or operations involving large numbers, where traditional LLMs fail. The study concludes that TALM enables smaller models to outperform larger non-augmented LLMs by leveraging external tools and iterative self-play, offering a scalable and cost-effective approach to enhancing LLM capabilities for complex, multi-source data analytics tasks.

Here [26] a framework for automating end-to-end data science workflows using a LLM-based agent is presented. This framework introduces two core modules: (i) Hierarchical Graph Modeling and (ii) Programmable Node Generation. The former decomposes complex data science workflows into a structured graph, where tasks are represented as nodes and dependencies as edges. This approach dynamically adjusts to evolving data and task requirements by iteratively refining the graph structure. The latter ensures each subproblem is accurately defined, refined, and verified by dynamically generating and optimizing task-specific code snippets. These nodes integrate external tools and libraries for robust task execution. The system uses contextual feedback to refine both task and action graphs iteratively, enabling adaptive execution and debugging in real-time. The Data Interpreter demonstrates significant improvements in data analytics, machine learning, and open-ended tasks across multiple benchmarks, showcasing adaptability and precision. Extensive evaluations highlight its superiority, with a 25% boost on the InfiAgent-DABench [27] dataset and marked advancements in machine learning and mathematical problem-solving tasks.

Furthermore, Qiao et al. [51] introduce *TaskWeaver*, a code-first framework designed to enable LLM-powered autonomous agents for complex task execution. TaskWeaver leverages LLMs to transform natural language user requests into executable Python code, integrating both custom and pre-defined plugins for diverse applications. The framework supports rich data structures (e.g., pandas DataFrames), ensures stateful code execution, and incorporates domain-specific knowledge via user-provided examples. A two-layer architecture is employed,

consisting of a Planner and a Code Interpreter (CI). The Planner decomposes high-level tasks into sub-tasks and manages their execution, while the CI generates and executes code, ensuring compliance with safety rules. TaskWeaver includes features such as plugin-only mode, dynamic plugin selection, secure code execution, and error-handling mechanisms like post-verification and auto-correction. Its modular design allows for the integration of additional functionalities and ensures scalability. The framework is validated through various case studies, including anomaly detection in time series data and stock price forecasting using ARIMA models [60].

Lastly, Wang et al. [69] present *AutoGen* a generalized framework for developing LLM-powered applications through multi-agent conversation systems. AutoGen provides a versatile infrastructure where agents, equipped with LLMs, tools, and human inputs, collaborate via customizable and dynamic conversation patterns. The framework supports complex workflows by defining agents with specific roles (e.g., problem-solving, retrieval, or decision-making) and programming their interactions using a mix of natural and programming languages. AutoGen's core innovations include Conversable Agents and Conversation Programming. Conversable Agents are modular and reusable entities that communicate through structured message exchanges, while Conversation Programming simplifies workflow orchestration by designing computation and control flow around inter-agent dialogs. These features enable agents to autonomously coordinate tasks, adapt to evolving challenges, and incorporate feedback for iterative refinement. The framework demonstrates its capabilities through diverse applications, including mathematical problem-solving, retrieval-augmented question answering, interactive decision-making, and dynamic group collaboration. AutoGen achieves state-of-the-art results by leveraging dynamic conversation patterns, integrating tool-backed and LLM-backed agents, and supporting human intervention where necessary. Experimental evaluations reveal AutoGen's ability to streamline workflows, enhance modularity, and deliver significant performance improvements over existing single-agent and multi-agent approaches.

5.2.6.2 Information Extraction

The study proposed in [78] introduces *LLMAAA*, a framework that uses LLMs as active annotators to create high-quality training datasets with minimal human intervention. The ability of LLMs to provide pseudo labels in a few-shot learning paradigm is empowered through active learning and prompt engineering strategies. The aim of *LLMAAA* is to make LLMs efficient and reliable annotators by amalgamating active learning into the annotation process. It iteratively generates stronger task-specific models (TAMs) through three stages, LLM annotator, active data acquisition, and robust learning with noisy labels. In the first step, LLMs generate pseudo-labels for NLP tasks such as named entity recognition (NER) and relation extraction (RE), enhancing LLM performances by using k-NN example retrieval and Label Verbalizer techniques. In the next phase, active learning strategies identify the most useful examples to annotate, reducing the effort of

LLMs. These strategies include Maximum Entropy, Least Confidence, and K-Means Clustering. Each annotated example is added to the labeled dataset, which gradually improves the task-specific model's performance. The final operation performed by LLMAAA is addressing noise in the label generated by the LLM by using an automatic reweighting technique. Performances of LLMAAA are evaluated on NER and RE tasks using datasets like Chinese OntoNotes 4.0 [68], English CoNLL03 [64], and Re-TACRED [61].

The study in [38] highlights how to leverage LLMs and Natural Language Inference (NLI) modules to enhance document-level relation datasets by generating relation triples for *Document-Level Relation Extraction (DocRE)*. Using this framework, the authors proposed DocGNRE which represents an enhanced version of the Re-DocRED dataset [63]. The proposed approach addresses the limitations of manual annotation and prior distant supervision techniques with two main steps. In the first step, a prompt composed of a generation demonstration and a specific context along with an entity list derived from the dataset is paired with each document, allowing the LLM to generate relation triples exclusively from the entities in the derived entity list. To obtain additional triples, LLM-generated outputs are reintroduced into the prompts, allowing the model to extend the set of relation triples iteratively. In the second step, a NLI model is used to map generated triples, often in natural language, to predefined relation types. The evaluation has been conducted under two different configurations, zero-shot document-level RE, and training with the augmented dataset by considering metrics like precision, recall, and F1 score. The approach was also compared to state-of-the-art DocRE models such as ATLOP [79] and KD-DocRE [62].

The authors of [24] proposed *Img2LLM*, a framework designed to be LLM-agnostic, capable of interfacing with any frozen LLM to perform zero-shot Visual Question Answering (VQA). This framework addresses the challenges of modality disconnect between language and vision and task disconnect between language modeling and VQA exploiting the creation of intermediate textual prompts that bridge these gaps. *Img2LLM* leverages a vision-language model (BLIP) [39] to generate question-relevant captions by identifying, through the use of GradCAM, regions of interest in an image that are relevant to the input question. From this generated caption, candidate answers are extracted by identifying key elements (e.g., nouns, verbs, adjectives) through NLP tools such as spaCy. These candidate answers serve as potential answers to questions about the image. For each of them, it is possible to use a question-generation network in order to generate specific questions. The set of question-answer (QA) pairs is specific to the current image, avoiding external training data. The prompts for LLM are composed by concatenating instruction, captions, and QA examples. Final results are derived from the LLM using greedy decoding. *Img2LLM*'s performances have been evaluated on VQAv2 [23], OK-VQA [45], and A-OKVQA [57] and compared against the following three prior VQA methods categories, *zero-shot methods with frozen LLMs* (e.g., PICA [73]), *zero-shot methods with extra multi-modal pretraining* (e.g., Flamingo [1]), and *few-shot methods* (e.g., ClipClap [47]), as well as on different LLMs. Moreover, studies on generalization and ablation have been performed.

In [40], *BLIP-2* is proposed, a framework designed to facilitate vision-language pre-training tasks. The framework, to bridge the gap between vision and language, introduces a lightweight and trainable Querying Transformer (Q-Former) that exploits frozen models (both image encoders and LLMs). Q-Former learns to extract the most relevant visual features from the frozen image encoder. It serves as a bottleneck between the frozen image encoder and the frozen LLM, where it selects and transforms visual data into a format interpretable by frozen LLMs. *BLIP-2* leverages a two-stage pre-training strategy to align visual and textual, *Vision-Language Representation Learning* and *Vision-to-Language Generative Learning*. In the first phase, Q-Former is connected to a frozen image encoder, which is trained to extract text-relevant visual features. This is achieved through tasks like Image-Text Contrastive Learning (ITC), which aligns image and text features in the embedding space; Image-Text Matching (ITM), which verifies if an image-text pair is correctly matched; and Image-Grounded Text Generation (ITG), where Q-Former is trained to generate descriptive text grounded in the visual content. In the second stage, the Q-Former is linked to a frozen LLM to generate natural language text conditioned on visual inputs. This stage exploits a fully connected projection layer to map the outputs of the Q-Former into the input space of the LLM and uses soft visual prompts to guide the text generation. *BLIP-2* has been evaluated across multiple vision-language tasks, demonstrating advancements over previous state-of-the-art models.

The study presented in [43] proposes *M²UGen*, a unified framework that integrates LLMs for multi-modal music understanding and generation. This framework combines multiple LLMs to bridge the gap between understanding and generating music across various modalities, including text, images, and videos. The framework uses several feature encoders to handle diverse input types, such as a music encoder to extract meaningful representations from music data, and image and video encoders to process visual content, transforming it into feature embeddings. These feature embeddings are then aligned through multi-modal adapters, consisting of convolutional layers, linear projections, and dense networks, in order to fuse information from different modalities for complex reasoning tasks. At the core of *M²UGen*, there is a LLM that integrates the processed embeddings at various layers to understand and reason about the input modalities. For music generation tasks, the framework employs specialized music decoders that are guided by conditional embeddings generated from the LLM to produce music aligned with input prompts. *M²UGen* has been evaluated across various music understanding and generation tasks against state-of-the-art models.

In [77], the authors proposed *Video-LLaMA*, a multi-modal framework that combines frozen LLMs and video and audio encoders to enable comprehensive video understanding. The architecture of *Video-LLaMA* is characterized by a dual-branch structure used to process visual and auditory information. In the vision-language branch, a pre-trained visual encoder extracts visual features from video frames, a position embedding layer injects temporal information into video frames, a video Q-Former aggregates these frame-level features into a unified video representation, and a linear projection layer aligns the video embeddings with the LLM's input space.

In the audio-language branch, a pre-trained audio encoder computes features from a short segment of audio, a position embedding layer injects temporal information into audio segments, an audio Q-Former aggregates these audio segment-level features into a unified audio representation, and a linear projection layer that maps audio features into the LLM's embedding space. The processed visual and audio embeddings are then concatenated with textual embeddings as soft prompts to guide the LLMs generation. The framework has been compared against existing models and demonstrated significant advancements in audio-visual understanding.

5.3 Discussion

In this chapter, we have explored how large language models are progressively transforming the life cycle of information systems, enhancing both strategic decision-making and operational efficiency. Rather than simply automating isolated tasks, LLMs are demonstrating their ability to act as intelligent collaborators, bridging gaps between unstructured and structured information, improving reasoning across complex workflows, and enabling a more adaptive approach to system development and maintenance.

From the early stages of planning, where LLMs assist in capturing requirements and synthesizing strategic insights, to conceptual modeling, where they facilitate the translation of informal descriptions into formal system representations, these models are redefining how ISs take shape. Their ability to understand context, generate structured outputs, and iteratively refine results introduces new possibilities for human-AI collaboration in design processes that were traditionally rigid and manual. As systems progress from design to implementation, LLMs have begun to influence architecture definition, not only by assisting in code generation but also by contributing to the formulation of structured development plans that align software design with business objectives. In testing and validation, their role extends beyond error detection, helping anticipate faults, optimize debugging workflows, and increase software resilience through adaptive learning techniques. The final stages of the IS life cycle, monitoring and maintenance, showcase LLMs as intelligent diagnostic tools, capable of identifying system anomalies, detecting security threats, and analyzing vast streams of log data with a level of efficiency and scalability previously unattainable.

Beyond the structured phases of an information system life cycle, LLMs also prove to be valuable assets in complementary areas such as information extraction and data analytics. In information extraction, LLMs demonstrate an impressive ability to process unstructured data from various sources—text, images, and multimedia—transforming them into structured, machine-readable formats. This capability enhances knowledge management, facilitates automation in document-heavy domains, and improves interoperability across different IS components. Meanwhile, in data analytics, LLMs enable sophisticated insights through techniques such as text-to-SQL, text-to-SPARQL, and multi-source reasoning via

autonomous LLM agents. These advancements are critical in decision support systems, real-time analytics, and predictive modeling, allowing IS to move from reactive to proactive intelligence.

Despite these advancements, the integration of LLMs into ISs is still in its early stages, facing critical challenges such as explainability, domain adaptation, and trustworthiness. Ensuring that these models provide reliable, unbiased, and interpretable outputs remains a key concern for real-world adoption. Future research must focus not only on refining LLM performance but also on understanding their limitations, improving their integration into existing IS frameworks, and developing methodologies that ensure human oversight to remain central in critical decision-making processes. Ultimately, this survey highlights the potential of LLMs to reshape how ISs are conceived, built, and maintained. While their capabilities continue to expand, their true impact will depend on how effectively they are integrated into the broader ecosystem of information systems, balancing automation with human expertise to create more intelligent, adaptive, and trustworthy digital infrastructures.

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J.L., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Bińkowski, M.a., Barreira, R., Vinyals, O., Zisserman, A., Simonyan, K.: Flamingo: A visual language model for few-shot learning. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 35, pp. 23716–23736. Curran Associates, Red Hook (2022). https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf
2. Ali, T., Kostakos, P.: Huntgpt: Integrating machine learning-based anomaly detection and explainable ai with large language models (llms). *arXiv preprint arXiv:2309.16021* (2023)
3. Amini, A., Gabriel, S., Lin, P., Koncel-Kedziorski, R., Choi, Y., Hajishirzi, H.: Mathqa: Towards interpretable math word problem solving with operation-based formalisms (2019). <https://arxiv.org/abs/1905.13319>
4. Arora, C., Grundy, J., Abdelrazek, M.: Advancing requirements engineering through generative AI: Assessing the role of LLMS. In: *Generative AI for Effective Software Development*, pp. 129–148. Springer, Berlin (2024)
5. Arteca, E., Harner, S., Pradel, M., Tip, F.: Nessie: automatically testing javascript apis with asynchronous callbacks. In: *Proceedings of the 44th International Conference on Software Engineering, ICSE '22*, pp. 1494–1505. Association for Computing Machinery, New York (2022). <https://doi.org/10.1145/3510003.3510106>
6. Ataei, M., Cheong, H., Grandi, D., Wang, Y., Morris, N., Tessier, A.: Elictron: An llm agent-based simulation framework for design requirements elicitation (2024). <https://arxiv.org/abs/2404.16045>
7. Athiwaratkun, B., Gouda, S.K., Wang, Z., Li, X., Tian, Y., Tan, M., Ahmad, W.U., Wang, S., Sun, Q., Shang, M., Gonugondla, S.K., Ding, H., Kumar, V., Fulton, N., Farahani, A., Jain, S., Giaquinto, R., Qian, H., Ramanathan, M.K., Nallapati, R., Ray, B., Bhatia, P., Sengupta, S., Roth, D., Xiang, B.: Multi-lingual evaluation of code generation models (2023). <https://arxiv.org/abs/2210.14868>

8. Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., Sutton, C.: Program synthesis with large language models (2021). <https://doi.org/10.48550/ARXIV.2108.07732>. <https://arxiv.org/abs/2108.07732>
9. Bavishi, R., Joshi, H., Cambronero, J., Fariha, A., Gulwani, S., Le, V., Radiček, I., Tiwari, A.: Neurosymbolic repair for low-code formula languages. *Proc. ACM Program. Lang.* **6**(OOPSLA2) (2022). <https://doi.org/10.1145/3563327>
10. Berabi, B., He, J., Raychev, V., Vechev, M.: Tfix: Learning to fix coding errors with a text-to-text transformer. In: Meila, M., Zhang, T. (eds.) *Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 139, pp. 780–791. PMLR (2021). <https://proceedings.mlr.press/v139/berabi21a.html>
11. Brugere, I., Sharma, S., Kariyappa, S., Nguyen, A.T., Lecue, F., et al.: Interpretable llm-based table question answering. *arXiv preprint arXiv:2412.12386* (2024)
12. Chen, W., Wang, H., Chen, J., Zhang, Y., Wang, H., Li, S., Zhou, X., Wang, W.Y.: Tabfact: A large-scale dataset for table-based fact verification (2020). <https://arxiv.org/abs/1909.02164>
13. Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H.P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F.P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W.H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A.N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., Zaremba, W.: Evaluating large language models trained on code (2021). <https://arxiv.org/abs/2107.03374>
14. Chen, X., Lin, M., Schärli, N., Zhou, D.: Teaching large language models to self-debug (2023). <https://doi.org/10.48550/ARXIV.2304.05128>. <https://arxiv.org/abs/2304.05128>
15. Conrardy, A.D., Cabot, J.: From image to uml: First results of image based uml diagram generation using llms. *ArXiv abs/2404.11376* (2024). <https://api.semanticscholar.org/CorpusID:269187592>
16. Cámara, J., Troya, J., Burgueño, L., Vallecillo, A.: On the assessment of generative ai in modeling tasks: an experience report with ChatGPT and UML. *Softw. Syst. Model.* **22**(3), 781–793 (2023). <http://doi.org/10.1007/s10270-023-01105-5>
17. Deka, B., Huang, Z., Franzen, C., Hibschan, J., Afergan, D., Li, Y., Nichols, J., Kumar, R.: Rico: A mobile app dataset for building data-driven design applications. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, p. 845–854. ACM (2017). <http://dx.doi.org/10.1145/3126594.3126651>
18. Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., Rass, S.: Pentestgpt: An llm-empowered automatic penetration testing tool (2024). <https://arxiv.org/abs/2308.06782>
19. Dong, Y., Ding, J., Jiang, X., Li, G., Li, Z., Jin, Z.: Codescore: Evaluating code generation by learning code execution (2024). <https://arxiv.org/abs/2301.09043>
20. Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via chatgpt (2024). <https://arxiv.org/abs/2304.07590>
21. Feng, S., Chen, C.: Prompting is all you need: Automated android bug replay with large language models. In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24*. Association for Computing Machinery, New York (2024). <https://doi.org/10.1145/3597503.3608137>
22. Fill, H.G., Fettke, P., Köpke, J.: Conceptual modeling and large language models: Impressions from first experiments with ChatGPT. *Enterp. Modell. Inf. Syst. Architectures* **18**, 1–15 (2023). <https://doi.org/10.18417/EMISA.18.3>. <https://emisa-journal.org/emisa/article/view/318>
23. Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D.: Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6325–6334 (2017). <https://doi.org/10.1109/CVPR.2017.670>

24. Guo, J., Li, J., Li, D., Huat Tiong, A.M., Li, B., Tao, D., Hoi, S.: From images to textual prompts: Zero-shot visual question answering with frozen large language models. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10867–10877 (2023). <https://doi.org/10.1109/CVPR52729.2023.01046>
25. He, M., Jia, T., Duan, C., Cai, H., Li, Y., Huang, G.: LLMeLog: An Approach for Anomaly Detection based on LLM-enriched Log Events. In: 2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE), pp. 132–143. IEEE Computer Society, Los Alamitos (2024). <https://doi.org/10.1109/ISSRE62328.2024.00023>. <https://doi.ieeeecomputersociety.org/10.1109/ISSRE62328.2024.00023>
26. Hong, S., Lin, Y., Liu, B., Liu, B., Wu, B., Zhang, C., Wei, C., Li, D., Chen, J., Zhang, J., Wang, J., Zhang, L., Zhang, L., Yang, M., Zhuge, M., Guo, T., Zhou, T., Tao, W., Tang, X., Lu, X., Zheng, X., Liang, X., Fei, Y., Cheng, Y., Gou, Z., Xu, Z., Wu, C.: Data interpreter: An llm agent for data science (2024). <https://arxiv.org/abs/2402.18679>
27. Hu, X., Zhao, Z., Wei, S., Chai, Z., Ma, Q., Wang, G., Wang, X., Su, J., Xu, J., Zhu, M., et al.: Infiagent-dabench: Evaluating agents on data analysis tasks. arXiv preprint arXiv:2401.05507 (2024)
28. Jiang, X., Dong, Y., Wang, L., Fang, Z., Shang, Q., Li, G., Jin, Z., Jiao, W.: Self-planning code generation with large language models (2024). <https://arxiv.org/abs/2303.06689>
29. Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., He, P., Gu, J., Lyu, M.R.: Lilac: Log parsing using llms with adaptive parsing cache (2024). <https://arxiv.org/abs/2310.01796>
30. Joshi, H., Sanchez, J.C., Gulwani, S., Le, V., Radiček, I., Verbruggen, G.: Repair is nearly generation: Multilingual program repair with llms. In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence, AAAI’23/IAAI’23/EAAI’23. AAAI Press (2023). <https://doi.org/10.1609/aaai.v37i4.25642>
31. Just, R., Jalali, D., Ernst, M.D.: Defects4j: A database of existing faults to enable controlled testing studies for java programs. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis, ISSTA ’14. ACM (2014). <https://doi.org/10.1145/2610384.2628055>
32. Kheddar, H.: Transformers and large language models for efficient intrusion detection systems: A comprehensive survey (2024). <https://arxiv.org/abs/2408.07583>
33. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE 2007-001, Keele University and Durham University Joint Report (2007). https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf
34. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modeling: Can generative ai empower domain experts in creating and redesigning process models? (2023). <https://api.semanticscholar.org/CorpusID:258291461>
35. Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.P.: Process Modeling with Large Language Models, p. 229–244. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-61007-3_18
36. Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al.: Natural questions: a benchmark for question answering research. Trans. Assoc. Comput. Linguist. **7**, 453–466 (2019)
37. Leigh, D.: SWOT analysis. In: Handbook of Improving Performance in the Workplace: Volumes 1–3, pp. 115–140 (2009), Wiley Online Library, Hoboken
38. Li, J., Jia, Z., Zheng, Z.: Semi-automatic data enhancement for document-level relation extraction with distant supervision from large language models. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 5495–5505. Association for Computational Linguistics, Singapore (2023). <https://doi.org/10.18653/v1/2023.emnlp-main.334>. <https://aclanthology.org/2023.emnlp-main.334/>

39. Li, J., Li, D., Xiong, C., Hoi, S.C.H.: Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In: International Conference on Machine Learning (2022). <https://api.semanticscholar.org/CorpusID:246411402>
40. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: Proceedings of the 40th International Conference on Machine Learning, ICML'23. JMLR.org (2023)
41. Li, J., Li, G., Li, Y., Jin, Z.: Structured chain-of-thought prompting for code generation (2023). <https://arxiv.org/abs/2305.06599>
42. Liu, Z., Chen, C., Wang, J., Che, X., Huang, Y., Hu, J., Wang, Q.: Fill in the blank: Context-aware automated text input generation for mobile gui testing. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp. 1355–1367 (2023). <https://doi.org/10.1109/ICSE48619.2023.00119>
43. Liu, S., Hussain, A.S., Wu, Q., Sun, C., Shan, Y.: M²ugen: Multi-modal music understanding and generation with the power of large language models (2024). <https://arxiv.org/abs/2311.11255>
44. Ma, Z., Chen, A.R., Kim, D.J., Chen, T.H., Wang, S.: Llmpraser: An exploratory study on using large language models for log parsing. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24, pp. 1–13. ACM (2024). <https://doi.org/10.1145/3597503.3639150>
45. Marino, K., Rastegari, M., Farhadi, A., Mottaghi, R.: Ok-vqa: A visual question answering benchmark requiring external knowledge. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 3190–3199 (2019). <https://api.semanticscholar.org/CorpusID:173991173>
46. Meskovskis, A., Kenyon, C.: Swot portfolio construction and validation using massive language models. Available at SSRN 4784515 (2024)
47. Mokady, R., Hertz, A.: Clipcap: Clip prefix for image captioning. ArXiv **abs/2111.09734** (2021). <https://api.semanticscholar.org/CorpusID:244346239>
48. Nan, L., Hsieh, C., Mao, Z., Lin, X.V., Verma, N., Zhang, R., Kryściński, W., Schoelkopf, N., Kong, R., Tang, X., Mutuma, M., Rosand, B., Trindade, I., Bandaru, R., Cunningham, J., Xiong, C., Radev, D.: FeTaQA: Free-form table question answering (2021). <https://arxiv.org/abs/2104.00369>
49. Parisi, A., Zhao, Y., Fiedel, N.: Talm: Tool augmented language models. arXiv preprint arXiv:2205.12255 (2022)
50. Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. arXiv preprint arXiv:1508.00305 (2015)
51. Qiao, B., Li, L., Zhang, X., He, S., Kang, Y., Zhang, C., Yang, F., Dong, H., Zhang, J., Wang, L., Ma, M., Zhao, P., Qin, S., Qin, X., Du, C., Xu, Y., Lin, Q., Rajmohan, S., Zhang, D.: Taskweaver: A code-first agent framework (2024). <https://arxiv.org/abs/2311.17541>
52. Rahman, T., Zhu, Y.: Automated user story generation with test case specification using large language model. ArXiv **abs/2404.01558** (2024). <https://api.semanticscholar.org/CorpusID:268857136>
53. Ronanki, K., Berger, C., Horkoff, J.: Investigating chatgpt's potential to assist in requirements elicitation processes (2023). <https://arxiv.org/abs/2307.07381>
54. Roziere, B., Lachaux, M.A., Chantussot, L., Lample, G.: Unsupervised translation of programming languages. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 20601–20611. Curran Associates, Red Hook (2020)
55. Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A.: A systematic survey of prompt engineering in large language models: Techniques and applications. arXiv preprint arXiv:2402.07927 (2024)
56. Sammut-Bonnici, T., Galea, D.: Swot analysis. In: Wiley Encyclopedia of Management, pp. 1–8. Wiley, Hoboken (2015)
57. Schwenk, D., Khandelwal, A., Clark, C., Marino, K., Mottaghi, R.: A-okvqa: A benchmark for visual question answering using world knowledge (2022). <https://arxiv.org/abs/2206.01718>

58. Schäfer, M., Nadi, S., Eghbali, A., Tip, F.: An empirical evaluation of using large language models for automated unit test generation. *IEEE Trans. Softw. Eng.* **50**(1), 85–105 (2024). <https://doi.org/10.1109/TSE.2023.3334955>
59. Shi, F., Fried, D., Ghazvininejad, M., Zettlemoyer, L., Wang, S.I.: Natural language to code translation with execution (2022). <https://doi.org/10.48550/ARXIV.2204.11454>. <https://arxiv.org/abs/2204.11454>
60. Shumway, R.H., Stoffer, D.S., Shumway, R.H., Stoffer, D.S.: Arima models. In: *Time Series Analysis and Its Applications: With R Examples*, pp. 75–163. Springer, Berlin (2017)
61. Stoica, G., Platanios, E.A., Póczos, B.: Re-tacred: Addressing shortcomings of the tacred dataset. *Proc. AAAI Conf. Artif. Intell.* **35**(15), 13843–13850 (2021). <https://doi.org/10.1609/aaai.v35i15.17631>
62. Tan, Q., He, R., Bing, L., Ng, H.T.: Document-level relation extraction with adaptive focal loss and knowledge distillation. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1672–1681. Association for Computational Linguistics, Dublin (2022). <https://doi.org/10.18653/v1/2022.findings-acl.132>. <https://aclanthology.org/2022.findings-acl.132/>
63. Tan, Q., Xu, L., Bing, L., Ng, H.T., Aljunied, S.M.: Revisiting docred – addressing the false negative problem in relation extraction (2022). <https://doi.org/10.48550/ARXIV.2205.12696>. <https://arxiv.org/abs/2205.12696>
64. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147 (2003). <https://aclanthology.org/W03-0419/>
65. Wang, Z., Zhang, H., Li, C.L., Eisenschlos, J.M., Perot, V., Wang, Z., Miculicich, L., Fujii, Y., Shang, J., Lee, C.Y., Pfister, T.: Chain-of-table: Evolving tables in the reasoning chain for table understanding (2024). <https://arxiv.org/abs/2401.04398>
66. Webster, J., Watson, R.T.: Analyzing the past to prepare for the future: Writing a literature review. *MIS Quart.* **26**(2), xiii–xxiii (2002). <http://www.jstor.org/stable/4132319>
67. Wei, Y., Xia, C.S., Zhang, L.: Copiloting the copilots: Fusing large language models with completion engines for automated program repair. In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '23*. ACM (2023). <https://doi.org/10.1145/3611643.3616271>
68. Weischedel, R., Palmer, M., Marcus, M., Hovy, E., Pradhan, S., Ramshaw, L., Xue, N., Taylor, A., Kaufman, J., Franchini, M., El-Bachouti, M., Belvin, R., Houston, A.: Ontonotes release 4.0 (2011). <https://doi.org/10.35111/GFJF-7R50>. <https://catalog.ldc.upenn.edu/LDC2011T03>
69. Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A.H., White, R.W., Burger, D., Wang, C.: Autogen: Enabling next-gen llm applications via multi-agent conversation (2023). <https://arxiv.org/abs/2308.08155>
70. Wu, Y., Roesner, F., Kohno, T., Zhang, N., Iqbal, U.: Secgpt: An execution isolation architecture for llm-based systems. *arXiv preprint arXiv:2403.04960* (2024)
71. Xia, C.S., Zhang, L.: Less training, more repairing please: revisiting automated program repair via zero-shot learning. In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE '22*, p. 959–971. ACM (2022). <https://doi.org/10.1145/3540250.3549101>
72. Xu, J., Cui, Z., Zhao, Y., Zhang, X., He, S., He, P., Li, L., Kang, Y., Lin, Q., Dang, Y., Rajmohan, S., Zhang, D.: Unilog: Automatic logging via llm and in-context learning. In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24*. Association for Computing Machinery, New York (2024). <https://doi.org/10.1145/3597503.3623326>
73. Yang, Z., Gan, Z., Wang, J., Hu, X., Lu, Y., Liu, Z., Wang, L.: An empirical study of GPT-3 for few-shot knowledge-based VQA. *Proc. AAAI Conf. Artif. Intell.* **36**(3), 3081–3089 (2022). <https://doi.org/10.1609/aaai.v36i3.20215>. <https://ojs.aaai.org/index.php/AAAI/article/view/20215>

74. Yasunaga, M., Liang, P.: Break-it-fix-it: Unsupervised learning for program repair. ArXiv **abs/2106.06600** (2021). <https://api.semanticscholar.org/CorpusID:235421942>
75. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., Radev, D.: Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2018). <https://doi.org/10.18653/v1/d18-1425>
76. Zhang, T., Yu, T., Hashimoto, T.B., Lewis, M., Yih, W.T., Fried, D., Wang, S.I.: Coder reviewer reranking for code generation (2022). <https://doi.org/10.48550/ARXIV.2211.16490>. <https://arxiv.org/abs/2211.16490>
77. Zhang, H., Li, X., Bing, L.: Video-llama: An instruction-tuned audio-visual language model for video understanding (2023). <https://arxiv.org/abs/2306.02858>
78. Zhang, R., Li, Y., Ma, Y., Zhou, M., Zou, L.: Llm-aaa: Making large language models as active annotators. In: Findings of the Association for Computational Linguistics: EMNLP 2023. Association for Computational Linguistics, Stroudsburg (2023). <https://doi.org/10.18653/v1/2023.findings-emnlp.872>
79. Zhou, W., Huang, K., Ma, T., Huang, J.: Document-level relation extraction with adaptive thresholding and localized context pooling. In: AAAI Conference on Artificial Intelligence (2020). <https://api.semanticscholar.org/CorpusID:225039888>
80. Zowghi, D., Coulin, C.: Requirements elicitation: A survey of techniques, approaches, and tools. In: Engineering and Managing Software Requirements, pp. 19–46. Springer, Berlin (2005)

Chapter 6

Programming Large Language Models



Marco Calamo  and Jacopo Rossi 

Abstract This chapter provides a comprehensive overview of tools and techniques for programming large language models (LLMs), covering both fundamental and advanced aspects of their implementation. The first section explores different programming languages and frameworks, with a particular focus on Python due to its ecosystem of libraries, such as Hugging Face and LangChain, which facilitate model training, fine-tuning, and deployment. The chapter then presents a practical overview of pre-training and fine-tuning methodologies, discussing approaches like unsupervised and supervised learning, instruction tuning, parameter-efficient fine-tuning, and alignment fine-tuning to optimize LLM performance. In the final section, prompt engineering techniques are examined, including zero-shot, few-shot, and chain-of-thought prompting, demonstrating how structured prompts enhance model reasoning and response accuracy. Through examples and code snippets, the chapter offers a hands-on guide for understanding and exploiting LLMs efficiently, balancing computational resources, and improving task-specific performance.

6.1 Tools and Libraries for Programming LLMs

Large language models (LLMs) are advanced AI models designed to process and generate human-like text. They are trained on vast amounts of data and can be used for a wide variety of tasks, such as chatbots, text summarization, code generation, and many others. Python is the most widely used language in this context, but there are also other languages like TypeScript and JavaScript that support LLMs. The prominence of Python in this field is due to its syntactic simplicity and readability, combined with the availability of a rich ecosystem of libraries and frameworks that facilitate the implementation of machine learning and natural language processing techniques. The ease of writing in Python allows developers to focus more on the use

M. Calamo (✉) · J. Rossi
Sapienza Università di Roma, Rome, Italy
e-mail: calamo@diag.uniroma1.it; j.rossi@diag.uniroma1.it

of models and solving problems, without having to understand complex syntactic details. Python's compatibility with all major LLMs frameworks makes it ideal for simplifying key processes such as training and fine-tuning. Moreover, integration with data management and visualization libraries, such as Pandas, NumPy, and Matplotlib, helps create a complete and interconnected ecosystem that significantly accelerates the development cycle of LLM applications.

In this context, the computing power of GPUs is assuming a crucial role in fully exploiting the potential of LLMs. GPUs allow thousands of operations to be executed simultaneously, which is essential for the intensive calculations required. A key element is the use of the CUDA platform, developed by NVIDIA, which offers specialized libraries and APIs to allow developers to fully exploit the power of GPUs. Thanks to CUDA, it is possible to optimize numerous algorithms used in the training of deep neural networks, ensuring highly efficient performance. GPUs with CUDA are also essential for the deployment of LLMs. In these scenarios, models require an optimal management of computational resources to allow the generation of real-time responses.

Going more into details, in the first part of this chapter, we analyze Python's role in using LLMs, focusing specifically on Hugging Face, a leading platform in the AI ecosystem. Hugging Face provides a large catalogue of pre-trained models, datasets, and training tools, making it easy to use and adapt them. We'll explore how to load pre-trained models, leverage ready-made datasets, and apply fine-tuning techniques to customize models to specific needs. This section will provide practical feedback on how to best leverage the available training and experimentation tools. In the second part, we'll move to LangChain, an open-source framework designed to simplify the orchestration of LLM-based applications. We will explore prompt engineering and strategies to improve the interaction of models with real data sources, optimizing performance and generating contextualized responses.

6.2 Pre-training and Preparing an LLM Model

In this section, we provide a comprehensive step-by-step guide on building a language model from scratch using Hugging Face. This practical implementation of basic LLM concepts offers an overview of the pre-training process for a small language model capable of running on a low-end machine without a dedicated GPU. The minimum requirement for running the pre-training code is approximately 2GB of RAM. However, the quality of the resulting model depends entirely on the amount of training data. For this demo, we use a very small dataset to illustrate the process. While a fully functional model requires trillions of tokens and extensive training time, our example is limited to a few thousand tokens and about one hour of training. To keep the implementation accessible, we use high-level abstractions and state-of-the-art libraries, avoiding in-depth mathematical details related to optimization. First, we demonstrate how to build an LLM from scratch, covering the key steps in the pre-training process. Then, we show how to use an existing LLM as a black box,

Table 6.1 Film reviews with sentiment analysis

| Text | Label |
|--|-------|
| I rented I AM CURIOUS-YELLOW from my video store because of all the controversy .. | 0 neg |
| “I Am Curious: Yellow” is a risible and pretentious steaming pile... | 0 neg |
| If only to avoid making this type of film in the future... | 0 neg |
| This film was probably inspired by Godard’s Masculin, féminin and I urge you to see that film instead... | 0 neg |

leveraging prompt engineering techniques to guide its behavior effectively without modifying the underlying model.

6.2.1 Dataset Loading

The first step in building a language model is preparing the dataset. The most important factor in training an effective language model is having a high-quality dataset. For this example, we use the imdb [13] dataset by Stanford NLP, directly accessed through Hugging Face.¹ The dataset contains selected anonymized reviews from the IMDb platform,² including also if the sentiment of the review was positive (1) or negative (0). A brief snippet is reported in Table 6.1.

The `datasets` library³ simplifies dataset management and loading from Hugging Face for various tasks. To load the IMDB dataset, execute the following command:

Dataset Loading

```

1 from datasets import load_dataset
2 # since the dataset is already split between train, test
3 # and unsupervised, we only load the train split
4 dataset = load_dataset("stanfordnlp/imdb", split='train')
```

To prepare the data for processing by the language model, we need to:

1. Tokenize the plain text (break it into tokens).
2. Pad the tokenized sequences to ensure uniform length.

¹ <https://huggingface.co/datasets/stanfordnlp/imdb>

² <https://www.imdb.com/>

³ <https://huggingface.co/docs/datasets/index>

It is important to use the correct tokenizer for the model; otherwise, it would not be possible to properly encode the text input in the tensor format of GPT-2 and decode the output from GPT-2 tensors to plain text.

Dataset Pre-processing

```

1  from transformers import AutoTokenizer, DataCollatorWithPadding,
   ↪ DataCollatorForLanguageModeling
2
3  # we choose to use pre-trained tokenizer
4  # it will help speed up the pre-training step
5  tokenizer = AutoTokenizer.from_pretrained('gpt2')
6  # setting up the tokenizer for padding support
7  tokenizer.pad_token = tokenizer.eos_token
8
9  # encoding function for tokenizing and padding data
10 def encode(examples):
11     # applying tokenization
12     result = tokenizer(examples['text'],
13                        truncation=True,
14                        padding='max_length')
15     # we set the gold standard for masked text.
16     # During training the LM will try to recreate
17     # token sequence from 'text', comparing it
18     # from 'labels'. For base pre-training they
19     # are the same
20     result["labels"] = result["input_ids"].copy()
21     return result
22
23 dataset = dataset.map(encode, batched=True, remove_columns=["id"])
24 data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

```

Once the dataset is prepared, we can load the model and feed data into it.

6.2.2 Model Loading

For this demonstration, we chose GPT-2 [15] as the base model due to its relatively small number of parameters (about 137M), which allows for training even on consumer machines.⁴

⁴ Note that while the training time will still be significant, following through with our examples will take up to 2 hours on an NVIDIA RTX 3090 with 24 GB of VRAM.

To initialize the model, we use the `transformers` library⁵ by Hugging Face, the state-of-the-art tool for working with LLMs at any scale. Creating an empty GPT-2 model is as simple as typing:

GPT-2 Model Loading

```
1 from transformers import GPT2Config, GPT2LMHeadModel
2 # instantiate the model with default config
3 model = GPT2LMHeadModel(GPT2Config())
4 # print out model details
5 print(model)
```

To be sure that we are looking at the correct model, we print out the model's information and compare it with the original GPT-2 architecture presented in [7], where the blue rectangle represents a gpt block. The final model is composed by 12 of those blocks.

Output

```
GPT2LMHeadModel(
  (transformer): GPT2Model(
    (wte): Embedding(50257, 768)
    (wpe): Embedding(1024, 768)
    (drop): Dropout(p=0.1, inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
        (ln_1): LayerNorm((768,), eps=1e-05,
          elementwise_affine=True)
        (attn): GPT2SdpaAttention(
          ...
        )
        (ln_2): LayerNorm((768,), eps=1e-05,
          elementwise_affine=True)
        (mlp): GPT2MLP(
          ...
        )
      )
    )
    (ln_f): LayerNorm((768,), eps=1e-05,
      elementwise_affine=True)
    )
    (lm_head): Linear(in_features=768,
      out_features=50257, bias=False)
  )
)
```

We also generate some text completion to be sure that the model has actually its weights initialized at random. To generate the text completion from our model, we

⁵ <https://huggingface.co/docs/transformers/index>

use the pipeline abstraction that offers Hugging Face library. It is capable of parsing the output of the language model for us and display the output in natural language.

GPT-2 First Testing

```

1 from transformers import pipeline, set_seed
2 # automated pipeline for text generation from
3 # an huggingface model
4 generator = pipeline('text-generation', model=model,
5   ↪ tokenizer=tokenizer)
6 # setting manual seed for reproducibility
7 set_seed(0)
8 # using the language model to complete the sentence
9 # with the most probable token sequence according
10 # to their weights
11 print(generator("Hello, I'm a language model,", max_length=30,
12   ↪ num_return_sequences=1)[0]['generated_text'])

```

Output

```
"Hello, I'm a language model, azorazororigoigoigoigo..."
```

As we can see, the model currently generates only nonsensical gibberish. We are now ready to train it on plain text to improve its ability to model natural language.

6.2.3 Pre-training

Once the model and dataset are ready, we can begin the training process. In this case, the trainer abstraction provided by the Hugging Face library is straightforward. First, we need to define the training arguments, i.e., the parameters for the optimization process.

Training Arguments

```

1 from transformers import TrainingArguments, Trainer
2 training_args = TrainingArguments(
3   output_dir="llms",
4   learning_rate=2e-5,
5   per_device_train_batch_size=4,
6   per_device_eval_batch_size=4,
7   num_train_epochs=1,
8   weight_decay=0.01,
9   eval_strategy="epoch",
10  save_strategy="epoch",
11  load_best_model_at_end=True,
12  eval_steps=1000,

```

```
13 gradient_accumulation_steps=2,  
14 eval_accumulation_steps=1,  
15 )
```

The chosen parameters are pretty standard for what concerns learning rate and weight decay values, we decided to add some more parameters for reducing the impact on the GPU memory like per-device batch size and accumulation steps, to make this code executable even on GPUs with fewer VRAM available. The training will run for only one epoch, so the results will not be optimal, but it will be enough to demonstrate the capabilities of language modeling of GPT-2.⁶ Finally, we are ready to start our training.

Training

```
1 trainer = Trainer(  
2 model=model,  
3 args=training_args,  
4 train_dataset=dataset,  
5 eval_dataset=dataset,  
6 data_collator=data_collator,  
7 processing_class=tokenizer,  
8 )  
9 trainer.train() # start the trainer
```

This snippet of code will execute the training on the task of generating the next most probable token given the past sequence, i.e., Causal Language Modeling. The train should last about an hour on a modern laptop. Afterward, we can try again the generation from earlier and observe the output:

Output

```
"Hello, I'm a language model, I have been a movie, I have been a movie.  
↪ I have been a movie. I was a movie"
```

We can see that the output is still very far from the model we are used to, but just from a small dataset and a quick training, the model already guessed correctly the English sentence structure and the main argument of the dataset (movies reviews).

In the next sections, we will see how to enhance the plain language model and how to fine-tune it for specific tasks.

⁶ For reference, it took about 400 epochs on a small dataset to a similar model to properly converge: <https://github.com/karpathy/nanoGPT>.

6.3 Fine-Tuning LLMs

In this section, we will go through all the fine-tuning methods of LLMs, and we will implement them starting from (full) pre-trained GPT-2 to test what kind of difference they make. We set the baseline by executing the generating script from Sect. 6.2 on the pre-trained GPT-2.

Output

```
"Hello, I'm a language model, language model. You know what I mean? A  
↪ language model. When I write, I'm like"
```

As we can observe, the model has an understanding of the English language. However, even the full pre-trained version struggles to form a coherent sentence without context. Let's see if when fine-tuning the model the results get more satisfactory.

6.3.1 Unsupervised Fine-Tuning

The first case is unsupervised fine-tuning. It will help our model comprehend a specific lexicon better. In our case, it will be about movies. The code is the same from Sect. 6.2, with one key difference at the model loading stage: in this case, we download from the Hugging Face repositories the pre-trained version.

Unsupervised Fine Tuning Model Loading Code

```
1 model = AutoModelForCausalLM.from_pretrained('gpt2')
```

The output with the fine-tuned model is:

Completion Output

```
"Hello, I'm a language model, and I enjoy speaking Latin as a child. My  
↪ favorite films is Lucio Morricone's (2007, 2003, 2004) and the  
↪ excellent animated short 'The Man Who Wrote Our Mother's Heart.'"
```

We can observe that even if some information is made up (like the title of the movie), the model has shifted its default focus on the movie lexicon, which is exactly what we expected from fine-tuning.

6.3.2 Supervised Fine-Tuning

Regarding supervised fine-tuning, the class used for this type of training will rely on reinforcement learning techniques. In this example, we aim to fine-tune, by training, the full pre-trained GPT-2 on the imdb dataset to perform a sentiment analysis task. This time the trainer will consider the labels assigned to each review, which can be either positive or negative (see Table 6.1). This approach should help improve sentence completion in a specific sentiment compared to unsupervised fine-tuning. The code follows the same patterns as before, but the key difference is that we load the sequence classification version of GPT-2, which includes an additional final softmax layer.

Supervised Fine Tuning Code

```

1  from datasets import load_dataset
2  from transformers import GPT2Tokenizer
3  from transformers import GPT2ForSequenceClassification
4  from transformers import Trainer, TrainingArguments
5
6  # dataset import
7  dataset = load_dataset("imdb")
8
9  # tokenizer loading
10 tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
11 tokenizer.pad_token = tokenizer.eos_token
12
13 # tokenize function
14 def tokenize_function(examples):
15     return tokenizer(examples["text"], padding="max_length",
16         ↪ truncation=True)
17
18 # get dataset ready
19
20 tokenized_datasets = dataset.map(tokenize_function, batched=True)
21
22 # import gpt-2 with a new softmax layer for classification
23 # right now the softmax is initialized at random
24 model = GPT2ForSequenceClassification.from_pretrained("gpt2",
25     num_labels=2)
26 model.config.pad_token_id = model.config.eos_token_id
27
28 # run training
29 training_args = TrainingArguments(
30     output_dir="./results",
31     learning_rate=2e-5,
32     per_device_train_batch_size=10,
33     num_train_epochs=3,
34     weight_decay=0.01,
35 )
36
37 trainer = Trainer(

```

```

36 model=model,
37 args=training_args,
38 train_dataset=tokenized_datasets["train"],
39 eval_dataset=tokenized_datasets["test"],
40 )
41
42 trainer.train()

```

As we can see, the full code is similar to the pre-training one: the dataset loading, the tokenizer loading, and the tokenize function are the same. The key difference is the kind of model that we load: we use the gpt2 for sequence classification abstraction. The huggingface trainer is capable of detecting a training goal that is no longer sequence completion but sequence classification and automatically adapt the training process.

After supervised fine-tuning, the GPT-2 model with a new softmax layer is now capable of classifying the sentiment of the movie reviews. We introduce a new snippet code for generating the desired output for classification:

Classification Output

```

1 def get_sentiment(sentence):
2     inputs = tokenizer(sentence, return_tensors="pt").to('cuda')
3     outputs = model(**inputs)
4     prediction = outputs.logits.argmax(-1).item()
5     return "Positive" if prediction == 1 else "Negative"
6
7 sentence = "I loved this movie!"
8 print(get_sentiment(sentence))

```

We needed a new function called `get_sentiment` to parse the numeric output provided by the softmax and transform them into text. The results are the ones we were expecting:

Sentiment Output

Positive

6.3.3 Instruction Fine-Tuning

To fine-tune a model using Instruction fine-tuning, we need to select the appropriate dataset and apply reinforcement learning. We introduce a new Hugging Face

Table 6.2 Instruction and output examples

| Instruction | Output |
|--|--|
| Create a function that takes a specific input... | <code>def f(x): """Takes a specific input and produces a specific output using any mathematical operators..."""</code> |
| Generate a unique 8-character string that... | <code>import string import random def random_password_string(): characters =...</code> |
| Given a course consisting of 5 lessons, write a... | <code>def display_first_lesson(lessons): print("The first lesson is: " + lessons[0]) Lessons = ["Python..."</code> |
| Create an algorithm to encourage work balance... | <code>One algorithm to encourage work balance and productivity in remote work is a scheduling...</code> |
| Write a JavaScript that changes the text of a... | <code>document.getElementById("myParagraph").innerHTML = "Goodbye, World!";</code> |
| Implement a sorting algorithm which takes an... | <code>def sorting_algorithm(arr): for i in range(len(arr)-1): for j in range(i+1, len(arr)):...</code> |

library called `trl`,⁷ which stands for Transformers Reinforcement Learning. It uses reinforcement learning to train the model to follow instructions. Specifically, we use the `SFTTrainer` class from `trl`, on the `CodeAlpaca20k`⁸ dataset (see Table 6.2). This dataset contains prompt descriptions along with code outputs in different programming languages.

By formatting the dataset into a chat template, even a small model like GPT-2 could learn to have chat like interactions and help with small code generation tasks. The code is:

Instruction Fine Tuning Preparation

```

1 from transformers import AutoModelForCausalLM, AutoTokenizer
2 from datasets import load_dataset
3 from trl import SFTConfig, SFTTrainer, DataCollatorForCompletionOnlyLM
4
5 dataset = load_dataset("lucasmccabe-lmi/CodeAlpaca-20k",
6 split="train")
7 tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
8 tokenizer.pad_token = tokenizer.eos_token
9
10 # helper function to prepare the text from dataset:
11 # it inserts into a chat template
12 def formatting_prompts_func(example):
13     output_texts = []
14     for i in range(len(example['instruction']))):

```

⁷ <https://huggingface.co/docs/trl/index>

⁸ <https://huggingface.co/datasets/lucasmccabe-lmi/CodeAlpaca-20k>

```

15 # add the template for instruction completion
16 # in the form of question and answer,
17 # like a chat
18 text = f"### Question: {example['instruction'][i]}\n
19 ### Answer: {example['output'][i]}"
20 output_texts.append(text)
21 return output_texts
22
23 response_template = " ### Answer:"
24 # preprocess the dataset to add the template
25 collator = DataCollatorForCompletionOnlyLM(response_template,
26 tokenizer=tokenizer)

```

To prepare the dataset for training, we use the *formatting_prompts_func* function that wraps the data entry in the dataset with a standard question-answer template to teach to the model the format of the response it should give when prompted according to the template.

Instruction Fine Tuning Training

```

1 # load the trainer with default parameters
2 trainer = SFTTrainer(
3 model,
4 train_dataset=dataset,
5 args=SFTConfig(output_dir="/tmp"),
6 formatting_func=formatting_prompts_func,
7 data_collator=collator,
8 )
9 # start the training
10 trainer.train()

```

The trainer abstraction is similar to what we have seen before, but we use the supervised fine-tuning trainer that applies reinforcement learning principles to the training.

After the brief instruction, fine-tuning the model is now capable of answering questions in a chat-like style following the template presented before. However, due to a very short training and a small number of parameters, the quality of the answers is still not very satisfactory. Here is an example:

Completion Output

```

### Question:
How can you help me?
### Answer:
You can: A: If you have any questions,
message me on Twitter, [...]

```

6.3.4 Parameter Efficient Fine-Tuning

To load an optimized version of our LLM, we can use another Hugging Face library called `peft`,⁹ which stands for Parameter Efficient Fine-Tuning. It supports several optimization techniques, like LoRa. This library works very well in tandem with other Hugging Face libraries, like `trl`. To run the same training from the previous section using about 33% less GPU memory, you could use this code:

Parameter Optimized Fine Tuning

```
1 from peft import LoraConfig
2
3 # create configuration for parameter optimization
4 peft_config = LoraConfig(
5     # optimization matrix rank
6     r=32,
7     # scaling factor
8     lora_alpha=32,
9     # parameter dropout for lora
10    lora_dropout=0.05,
11    bias="none",
12    task_type="CAUSAL_LM",
13    # tune the embedding layer and prediction head
14    modules_to_save=["wte", "lm_head"]
15)
16
17 # setting up the trainer with default parameters
18 training_args = SFTConfig(output_dir="/tmp",
19    per_device_train_batch_size=2,
20    per_device_eval_batch_size=2,
21    num_train_epochs=2,)
22
23 # fine tuning with direct peft_config
24 trainer = SFTTrainer(
25     'gpt2',
26     train_dataset=dataset,
27     args=training_args,
28     peft_config=peft_config
29 )
30
31 trainer.train()
```

⁹ <https://huggingface.co/docs/peft/index>

To run quantization, we can also load the model using the `bitsandbytes` library.¹⁰ However, as GPT-2 is not supported by that, we use Opt-350M model [21] as an example:

Memory Optimized Fine Tuning

```

1  from transformers import AutoModelForCausalLM, BitsAndBytesConfig
2
3  # loading configuration with 8-bit integer weights
4  quantization_config = BitsAndBytesConfig(load_in_8bit=True)
5
6  # loading the model
7  model_8bit = AutoModelForCausalLM.from_pretrained(
8  "facebook/opt-350m",
9  quantization_config=quantization_config,
10 torch_dtype="auto"
11 )

```

6.3.5 Alignment Fine-Tuning

To demonstrate alignment fine-tuning, we implemented a Direct Preference Optimization (DPO) setup due to its direct support in Hugging Face and its comparable performance. Proper alignment requires selecting a reference dataset. For this demo, we chose the UltraFeedback Binarized dataset¹¹ [4]. As shown in Table 6.3, the dataset includes an instruction, an accepted completion, and a rejected completion, each assigned a score from 1 to 10.

To align fine-tune GPT-2 with the dataset provided, we first need to format the data in a chat-like structure. Once prepared, we can use the `DPOTrainer` from the `trl` library to perform fine-tuning.

DPO Alignment Code

```

1  from datasets import load_dataset
2  from trl import DPOConfig, DPOTrainer
3  from transformers import AutoModelForCausalLM, AutoTokenizer
4
5  model = AutoModelForCausalLM.from_pretrained("gpt2")
6  tokenizer = AutoTokenizer.from_pretrained("gpt2")
7  tokenizer.pad_token = tokenizer.eos_token
8
9  # this time we need an explicit chat template

```

¹⁰ <https://huggingface.co/docs/bitsandbytes/main/index>

¹¹ https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized

```
10 tokenizer.chat_template = """{% for message in messages %}
11 {% if loop.first and messages[0]['role'] != 'system' %}
12 {{ '<|im_start|>system\nYou are a helpful assistant.<|im_end|>\n' }}
13 {% endif %}}{% '<|im_start|>'
14 + message['role'] + '\n' + message['content']
15 + '<|im_end|>' + '\n'}}{% endfor %}
16 {% if add_generation_prompt %}{{ '<|im_start|>assistant\n' }}
17 {% endif %}"""
18
19 train_dataset = load_dataset("trl-lib/ultrafeedback_binarized",
20 split="train")
21
22 training_args = DPOConfig(output_dir="openai-community/gpt2",
23 logging_steps=10,
24 per_device_train_batch_size=2,
25 per_device_eval_batch_size=2,)
26 trainer = DPOTrainer(model=model, args=training_args,
27 processing_class = tokenizer, train_dataset=train_dataset)
28 trainer.train()
```

We apply a standard chat template to the dataset by including it in the tokenizer. The training abstraction is once again similar to the standard trainer, but it applies under the hood the DPO training loop.

Table 6.3 Comparison of instructions, chosen, rejected, and scores

| Instruction | Chosen | Rejected | Score comparison |
|--|--|--|------------------|
| Use the pygame library to write a version of the classic game Snake, with a unique twist | Use the pygame library to write a version of the classic game Snake, with a unique twist | Use the pygame library to write a version of the classic game Snake, with a unique twist | 6 vs 4 |
| She was a horrible pet owner; she would put a what on her cat? Options: leave outside, sharp teeth, get wet | She was a horrible pet owner; she would put a what on her cat? Options: leave outside, sharp teeth, get wet | She was a horrible pet owner; she would put a what on her cat? Options: leave outside, sharp teeth, get wet | 8 vs 7.5 |
| Detailed instructions: In this task, classify if a review is positive/negative, indicating your answer as P or N | Detailed instructions: Classify reviews as positive or negative | Detailed instructions: Classify reviews as positive or negative | 8 vs 4 |
| Write me an in-game shop system in Unity that uses gacha | Write a basic gacha in-game shop system | Write a basic gacha in-game shop system | 8 vs 8 |
| Develop a 10-page research paper on the impact of meditation on mental health | Write about the impact of meditation on mental health | Write about the impact of meditation on mental health | 7 vs 5 |

After alignment tuning, the GPT-2 model is now capable of following a standardized chat template with system, user, and assistant prompt. We can use this snippet of code to generate proper chat completion:

Chat text generation Code

```
1 from transformers import pipeline, set_seed
2
3 generator = pipeline('text-generation',
4 model=model,
5 tokenizer=tokenizer)
6 set_seed(0)
7 chat_history = [
8 {"role": "system", "content": "You are a chatbot"},
9 {"role": "user", "content": "Help me?"},
10 ]
11 outputs = generator(
12 chat_history,
13 max_new_tokens=52,
14 )
15 print(outputs[0]["generated_text"])
```

We observe a behavior similar to that seen after instruction tuning: the model follows the template, but due to the short training time and limited number of parameters, its responses remain unsatisfactory.

Completion Output

```
{'role': 'system', 'content': 'You are a chatbot'},
{'role': 'user', 'content': 'Help me?'},
{'role': 'assistant', 'content': 'Notes:
    This command ignores message-level messages,
    making communication with external tools harder. [...]}'
```

6.4 Prompt Engineering

Prompt engineering is the practice of carefully designing and refining prompts to optimize the responses generated by LLMs [16]. The concept of a *prompt* refers to a set of task-specific instructions designed for an LLM to execute. A well-structured prompt acts as a guide for the LLM, helping to understand the user's intent, handle ambiguity, and focus on providing results that align with the desired goal. The importance of prompts lies in their ability to influence how effectively an LLM performs various tasks. For example, for simple questions, such as asking for a brief explanation or definition, a direct prompt is often sufficient. However, in more complex cases, such as generating detailed reports or solving multi-step

problems, imprecise prompts may result in responses that are vague, incomplete, or unrelated [3]. This is where the distinction between simple prompts and advanced prompts becomes critical. *Simple prompts* are typically concise and straightforward, consisting of basic questions or instructions that require a minimum level of detail. While they are effective for simple tasks, they lack the structure and specificity needed to handle more complex requests. In contrast, *advanced prompts* are more elaborate and often include additional context, examples, constraints, or formatting guidelines. By incorporating these elements, advanced prompts direct with more efficiency the behavior of the model, allowing it to produce responses that are not only precise but also tailored to specific needs or expectations [12].

Prompt engineering is not limited to the simple design and development of prompts but involves a set of skills and techniques that are essential for interacting and working with LLMs. Through prompt engineering, it is not only possible to enhance the performance of an LLM, but there is also the chance to develop new features, such as integrating domain-specific knowledge or using external tools to extend the capabilities of the model [20]. In the following sections, we explore how to use advanced prompt engineering techniques to unlock the full potential of LLMs, enabling them to perform increasingly complex tasks with higher levels of consistency, reliability, and performance.

6.4.1 Zero-Shot Prompting

Zero-shot prompting is a technique that consists of directly asking the model to perform a task by providing only the task description or query. Zero-shot prompting relies entirely on the model's pre-trained knowledge to interpret the task and answer appropriately. This method is quick and efficient and works particularly well for simple tasks or when the model already has sufficient knowledge about the request [10].

Zero-shot excels in handling simple and well-defined tasks, such as explaining a concept, summarizing or translating text, and creative writing. This approach is also ideal to develop LLM-based solutions without requiring extensive setup or examples to perform rapid tests on them. Additionally, zero-shot prompting uses the language model's training data to handle general knowledge queries, providing contextually relevant responses.

Here, are some examples of possible zero-shot prompts:

Explanation Prompt

Explain the concept of gravity as if you were teaching it to a
↪ 10-year-old. Keep the explanation simple and avoid complex terms.

Summarization Prompt

Summarize the following news article in one sentence: 'The city council
↪ has approved a plan to build a new community park, which will
↪ feature walking trails, playgrounds, and picnic areas. The project
↪ is expected to be completed by 2026.'

Translation Prompt

Translate the following sentence from English to French: 'I would like
↪ a cup of coffee, please.' Use formal and polite language.

Creative Writing Prompt

Write a short story of 100 words about a lost dog finding its way home.
↪ Make it heartwarming and suitable for children.

Observing these examples, it is important to underline that to get the most out of zero-shot prompting, the prompt should be:

- Clear: use simple and unambiguous language.
- Specific: clearly define what you want the model to do (e.g., summarize, explain, translate, etc.).
- Contextual: provide any necessary background information or constraints.
- Goal-Oriented: state the objective of the task to help the model focus on the desired result.

The strength of zero-shot prompting lies in its simplicity. Designing a prompt with this technique requires minimal effort and allows one to execute tasks in a rapid way without doing a particular setup. However, limitations include less precision when handling complex or ambiguous tasks and the potential for misinterpretation due to the absence of task examples or demonstrations.

6.4.2 Few-Shot Prompting

Few-shot learning is an approach that can be adopted to perform specific tasks. Unlike zero-shot prompting, which relies solely on the responses generated by the model without providing specific guidelines, it uses examples to help the model identify patterns and relationships [17]. This approach is particularly useful when it is not possible to collect large amounts of labeled data, such as rare events, specialized domains, or rapidly changing contexts. The adoption of techniques based on few-shot learning is a key element for the effective use of LLMs. By structuring data in a consistent format, models can be configured to obtain

appropriate answers to specific tasks, reducing the need for large amounts of training data. Few-shot learning can be described as a methodology that exploits the knowledge of the model in a more targeted way. In particular, it allows obtaining more structured answers, with less variability and greater consistency, improving the ability of the model to solve specific problems. Studies confirm that few-shot learning is a highly effective strategy [2], especially when using structured prompts. The use of carefully selected examples improves the performance of the model and proves to be useful in scenarios characterized by limited availability of data.

To create an effective few-shot prompt, we can identify the following steps:

- Define the task: specify the task or objective that the LLM is expected to perform. It is very important to define the type of input the LLM will process and the desired output format.
- Choose examples: select examples that effectively represent how to solve the task. These examples should illustrate the desired input-output mapping with clarity and reflect the diversity of the data distribution to ensure robustness.
- Organize the prompt: structure the prompt in a coherent and logical order. Examples are typically associated with the task of assisting the response generation. They should be formatted consistently and arranged to highlight their relevance to the prompt.
- Test and refine: once an initial prompt is defined, it is critical to evaluate its effectiveness. This involves analyzing the outputs' accuracy and consistency with the goals of the task. Based on these evaluations, it is possible to refine the prompt by adjusting examples, reformulating the query, or changing the overall structure. Iterative testing allows for the identification of issues, such as biases introduced by example selection or misinterpretations of the task by the LLM.

Here there are some examples of few-shot prompting:

Explanation

Explain concepts in simple terms for a 10-year-old.

These are some examples of how to perform the task:

Topic: Electricity

Explanation: Electricity is like magic energy that flows through wires

- ↪ to make things like lights and TVs work. It travels super fast and
- ↪ powers up everything around us.

Topic: Photosynthesis

Explanation: Photosynthesis is how plants make their food. They use

- ↪ sunlight, water, and air to create energy, just like how we eat
- ↪ food to stay strong.

Topic: Gravity

Explanation: Gravity is like an invisible hand that pulls everything
↳ down to the ground. It's why we don't float away and why things
↳ fall when you drop them. Even the moon stays near Earth because of
↳ gravity.

Explain the following concept:

Topic: The Water Cycle

Explanation:

Named Entity Recognition

Extract entities from the sentences and classify them.

These are some examples of how to perform the task:

Sentence: "Apple's headquarters are in Cupertino."

Entities: [Apple: Organization, Cupertino: Location]

Sentence: "Google was founded in California."

Entities: [Google: Organization, California: Location]

Sentence: "Elon Musk is the CEO of Tesla."

Entities: [Elon Musk: Person, Tesla: Organization]

Sentence: "The Eiffel Tower is located in Paris."

Entities: [Eiffel Tower: Location, Paris: Location]

Sentence: "Jeff Bezos founded Amazon in 1994."

Entities: [Jeff Bezos: Person, Amazon: Organization]

Extract entities and classify them from the following sentence:

Sentence: "Tony Effe was born in Rome"

Entities:

Topic Classification

Classify sentences into categories: Sports, Technology, or
↔ Entertainment.

These are some examples of how to perform the task:

Sentence: "Cristiano Ronaldo scored the winning goal in the match."
Category: Sports

Sentence: "Apple just launched its latest iPhone model."
Category: Technology

Sentence: "The new Marvel movie is breaking box office records."
Category: Entertainment

Sentence: "The Lakers won their third consecutive game last night."
Category: Sports

Sentence: "Samsung unveiled its new foldable smartphone at the event."
Category: Technology

Sentence: "Taylor Swift's latest album topped the charts this week."
Category: Entertainment

Classify the following sentence into one of the predefined categories:

Sentence: "The streaming events of Blur attract thousands of users."
Category:

To ensure effective communication and clarity in presenting examples for a task, it is essential to focus on including examples that provide a concrete demonstration of what the result should be. Maintaining a uniform structure across examples reinforces the pattern to highlight. This consistency not only aids comprehension but also helps the LLM internalize the structure, improving its ability to generalize and apply the pattern in different situations [14]. Finally, even if it is important to provide examples with enough detail, it is important to not use overly lengthy examples that can lead to exceeding the token limit.

6.4.3 Chain-of-Thought Prompting

Chain-of-thought (CoT) is a reasoning process that allows to tackle complex problems by breaking them down into a sequence of simpler and more understandable steps [19]. This approach reflects how humans typically approach a problem, placing particular emphasis on the intermediate steps of reasoning. In practice, it improves the ability to manage complex tasks by highlighting the value of process-oriented

thinking and the ability to connect steps following rigorous logic. Unlike traditional prompting techniques, where you simply provide some inputs and their outputs, CoT allows to structure the prompt in a way that encourages the LLM to articulate the intermediate steps to improve the accuracy of the results.

CoT can be combined with few-shot prompting techniques, where the LLM receives examples that include inputs and outputs accompanied by the division of the task into intermediate steps. These examples help the model understand how to perform the list of intermediate steps to achieve the final result.

Here we present a prompt with a possible application of CoT:

Pairwise Product Check

Determine whether a number is the product of any two numbers in a given
↪ list.

These are some examples of how to perform the task:

Task: Is 15 the product of any two numbers in the list [4, 8, 9, 12,
↪ 2]?

Reasoning:

Check all pairwise products:

$4 \times 8 = 32$ (Not 15)

$4 \times 9 = 36$ (Not 15)

$4 \times 12 = 48$ (Not 15)

$4 \times 2 = 8$ (Not 15)

$8 \times 9 = 72$ (Not 15)

$8 \times 12 = 96$ (Not 15)

$8 \times 2 = 16$ (Not 15)

$9 \times 12 = 108$ (Not 15)

$9 \times 2 = 18$ (Not 15)

$12 \times 2 = 24$ (Not 15)

None of the pairwise products equal 15.

Answer: False.

Task: Is 45 the product of any two numbers in the list [6, 4, 9, 2, 8]?

Reasoning:

Check all pairwise products:

$6 \times 4 = 24$ (Not 45)

$6 \times 9 = 54$ (Not 45)

$6 \times 2 = 12$ (Not 45)

$6 \times 8 = 48$ (Not 45)

$4 \times 9 = 36$ (Not 45)

$4 \times 2 = 8$ (Not 45)

$4 \times 8 = 32$ (Not 45)

$9 \times 2 = 18$ (Not 45)

$9 \times 8 = 72$ (Not 45)

$2 \times 8 = 16$ (Not 45)

None of the pairwise products equal 45.

Answer: False.

Task: Is 72 the product of any two numbers in the list [6, 4, 9, 12, ↪ 2]?

Reasoning:

Check all pairwise products:

$6 \times 4 = 24$ (Not 72)

$6 \times 9 = 54$ (Not 72)

$6 \times 12 = 72$ (Matches!)

A pair exists (6×12) that equals 72.

Answer: True.

Solve the following problem.

Task: Is 20 the product of any two numbers in the list [10, 7, 13, 5, ↪ 3]?

Answer:

CoT has been described mainly in combination with few-shot learning. Alternatively, we can highlight some innovative approaches, such as the one introduced in [11], which is notable for its use of zero-shot prompting. This method involves adding the sentence *Let's think step by step* at the end of the input prompt. This change, although minimal, has shown a significant impact, highlighting how small changes in the prompt can improve the performance of models.

Another relevant technique is Auto-CoT [22], which automates the generation of demonstrations by implementing step-by-step reasoning with LLMs. The process begins with a question-clustering phase, where questions are grouped into clusters using vector representations and k-means clustering. Next, in the demonstration sampling stage, a representative question from each cluster is selected based on its proximity to the cluster center. This representative question is then fed into an LLM, which, using the Zero-Shot-CoT approach [11], generates a reasoning chain to generate the result.

An evolution of the CoT technique is represented by the approach introduced in the work on Self-Consistency [18]. This method is based on the idea that complex problems can be solved by following different reasoning paths. Comparing these paths, it is possible to identify the most effective one, that is, the one capable of producing a more accurate solution. In Self-Consistency, first, a CoT prompt is defined and designed with manually written chain-of-thought exemplars [19]. Then, there is a generation of diverse set of reasoning paths by sampling multiple candidate outputs from the model's decoder. This can be achieved by running the model on the same problem multiple times, using sampling techniques such as temperature sampling [1, 6], top-k sampling [5, 8, 15], and nucleus sampling [9]. Finally, the reasoning paths are analyzed using aggregation metrics, and there is a selection of the most consistent answer among the generated ones.

6.4.4 Prompt Engineering in Action

In this section, we describe the general procedures for invoking LLMs using the prompt engineering techniques described above. We use the LangChain library to simplify the interaction with LLMs and integrate models hosted on the Hugging Face platform. The code is also easily adaptable for loading and using local models, ensuring greater flexibility in model deployment.

Zero-Shot, Few-Shot, Chain of Thought

```

1  from langchain_huggingface import HuggingFaceEndpoint
2  from langchain_core.prompts import ChatPromptTemplate
3  from langchain_core.output_parsers import StrOutputParser
4  import os
5
6  # Set up Hugging Face API token from environment variables
7  os.environ["HUGGINGFACEHUB_API_TOKEN"] = HUGGINGFACEHUB_API_TOKEN
8  HUGGINGFACEHUB_API_TOKEN = os.environ["HUGGINGFACEHUB_API_TOKEN"]
9
10 # Define the prompt (updated with the actual prompt text)
11 prompt = "prompt description..."
12
13 # Create a ChatPromptTemplate with a user message
14 prompt_template = ChatPromptTemplate.from_messages([
15     ('user', prompt) # Define the user message as part of the prompt
16     ↳ template
17 ])
18
19 # define the Hugging Face model endpoint
20 model = HuggingFaceEndpoint(
21     repo_id="mistralai/Mistral-7B-Instruct-v0.2", # Model ID for the
22     ↳ Hugging Face endpoint
23     temperature=0.5, # Controls randomness in responses
24     huggingfacehub_api_token=HUGGINGFACEHUB_API_TOKEN, # API token for
25     ↳ authentication
26 )
27
28 # Output parser to convert the model response into a string
29 parser = StrOutputParser()
30
31 # Function to execute the prompt and return the response
32 def ask_question():
33     # Set up the chain of operations: prompt template -> model -> output
34     ↳ parser
35     chain = prompt_template | model | parser
36
37     # Execute the chain and return the result
38     response = chain.invoke({})
39     return response
40
41 # Run the script if executed directly

```

```

38 if __name__ == "__main__":
39     try:
40         result = ask_question()
41         print("Result:", result)
42     except Exception as e:
43         print("Error:", e) # Handle and print any errors that occur during
                               ↪ execution

```

Analyzing the code in details,

Prompt definition

```

1  % prompt_template = "prompt description..."
2  % prompt_template = ChatPromptTemplate.from_messages([
3  %      ('user', prompt_template)
4  %  ])
5  %

```

in lines 11–16, the prompt is structured using `ChatPromptTemplate`, which organizes the interaction as a series of messages. In this case, the message is attributed to the ('user'). By modifying the content of the prompt, it is possible to customize the request to suit different applications. This ability to configure the prompt allows for the application of various prompt engineering techniques. Indeed, by structuring the prompt at line 11 according to the guidelines outlined in the previous sections, these techniques can be implemented effectively.

Model Integration

```

1  % model = HuggingFaceEndpoint(
2  %      repo_id="mistralai/Mistral-7B-Instruct-v0.2",
3  %      temperature=0.5,
4  %      huggingfacehub_api_token=HUGGINGFACEHUB_API_TOKEN,
5  %  )
6  %

```

Lines 19 to 23 explain how to invoke the `Mistral-7B-Instruct-v0.2` model, which is hosted on the Hugging Face repository. It is important to note that the same structure can be used to load a local model, such as a fine-tuned model designed for specific tasks, like the one described in this chapter. The configuration of the temperature parameter allows you to control the randomness of the model's responses, with lower values producing more deterministic responses. The `HUGGINGFACEHUB_API_TOKEN` variable allows to authenticate to the Hugging Face service to access the models available.

Output Parser

```
1 % parser = StrOutputParser()
2 %
```

The parser, at line 26, transforms the model output into a readable string, making it easier to interpret the results.

Code Execution

```
1 % def ask_question():
2 %     chain = prompt_template | model | parser
3 %     response = chain.invoke({})
4 %     return response
5 %
```

The `ask_question()` function, lines 29 to 35, set up the operation to invoke the LLM by defining the model, the prompt, and the parser.

As we can observe, the code structure is a practical and flexible example of how to use LLMs. The basic configuration can be easily adapted to use different models, both remote and local, and to perform advanced prompt engineering techniques, through customizing prompts. This implementation is not only a guide to invoke LLMs but also represents a methodological basis that can be extended to cover multiple use case applications.

References

1. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for boltzmann machines. *Cognit. Sci.* **9**(1), 147–169 (1985)
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **33**, 1877–1901 (2020)
3. Chen, B., Zhang, Z., Langrené, N., Zhu, S.: Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735* (2023)
4. Cui, G., Yuan, L., Ding, N., Yao, G., Zhu, W., Ni, Y., Xie, G., Liu, Z., Sun, M.: Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377* (2023)
5. Fan, A., Lewis, M., Dauphin, Y.: Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833* (2018)
6. Fidler, J., Goldberg, Y.: Controlling linguistic style aspects in neural language generation. *arXiv preprint arXiv:1707.02633* (2017)
7. Heilbron, M., Ehinger, B., Hagoort, P., De Lange, F.P.: Tracking naturalistic linguistic predictions with deep neural language models. *arXiv preprint arXiv:1909.04400* (2019)
8. Holtzman, A., Buys, J., Forbes, M., Bosselut, A., Golub, D., Choi, Y.: Learning to write with cooperative discriminators. *arXiv preprint arXiv:1805.06087* (2018)
9. Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* (2019)

10. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) *Advances in Neural Information Processing Systems*, vol. 35, pp. 22199–22213. Curran Associates, Newry (2022)
11. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. *Adv. Neural Inf. Process. Syst.* **35**, 22199–22213 (2022)
12. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surveys* **55**(9), 1–35 (2023)
13. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150. Association for Computational Linguistics, Portland, Oregon (2011). <http://www.aclweb.org/anthology/P11-1015>
14. Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., Zettlemoyer, L.: Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837* (2022)
15. Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., Sutskever, I.: Better language models and their implications. *OpenAI blog* **1**(2) (2019). <https://openai.com/index/better-language-models/>
16. Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A.: A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927* (2024)
17. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023)
18. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022)
19. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **35**, 24824–24837 (2022)
20. Yong, G., Jeon, K., Gil, D., Lee, G.: Prompt engineering for zero-shot and few-shot defect detection and classification using a visual-language pretrained model. *Comput.-Aided Civil Infrastruct. Eng.* **38**(11), 1536–1554 (2023)
21. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., et al.: Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022)
22. Zhang, Z., Zhang, A., Li, M., Smola, A.: Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493* (2022)

Chapter 7

Retrieval-Augmented Generation



Filippo Bianchini 

Abstract Retrieval-augmented generation (RAG) has emerged as a transformative approach to enhancing the capabilities of large language models (LLMs) by integrating real-time information retrieval with generative text synthesis. Traditional LLMs rely on pre-trained data, limiting their effectiveness when handling tasks requiring up-to-date or domain-specific information. RAG addresses this constraint by embedding a retrieval mechanism within the LLM framework, allowing the model to access external knowledge sources dynamically. This integration enables LLMs to generate contextually relevant and factually accurate responses, extending their utility within the realm of information systems (IS), where timely and precise information is critical. Starting from the history and evolution of retrieval-augmented generation, passing through different framework and methodologies, and moving to practical applications by different libraries, in this chapter, we offer a theoretical and practical overview of this innovative and powerful technique.

7.1 Introduction

The origins of retrieval-augmented generation (RAG) trace back to the early developments in question-answering systems [5] from the 1970s, when researchers in information retrieval began building systems capable of using natural language processing (NLP) to retrieve answers from text collections. Initially, these systems were confined to narrow domains, but they laid the groundwork for modern techniques by enabling machines to understand and process human questions. An example from the mid-1990s is *Ask Jeeves*¹ [17], which introduced a conversational,

¹ *Ask Jeeves* was a search engine where users could ask questions in natural language: <https://www.ask.com/>.

F. Bianchini (✉)
Sapienza Università di Roma, Rome, Italy
e-mail: bianchini@diag.uniroma1.it

user-friendly question-answering service with a virtual assistant mascot. This helped popularize the idea that users could ask natural language questions and receive relevant answers, an important step toward modern information retrieval systems [12].

A significant advancement in AI-driven information retrieval occurred in 2011 when IBM's Watson [9] demonstrated the potential of advanced AI by defeating human champions on the quiz show *Jeopardy!*.² Unlike earlier question-answering systems that relied solely on predefined databases, Watson leveraged NLP and sophisticated retrieval and ranking algorithms to analyze vast amounts of unstructured text in real time.

As AI research progressed, the limitations of purely generative models became increasingly evident, particularly their tendency to produce factually incorrect or unverifiable information. Despite steady advancements in NLP, the fundamental principles of text mining and information retrieval remained critical for ensuring factual consistency. The challenge, therefore, was to develop systems that could generate humanlike responses while grounding their output in verifiable, up-to-date sources. In this context, another important milestone for RAG is the development of the transformer architecture [29], a fundamental advancement that enabled language models to integrate additional knowledge through pre-trained models (PTMs) [3, 19]. This progression and the demand for more reliable AI-generated content led Patrick Lewis to introduce RAG in a 2020 paper [19], marking a key evolution in generative AI. His work proposed a novel framework that seamlessly combines a retrieval mechanism with a language model to enhance the handling of knowledge-intensive NLP tasks. Specifically, RAG integrates two key components: a *retriever*, which fetches relevant documents from an external knowledge base, and a *generator*, which synthesizes responses based on the retrieved information. In an interview, Lewis expressed regret over the unflattering acronym, acknowledging that they would have chosen a more appealing name had they anticipated the widespread impact of their work. RAG has since evolved into a recognized framework that encompasses a growing array of methods utilized in numerous academic papers and commercial applications. An overview of Lewis' approach is presented in Fig. 7.1.

This hybrid approach bridges the gap between traditional retrieval-based systems and modern generative models, improving both factual accuracy and contextual relevance and prepared the way for modern RAG technologies, as we know it today, which enhance LLMs by integrating real-time external data for greater accuracy and relevance. Figure 7.2 illustrates the rapid evolution and expansion of RAG techniques, from the release of GPT-3 in 2020, through the launch of ChatGPT, to the introduction of GPT-4.

These models have transformed numerous tasks such as summarizing text, answering questions, and creative writing. The reach of LLMs is now pervasive,

² A popular American quiz show where contestants respond to general knowledge clues with questions, aiming to accumulate the most money by the end of each round: <https://www.jeopardy.com/>.

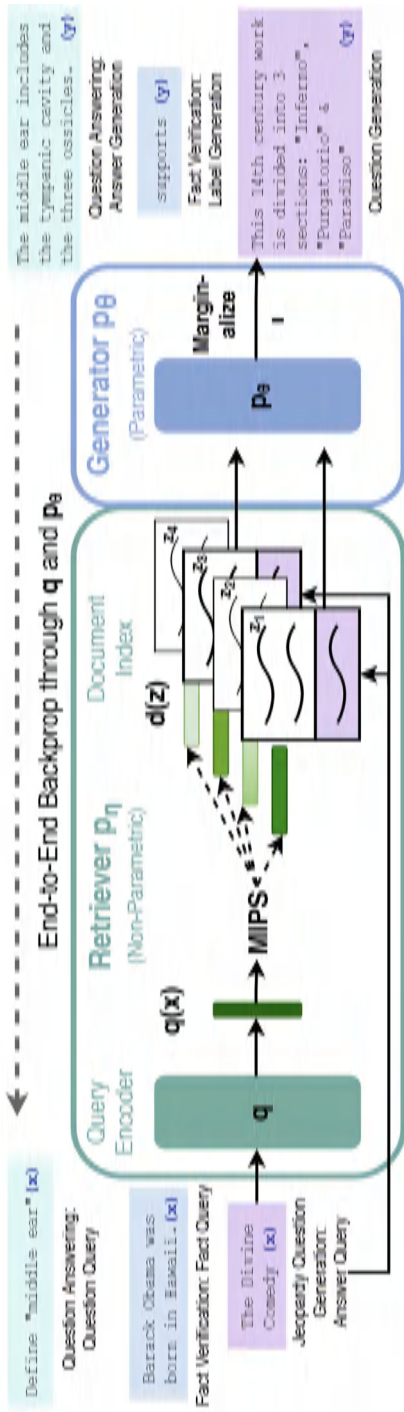


Fig. 7.1 Overview of Lewis' approach for RAG [19]

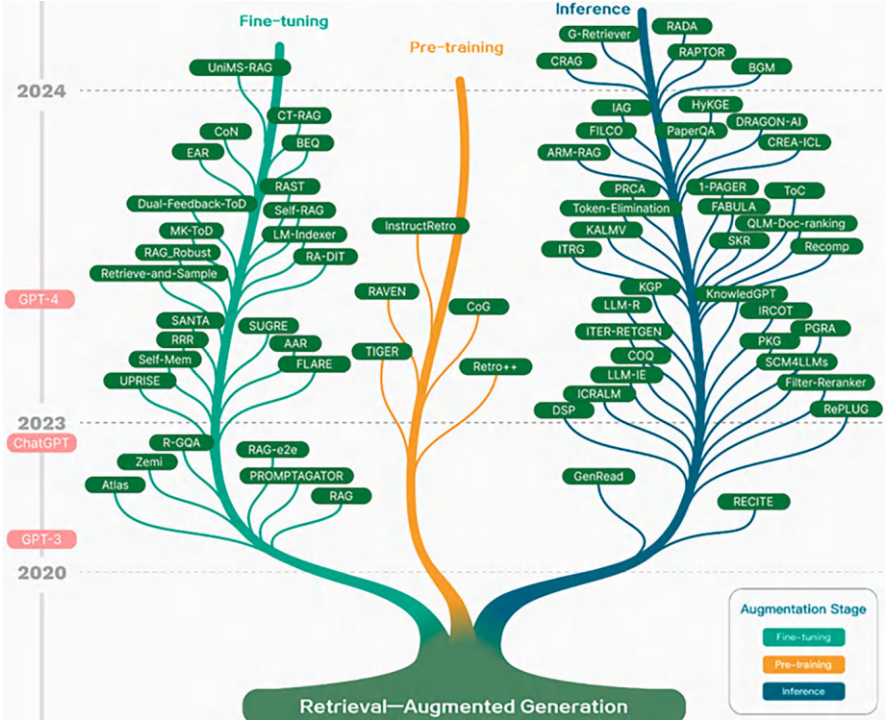


Fig. 7.2 Technology tree of RAG research [10]

extending beyond expert users to a wider audience. As their adoption grows, so does our understanding of their strengths and limitations. A key issue that has surfaced is the tendency of LLMs to generate responses that, while coherent and convincing, may contain information that is partially or entirely incorrect. This phenomenon, often referred to as *hallucination* in the AI community, poses a significant challenge in high-stakes environments. Hallucination occurs due to the probabilistic nature of LLMs, which generate text based on probability rather than fact-checking [30]. As a result, LLMs sometimes *fill in* information in a way that sounds plausible but is factually wrong [16]. This design enables impressive fluency and flexibility but lacks an internal verification mechanism to cross-check factual accuracy.

When an LLM powered by RAG receives a prompt or question, it first consults an external information source, which could be a search engine, a database, or another structured repository. The retrieval component finds relevant documents, passages, or data points based on the prompt. Instead of generating an answer solely from its pre-trained language patterns, the LLM uses this retrieved information to guide and inform its response. The result is a hybrid output: the natural, conversational style of an LLM response combined with real-time data, significantly improving factual accuracy. With these advancements, RAG research shifted toward enhancing LLMs’ ability to handle complex, information-rich queries during the inference stage, leading to accelerated progress in RAG methodologies.

7.2 RAG: Working Logic

The typical RAG framework combines a retrieval mechanism with generative language models to enhance the quality and accuracy of responses in natural language processing tasks. The functioning involves a two-step process: *retrieving* and *generation*. Before these steps are executed, a preliminary phase takes place. This phase is essential for preparing and organizing the data, ensuring effective querying by the retrieval system.

7.2.1 Preliminary Step

The core workflow of RAG begins with several preparatory steps, including data loading, parsing, embedding, and indexing. These steps ensure that external knowledge sources can be efficiently queried by the retrieval mechanism.

7.2.1.1 Data Loading and Parsing

The initial step in the RAG system involves loading the necessary data, which may originate from various sources and exist in different formats, such as PDF, HTML, Word, Markdown, or database exports. These various formats require processing to ensure compatibility with the retrieval system. Once loaded, the raw text undergoes a critical parsing phase, essential for achieving uniformity and ensuring clean, structured formatting suitable for indexing and querying. Parsing encompasses several key operations:

- Removing unnecessary metadata and encoding artifacts that might interfere with efficient processing
- Tokenizing the text into smaller, manageable segments to facilitate easier indexing and retrieval
- Splitting large documents into smaller, contextually meaningful chunks that preserve semantic coherence while optimizing retrieval performance

The goal of parsing is to prepare the raw data for transformation into a suitable embedding space and to ensure efficient indexing for similarity searches.

7.2.1.2 Embedding

The next stage in the RAG system is converting the cleaned and preprocessed text into numerical representations using embedding models. Embeddings are vector representations of the text chunks and play a central role during the retrieval phase by enabling the system to compute semantic similarity between user queries and

indexed data [4]. Essentially, embeddings map text into a high-dimensional vector space, capturing their contextual and semantic meaning in a way that facilitates efficient comparison and matching.

Common embedding methods include a variety of pre-trained models that have proven effective in encoding semantic relationships. Among the most widely used methods are:

- **Word2Vec:** a model introduced by Mikolov et al. [21] that represents words as dense vectors³ by analyzing their co-occurrence in large corpora. Word2Vec embeddings capture word similarity based on their contextual usage.
- **GloVe:** Global Vectors for Word Representation was developed by Pennington et al. [24]. GloVe embeddings are trained using global word co-occurrence statistics over large datasets, effectively combining global and local statistics for semantic modeling.
- **BERT-based models:** the most prominent BERT-based models for embedding generation are Sentence-BERT (SBERT) [25] and RoBERTa [20], both optimized extensions of the original BERT⁴ architecture [7]. Both improve BERT's capability for similarity tasks. This adaptation significantly boosts their performance in applications such as semantic search, question answering, and clustering.
- **Transformer-based models:** modern models like `text-embedding-3-small` or `text-embedding-3-large` by OpenAI,⁵ `NV-Embed-v2`⁶ by NVIDIA [18], or `meta-llama/LLaMA-3.1` by Meta⁷ are highly effective embeddings due to their attention mechanisms [29] and pre-training on massive corpora.

These embeddings serve as the foundation for retrieval, as they allow the system to represent text chunks numerically, transforming semantic relationships into mathematical comparisons.

7.2.1.3 Indexing

After generating embeddings, the next step is indexing these vector representations into a system optimized for efficient retrieval. Indexing organizes the vectors in a structured and searchable format, enabling the system to swiftly match incoming

³ A dense vector is a numerical representation of data where most or all dimensions contain non-zero values.

⁴ Bidirectional Encoder Representations from Transformers.

⁵ <https://platform.openai.com/docs/guides/embeddings/embedding-models>.

⁶ <https://huggingface.co/nvidia/NV-Embed-v2>.

⁷ <https://huggingface.co/meta-llama>.

queries with stored data. This process typically involves selecting between *vector libraries* and *vector databases*, each tailored to specific use cases:

- **Vector libraries:** these systems are primarily used for static datasets where the indexed data remains immutable. Vector libraries focus solely on storing vector embeddings, without preserving the associated objects they were generated from. This approach means they lack support for CRUD (Create, Read, Update, Delete) operations. Examples of popular vector libraries include FAISS,⁸ which is widely adopted for its efficiency in static and large-scale datasets; Annoy,⁹ developed by Spotify and quite similar to FAISS since it is designed to work with static data; or ScaNN¹⁰ by Google, designed for fast approximate nearest neighbor searches in high-dimensional spaces.
- **Vector database:** designed for dynamic and scalable applications, dedicated vector databases offer native support for vector operations and the associated data objects. These systems enable advanced functionalities such as CRUD operations, metadata integration, and real-time updates. Examples include Pinecone,¹¹ Weaviate,¹² and Milvus.¹³ Such databases are particularly suitable for use cases requiring frequent updates or where contextual information must accompany the vectors for refined retrieval.

Efficient indexing ensures that the retrieval system can quickly compute similarity scores between incoming queries and stored vectors. The choice between a vector library and a vector database depends on the application's requirements, such as data mutability, scalability, and the need for enhanced metadata handling.

7.2.2 Retrieving Phase

Once embeddings are indexed, the retrieval mechanism is activated during the query phase, which is central to the RAG framework. This process involves transforming a user query into a format that can be effectively matched against the indexed embeddings to retrieve relevant context passages. Here's a detailed explanation of the retrieval phase:

- **Query encoding:** when a user issues a query, it is first encoded using the same embedding model that was used during the indexing phase. This ensures compatibility between the query's vector representation and the indexed embeddings. The

⁸ Facebook AI Similarity Search: <https://github.com/facebookresearch/faiss/wiki>.

⁹ Approximate Nearest Neighbors Oh Yeah: <https://github.com/spotify/annoy>.

¹⁰ Scalable Nearest Neighbors: <https://github.com/google-research/google-research/tree/master/scann>.

¹¹ <https://github.com/pinecone-io>.

¹² <https://github.com/weaviate/weaviate>.

¹³ <https://github.com/milvus-io/milvus>.

encoding step transforms the query into a dense vector that captures its semantic meaning.

- **Similarity computation:** the encoded query vector is compared against the stored embeddings in the vector database or library. This comparison is performed using similarity metrics, such as:
 - *Cosine similarity*, which measures the cosine of the angle between two vectors in multidimensional space
 - *Euclidean distance*, a measure for the straight-line distance between two vectors in space, suitable for certain use cases where magnitude¹⁴ is important
 - *Dot product*, which computes the projection of one vector onto another, often optimized for speed in high-dimensional spaces
- **Result ranking:** the similarity scores computed in the previous step are used to rank the indexed embeddings. The system identifies the top- K embeddings, those with the highest similarity scores, corresponding to the most relevant chunks of data.
- **Passage retrieval:** the top-ranked embeddings are then mapped back to their original text chunks (passages). Depending on the indexing system, additional metadata (e.g., document IDs, timestamps, or source information) can also be retrieved to enrich the results.

7.2.3 Generation Phase

After the retrieval process, the selected context (retrieved passages) is combined with the user query to form a prompt for the generative language model. This prompt serves as the input to the LLM, conditioning it on both the user query and retrieved knowledge. The model can then generate a response that incorporates this external context. The model's response is influenced by task-specific factors. It can use either (1) *parametric knowledge*, namely, inherent general knowledge from the model's pre-trained parameters, or (2) *non-parametric information*, which is information explicitly retrieved from external context. Figure 7.3 shows how the typical RAG framework works.

¹⁴ *Magnitude* is the length of a dense vector, measuring its scale in the embedding space. It is crucial for normalization and similarity metrics, ensuring accurate matching between query vectors and stored data.

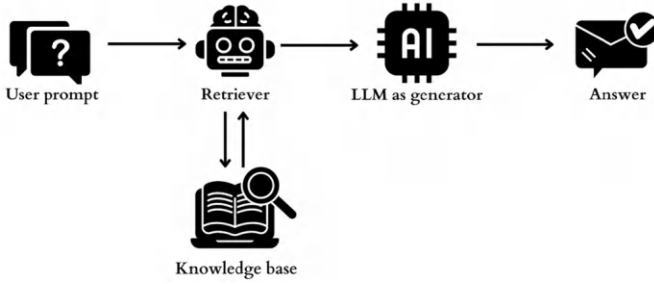


Fig. 7.3 Typical RAG framework

7.2.4 RAG Classification

While the RAG paradigm is continuously evolving, researchers have proposed several categorizations based on multiple key components. Gao et al. [10] present a valuable classification that divides RAG into three main framework: *Naive RAG*, *Advanced RAG*, and *Modular RAG*. Additionally, RAG can be categorized based on the methodology employed by the retriever during the retrieval phase. Tonmoy et al. [27] propose three distinct RAG methodologies, specifically focusing on whether the retrieval occurs *before*, *during*, or *after* output generation. Zhao et al. [32] further extend this classification by categorizing RAG into four classes based on the retriever’s methodology to augment the generator: *Query-based RAG*, *Latent Representation-based RAG*, *Logit-based RAG*, and *Speculative RAG*.

In addition to these categorizations, another valuable classification arises from the type of knowledge source utilized in the RAG framework. RAG systems can be tailored to different knowledge sources, such as documents, knowledge graphs (KGs), and ontologies, each requiring distinct approaches for data retrieval and integration into the pipeline. The following subsections will go deeper into each stage of RAG, as well as explore the distinctions based on the type of information source employed.

7.2.4.1 RAG Framework

Building on the categorization presented by Gao et al. [10], this paragraph aims to provide a thorough and detailed examination of the three framework of the RAG, *Naive RAG*, *Advanced RAG*, and *Modular RAG*.

Naive RAG The Naive RAG represents the earliest iteration of the RAG. It operates through a straightforward sequence of steps: parsing, embedding, indexing, retrieval, and generation. While Naive RAG provides a foundational framework for RAG systems, its simplicity comes with several limitations: (i) the retrieval phase often struggles with accuracy, frequently retrieving irrelevant or incorrect

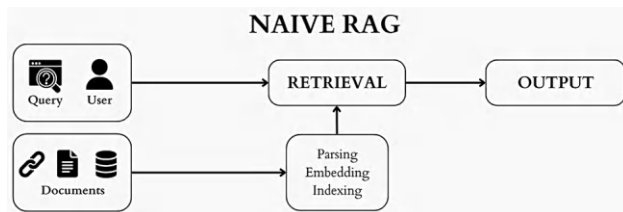


Fig. 7.4 Naive RAG framework

chunks and failing to capture all necessary information; *(ii)* during the generation phase, the model may produce hallucinations, generating content unsupported by the retrieved context. Additionally, *(iii)* responses can suffer from irrelevance, bias, or outright errors; integrating the retrieved information with the task at hand can be challenging, sometimes resulting in inconsistent or unclear outputs. Redundancy is another prevalent issue, as similar chunks may be retrieved multiple times, leading to repetitive responses. In Fig. 7.4, the Naive RAG framework with its steps is presented.

Advanced RAG Advanced RAG is designed to overcome the limitations of Naive RAG. The strategy adopted in this framework introduces two new processes, namely, *pre-retrieval* and *post-retrieval*, to optimize both indexing and query handling.

The *pre-retrieval* process focuses on two key elements: index optimization and query optimization. The goal of index optimization is to make the content of indexes more precise and representative to enhance retrieval accuracy. Techniques for this include improving data granularity (e.g., breaking down text into smaller, refined chunks for better management) and using mixed retrieval strategies that combine various types of data from multiple sources. On the other hand, query optimization involves the following processes: (i) *Query routing*, controlling which RAG flow the query is directed to; (ii) *Query rewriting*, rephrasing the query given in input from the user; and (iii) *Query expansion*, expanding the query with related terms to widen the search;

The *post-retrieval process* involves an initial re-ranking phase of the retrieved chunks to move the most relevant information to the top of the list provided to the LLM. This step is essential for prioritizing key information and generating more accurate outputs. Subsequently, a filtering step is performed: information is condensed through summarization and identification of critical sections to avoid overloading and help the model focus on essential and relevant details. Figure 7.5 illustrates the Advanced RAG framework.

Modular RAG The final framework in the evolution of the RAG is the *Modular RAG* [11], a structured, multi-level architecture designed to handle the increasing complexity of modern information retrieval and generation tasks. This architecture is organized across three distinct levels, each responsible for enhancing system modularity, scalability, and efficiency.

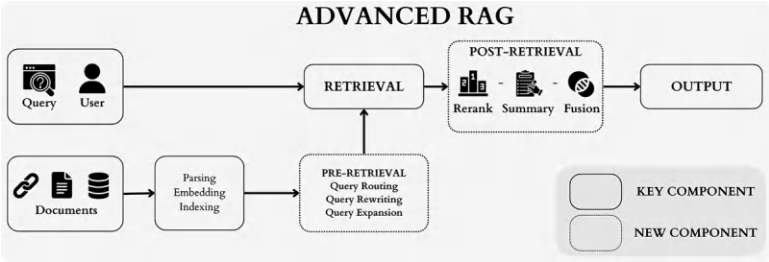


Fig. 7.5 Advanced RAG framework

Primary Level The framework follows a structured pipeline consisting of the five main stages presented in advanced RAG: parsing, embedding, and indexing, pre-retrieval, retrieval, post-retrieval, and generation. Each stage is responsible for a specific phase in the information processing workflow, ensuring a systematic progression from data preparation to content generation.

Secondary Level In this level, each stage of the pipeline is decomposed into specialized modules that handle distinct sub-tasks. For example, within the first step of pipeline (parsing, embedding, and indexing), modules like *chunk optimization* and *structural organization* ensure that data is processed and structured efficiently for retrieval. In pre-retrieval, components such as *query transformation*, *query expansion*, and *query construction* (as it has been already mentioned in advanced RAG explanation) refine the input query to improve retrieval accuracy. The retrieval stage integrates advanced mechanisms like *retriever fine-tuning*, *retriever source*, and *retriever selection*, enabling fine-grained control over data retrieval strategies. Similarly, post-retrieval involves modules for *reranking*, *compression*, and *selection*, optimizing the relevance and efficiency of retrieved information before generation. Finally, the generation phase incorporates *generator fine-tuning* and *verification* to ensure high-quality, accurate outputs.

Third Level This is the orchestration level in which the framework coordinates the interactions between modules through adaptive control mechanisms that dynamically govern the RAG process. Unlike rigid, linear approaches, Modular RAG introduces decision-making at pivotal junctures, allowing for flexible *routing*, *scheduling*, and *knowledge-guided reasoning*. (i) Routing directs queries through specific pipelines based on the input’s characteristics, using both metadata and semantic information to determine the most appropriate modules for each scenario. This decision-making process involves assigning scores to potential routes, enabling the system to activate only the most relevant modules for a given query. (ii) Scheduling manages the execution flow, identifying critical moments where additional retrieval or processing might be necessary. This mechanism supports recursive, iterative, and adaptive retrieval, ensuring that the system dynamically adjusts its operations based on the quality of intermediate results. (iii) The knowledge guide integrates knowledge graphs to construct reasoning chains that guide retrieval and

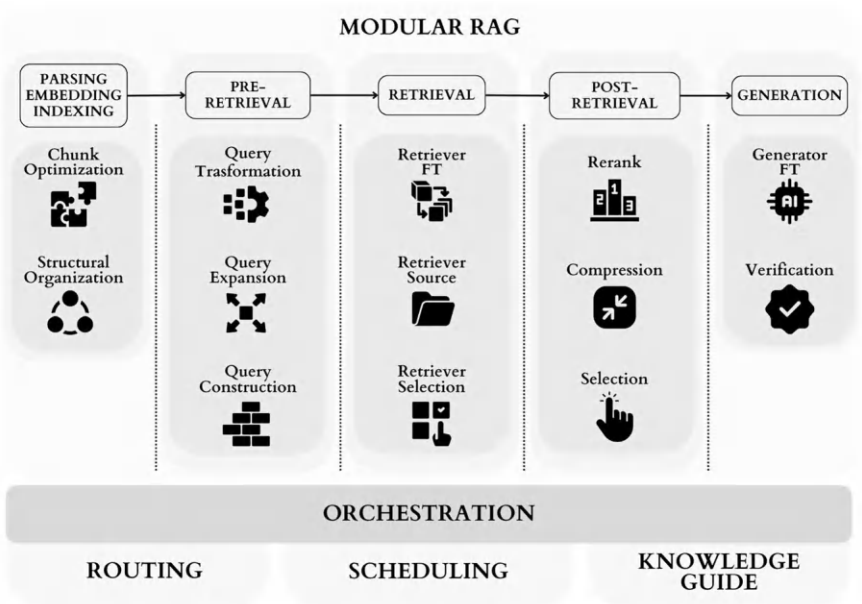


Fig. 7.6 Modular RAG framework

generation tasks. This part will be better explained in the next section since RAG across different knowledge sources will be discussed. In Fig. 7.6 modular RAG framework is presented.

7.2.4.2 RAG Across Knowledge Sources

Having established a detailed understanding of the fundamental workings of RAG, these paragraphs shift focus to explore how the framework can be adapted to different types of knowledge sources. The emphasis will be on demonstrating how the same foundational principles previously presented can be extended and customized to interact with various data formats to tackle different linguistic tasks. Specifically, we will focus on Classical RAG, RAG with KG, and RAG with ontologies. While no further details on knowledge sources are necessary for Classic RAG, as it is based on documents in various formats, it is essential to briefly explain what a knowledge graph and an ontology are.

According to [14], a KG is a *graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities*. Given this definition, it is clear that KG represents information as a network of entities (nodes)

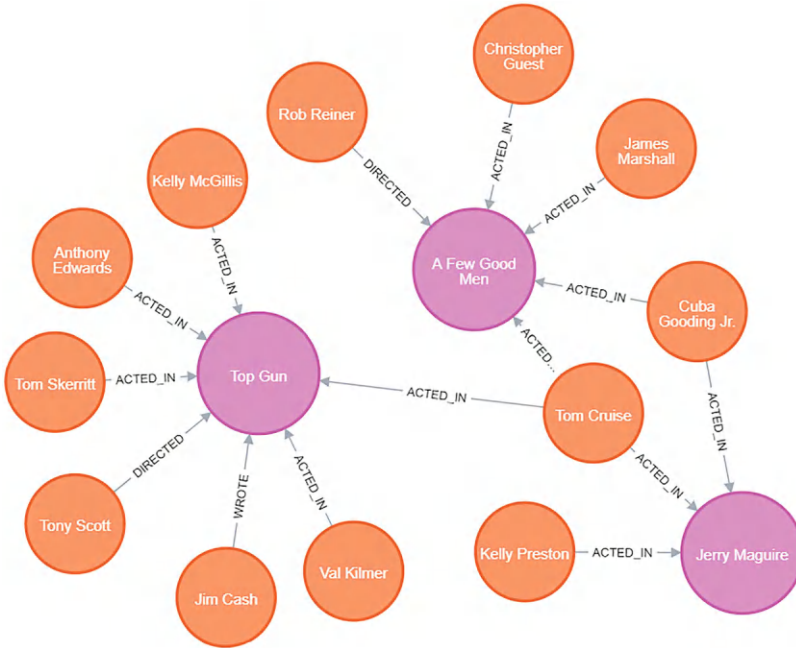


Fig. 7.7 An example of generic KG of people (actors and directors) who directed or acted in films. This KG is extracted from the default MOVIE DB within neo4j

and their relationships (edges), specifically as triplets *subject-predicate-object*. An example of KG created using Neo4j¹⁵ is presented in Fig. 7.7.

The strengths of knowledge graphs are several. Firstly, their *semantic structure* enables the embedding of explicit relationships, making them more intuitive for both human and machine reasoning [8]. Additionally, KGs offer significant *interconnectivity*, as an entity can have multiple connections to other entities and/or concepts, facilitating seamless access to both direct and indirect relationships. This structure allows for the exploration of first- and second-level links, providing a more comprehensive understanding of the data [2]. Finally, KGs are highly *scalable* [15], as they can represent knowledge at varying levels of granularity. They range from global knowledge bases like Wikidata¹⁶ or DBpedia¹⁷ to highly specialized, domain-specific graphs.

¹⁵ Neo4j is a native graph database platform designed to store and query connected data using a graph-based model of nodes, relationships, and properties, optimized for analyzing complex interconnections: <https://neo4j.com/>.

¹⁶ Wikidata [23] is a free, collaborative, multilingual knowledge graph: https://www.wikidata.org/wiki/Wikidata:Main_Page.

¹⁷ DBpedia [1] is a project that extracts structured content from Wikipedia, transforming it into a semantic knowledge base: <https://www.dbpedia.org/>.



Fig. 7.8 An example of an ontology extracted from BioPortal in the form of a list of classes and entities (left) and the graphical representation (right)

On the other hand, an ontology is a *description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents* [13]. Hence, it defines:

- *Classes*: categories or types of entities within the domain
- *Relations*: defined connections between classes or instances
- *Axioms*: logical rules and constraints that govern the relationships between classes and instances, enabling inference and reasoning

An example of an ontology created using Protegé¹⁸ is shown in Fig. 7.8, where we see an ontology extracted from the BioPortal¹⁹ repository in the form of a class and entity list (left), along with its subclasses and the same ontology as a graph (right), illustrating the relationships between the various classes and entities.

Although ontologies and knowledge graphs are often used interchangeably, they differ significantly in focus and structure. Knowledge graphs primarily represent entities and their interrelationships, linking real-world data with minimal predefined structure. For example, in a knowledge graph, *Albert Einstein* might be connected to *Theory of Relativity* through the relationship *discovered*. The emphasis in knowledge graphs is on connecting factual information to show how different pieces of data relate to each other. Ontologies, on the other hand, offer a more formal and structured approach to defining concepts and their relationships, focusing on capturing semantic meaning. In the case of *Albert Einstein*, an ontology might define that *Albert Einstein* is a *person* and that *Theory of Relativity* is a *scientific*

¹⁸ Protegé [22] is a free, open-source ontology editor and framework developed by the Stanford Center for Biomedical Informatics Research. It allows users to create, edit, and visualize ontologies: <https://protege.stanford.edu>.

¹⁹ <https://biportal.bioontology.org>.

theory, with a relationship that links them through a more abstract concept, such as *discovered*, but also adding logical rules such as *if someone is a scientist, then they are a person*. This allows the system to make inferences and reason about the relationships, offering a deeper, more structured understanding of the concepts involved. Ontologies are thus particularly useful in contexts that require logical reasoning and inference, such as in knowledge management systems or AI applications.

Classical RAG Classic RAG is the foundational form of RAG, leveraging a retriever and generator architecture to process and synthesize information from unstructured text documents. These documents can come in various formats, such as PDFs, Word files, HTML pages, or plain text, and do not require any predefined schema or structural organization. The system's versatility lies in its ability to handle diverse data formats without prior restructuring, making it a straightforward approach suitable for a wide range of applications. This simplicity, however, may limit its effectiveness in scenarios requiring complex reasoning or deeper integration of structured knowledge.

RAG with KG RAG with KG builds upon the classic RAG approach by incorporating structured knowledge into the retrieval process. In RAG-KG, the retriever operates on the graph structure rather than unstructured text. Queries are processed to identify relevant entities or subgraphs that provide context-rich and highly structured information. The retrieved subgraph or triples (subject-predicate-object) are then used to enhance the generation phase.

RAG with Ontologies RAG with ontologies is an innovative extension of the traditional RAG paradigm, where the knowledge base used for retrieval and generation is structured as an ontology [6, 31]. Unlike classic RAG models that rely on unstructured documents or KG, RAG with ontologies takes advantage of the formal, logically structured nature of ontologies to enable more precise, context-aware reasoning and retrieval.

7.3 RAG Applications

This section transitions into practical implementations. The first focus will be on Python-based implementations of RAG pipelines, specifically on Classic RAG, RAG with KG, and RAG with ontologies, utilizing two prominent libraries: LangChain²⁰ and LlamaIndex.²¹ These tools provide comprehensive frameworks for integrating retrieval and generation components, simplifying the construction

²⁰ <https://www.langchain.com/>.

²¹ <https://www.llamaindex.ai/>.

of scalable and effective RAG systems. The second regards framework for RAG evaluation, specifically DeepEval and RAGAS.

By demonstrating how to leverage these libraries, we aim to equip practitioners with the practical knowledge needed to build customized RAG pipelines tailored to specific use cases.

7.3.1 Practical Application—LlamaIndex

This first hands-on guide provides a detailed walk-through of how to implement the aforementioned RAG pipelines using LlamaIndex. Thanks to the flexible functionalities offered by this framework, the import, configuration, and final query processes remain identical across all three RAG types. However, the core processes of parsing, embedding, and indexing differ for each type. This design enables a perfectly modular architecture where the central components of the code can be seamlessly swapped to transition from one RAG type to another.

Import

```
1 import openai
2 import os
3 import fitz
4 import nest_asyncio
5
6 from rdflib import Graph
7 from dotenv import load_dotenv
8
9 from llama_index.llms.openai import OpenAI
10 from llama_index.core.settings import Settings
11 from llama_index.core import (
12     Document,
13     VectorStoreIndex,
14     PropertyGraphIndex,
15     SimpleDirectoryReader
16 )
```

This first block of code sets up the necessary libraries and configurations for implementing a RAG pipeline. It uses LlamaIndex for managing document retrieval and indexing. Key components from LlamaIndex are `VectorStoreIndex` for storing and querying vector embeddings, `PropertyGraphIndex` for constructing property graphs from the document's data, and `SimpleDirectoryReader` for file loading. Additionally, the code integrates OpenAI models for the generation process, `Settings` to configure global settings for language models, embeddings, and other behaviors and uses `dotenv` for securely managing environment variables like API keys. `nest_asyncio` enables nested asynchronous event loops, very

useful in interactive environments where an event loop is already running in the background, like in Jupyter Notebooks.

Configuration

```
1 # Apply the asyncio patch to allow nested event loops.
2 nest_asyncio.apply()
3
4 # Load environment variables from a .env file.
5 load_dotenv()
6
7 # Retrieve the OpenAI and Llama API key from the environment variables.
8 openai_api_key = os.getenv("OPENAI_API_KEY")
```

This block of code securely manages sensitive information, such as OpenAI API key²² (line 8). The `load_dotenv()` function (line 5) reads environment variables from a `.env` file located in the same directory as the script. This practice keeps sensitive information separate from the code, minimizing the risk of exposure in version control systems or public repositories. Environment variables can then be accessed using `os.getenv()` and set with `os.environ`, ensuring their security and anonymity. For proper functionality, ensure the **.env file is in the same directory as the script**.

File loading

```
1 # Specify required extensions as a list (if needed)
2 required_exts = [] # .png, .jpg, .pdf, .md, .txt, .csv etc...
3
4 # Load documents from a specified directory
5 reader = SimpleDirectoryReader(
6     input_dir="path/to/your/directory",
7     input_files=["path/to/your/directory/document_1",
8                 "path/to/your/directory/document_2"],
9     required_exts=required_exts
10 )
11
12 # Load and parse document from the reader
13 document = reader.load_data(num_workers=4)
```

In this part, `SimpleDirectoryReader` function serves to load multiple documents from a specified directory. The `input_dir` parameter (line 6) specifies the main directory from which all documents will be loaded automatically. This is useful for batch processing files without manually listing each one. On the other

²² To activate an API key for OpenAI, visit: <https://platform.openai.com/api-keys>.

hand, the `input_files` parameter (line 7) allows for specifying individual file paths directly, giving you the flexibility to load specific files from different locations if needed. Finally, the `required_exts` list (line 2) can be customized to filter files based on their extensions, such as `.pdf`, `.txt`, `.md`, or `.csv`. This setup enables the batch processing of multiple files simultaneously, making it ideal for preparing documents to populate a vector database. Finally, documents are loaded from the reader using `load_data` method (line 13).

These first blocks of code (import, configuration, and file loading) can be used as the basis for each following RAG type.

7.3.1.1 Classic RAG

Classic RAG - Pipeline set-up

```

1  # Create a vector index from the parsed documents.
2  index = VectorStoreIndex.from_documents(document)
3
4  # Initialize the OpenAI language model (LLM) for generating responses.
5  llm = OpenAI(
6  model="gpt-4-turbo",
7  strict=True,           # True default
8  temperature=0,        # from 0 to 1
9  max_tokens=2000,      # from 0 to infinity (None default)
10 top_p=0.8)             # from 0 to 1
11
12 # Convert the vector index into a query engine for running queries on
   ↳ the indexed documents.
13 query_engine = index.as_query_engine(
14 llm=llm,
15 response_mode = "compact",
16 streaming = True,
17 similarity_top_k=2,
18 verbose=True)

```

In this core part of the RAG pipeline, the process begins with the indexing of the document using `VectorStoreIndex` (line 2), which organizes the content into vector embeddings. These embeddings enable efficient similarity-based retrieval. Next, the OpenAI language model is initialized with the `OpenAI` function (lines 5). The parameters are as follows: `strict=True` ensures that input data is strictly validated to conform to expected formats; `temperature=0` is set to generate deterministic responses, reducing randomness for more consistent results; `max_tokens=2000` allows the model to generate longer responses when needed, up to a maximum of 2000 tokens; and `top_p=0.8` controls the diversity of the responses, narrowing down the possible options while maintaining some variety. Finally, a query engine is created using the `as_query_engine` method (line 13). The `verbose=True` option

enables detailed logging, providing real-time information about the execution and process status.

7.3.1.2 RAG with KG

RAG with KG - Pipeline set-up

```

1  # Create a PropertyGraphIndex from the parsed documents.
2  index = PropertyGraphIndex.from_documents(document)
3
4  # Initialize the OpenAI language model (LLM) for generating responses.
5  llm = OpenAI(
6  model="gpt-4-turbo",
7  strict=True,          # True default
8  temperature=0,        # from 0 to 1
9  max_tokens=2000,      # from 0 to infinity (None default)
10 top_p=0.8)            # from 0 to 1
11
12 # Convert the vector index into a query engine for running queries on
13 ↪ the indexed documents.
14 query_engine = index.as_query_engine(
15 llm=llm,
16 response_mode = "compact",
17 streaming = True,
18 similarity_top_k=2,
19 verbose=True)

```

This further block represents the core of RAG with KG. First, the documents are indexed using `PropertyGraphIndex` (line 2), a method tailored for constructing property graphs from the document's data. This indexing approach supports the representation of relationships and metadata, enhancing the system's ability to query and analyze structured knowledge. Finally, `llm` with `OpenAI` function (line 5) and a query engine with `query_engine` (line 13) are configured using `as_query_engine`. Key parameters are the same for Classic RAG.

7.3.1.3 RAG with Ontologies

RAG with Ontologies - Pipeline set-up

```

1  # Initialize an RDF graph and parse the selected XML file into it.
2  g = Graph()
3  g.parse(str(reader.input_files[0]), format="xml") # must be .owl, .json
4  ↪ or .csv
5
6  # Create a list of nodes from the RDF graph
7  nodes = []

```

```

7 documents_ontology = []
8
9 for s, p, o in g:
10 nodes.append({
11     "subject": str(s),
12     "predicate": str(p),
13     "object": str(o), })
14
15 # Generate documents from RDF nodes and store them in the
16 ⇨ documents_ontology list.
17 for node in nodes:
18 content = f"Subject: {node['subject']}, Predicate: {node['predicate']},
19 ⇨ Object: {node['object']}"
20 documents_ontology.append(Document(text=content))
21
22 # Index the documents into a PropertyGraphIndex for semantic search.
23 index_ontology = PropertyGraphIndex.from_documents(documents_ontology)
24
25 # Initialize the OpenAI model with specific configurations for response
26 ⇨ generation.
27 llm = OpenAI(
28 model="gpt-4-turbo",
29 strict=True, # True default
30 temperature=0, # from 0 to 1
31 max_tokens=2000, # from 0 to infinity (None default)
32 top_p=0.8), # from 0 to 1
33
34 # Set up the query engine to perform semantic search on ontology index.
35 query_engine = index_ontology.as_query_engine(
36 llm=Settings.llm,
37 response_mode = "compact",
38 streaming = True,
39 similarity_top_k=2,
40 verbose=True)

```

This final block of code illustrates the core implementation of an ontology-based RAG pipeline, which uses RDF data to construct a retrieval system through a property graph. Initially, the code leverages the RDF graph parser `Graph()` (lines 2–3) to load and parse the selected ontology file. The ontology must be provided in `.owl`, `.json`, or `.csv` format, with the data being processed as XML. The RDF triples are then iterated through, and each triple is stored as a structured node, where each node captures a distinct relationship or fact derived from the ontology. Next, these nodes are converted into text documents, stored in the list `documents_ontology` (line 18). Each document represents an RDF triple as a textual string, ensuring the data is suitable for downstream indexing and retrieval. These textual representations are indexed to create a `PropertyGraphIndex` (line 21), which organizes the ontology data into a graph structure. The `llm` is set with `OpenAI` function (line 24). Finally, a query engine using `query_engine` (line 32) is instantiated using the property graph. The `llm` and the engine are configured with

parameters previously discussed, enabling it to process user queries by retrieving relevant results based on the indexed ontology data.

7.3.1.4 Query and Response

Query and response

```
1 user_query = input("Enter your query: ")
2 response = query_engine.query(user_query)
```

This final segment of the pipeline is designed to execute user queries across **any of the RAG configurations previously discussed**. The `user_query` (line 1) serves as the input for the query engine (line 2), which processes the query and retrieves relevant context from the indexed data to generate a response. The retrieval and response generation process is entirely interchangeable and adapts seamlessly to the underlying knowledge source. Additionally, the `user_query` can be enriched using prompt engineering techniques. By carefully structuring the query with specific instructions, constraints, or context, the system can be better directed toward performing the desired task, such as summarization, reasoning, or detailed explanation.

7.3.2 Practical Application—Langchain

This second hands-on guide focuses on implementing RAG pipelines using the LangChain framework. Unlike the previous section, where we provided plug-and-play code structures applicable to all three types of RAG (classical, RAG with knowledge graphs, and RAG with ontologies), the implementations here are more specialized. While the initial steps for classical RAG and ontology RAG can be unified, implementing RAG with KG requires a separate block of code.

7.3.2.1 Classic RAG and Ontology RAG

Import

```
1 import os
2 import json
3 import nest_asyncio
4
5 from dotenv import load_dotenv
6
7 from langchain.text_splitter import RecursiveCharacterTextSplitter
```

```

8 from langchain.vectorstores import FAISS
9 from langchain.chains import RetrievalQA
10 from langchain_openai import ChatOpenAI, OpenAIEmbeddings
11 from langchain.document_loaders import (
12     TextLoader,
13     PyPDFLoader,
14     CSVLoader,
15     JSONLoader
16 )

```

This first block of code sets up the necessary libraries and configurations for implementing a RAG pipeline using Langchain. It imports various modules for document loading, such as `TextLoader`, `PyPDFLoader`, `CSVLoader`, and `JSONLoader` (lines 11–15), which handle different file formats (.txt, .pdf, .csv, .json). The `RecursiveCharacterTextSplitter` (line 7) is used to split large documents into smaller chunks for easier processing and querying. The code utilizes `FAISS` (line 8) for efficient vector storage and similarity-based retrieval of document embeddings. For the generation process, the `ChatOpenAI` model and `OpenAIEmbeddings` (line 10) are integrated to facilitate semantic search and response generation. The purpose of `dotenv` and `nest_asyncio` libraries have already been explained in previous section.

Configuration

```

1 # Apply the asyncio patch to allow nested event loops.
2 nest_asyncio.apply()
3
4 # Load environment variables from a .env file.
5 load_dotenv()
6
7 # Retrieve the OpenAI API key from the environment variables.
8 openai_api_key = os.getenv("OPENAI_API_KEY")

```

Since this is identical to the previous implementation, no further explanation is needed.

File Loading

```

1 # Load a PDF document from the specified path using PyPDFLoader.
2 loader = PyPDFLoader('path/to/your/directory/document.pdf')
3 # loader = TextLoader('path/to/your/directory/document.txt')
4 # loader = CSVLoader('path/to/your/directory/document.csv')
5 # loader = JSONLoader('path/to/your/directory/document.json')
6
7 document = loader.load()

```

This simple block of code manages file selection and document loading for the RAG pipeline. Specifically, it loads PDF files using the `PyPDFLoader` function (line 2). If a different document format needs to be loaded, this function can be replaced with `TextLoader` for `.txt` files, `CSVLoader` for `.csv` files, or `JSONLoader` for `.json` files, as presented in comments (lines 3–5).

Parsing, Embedding, Indexing

```

1  # Split documents into smaller chunks for easier processing.
2  text_splitter = RecursiveCharacterTextSplitter(
3      chunk_size=1000,
4      chunk_overlap=200
5  )
6
7  # Split the documents into smaller chunks using the text splitter.
8  docs = text_splitter.split_documents(document)
9
10 # Create embeddings using the "text-embedding-3-large" model
11 embeddings = OpenAIEmbeddings(model="text-embedding-3-large")
12
13 # Create a FAISS vector store from the documents and embeddings for
14 # ↪ efficient similarity search.
15 vectorstore = FAISS.from_documents(docs, embeddings)
16
17 # Initialize the ChatOpenAI model with specific configurations for
18 # ↪ response generation.
19 llm = ChatOpenAI(
20     model="gpt-4o",
21     temperature=0,          # from 0 to 1
22     max_tokens=2000,        # from 0 to infinity
23     top_p=0.8,              # from 0 to 1
24     api_key=openai_api_key)
25
26 # Set up a RetrievalQA chain for querying
27 qa_chain = RetrievalQA.from_llm(
28     llm=llm,
29     retriever=vectorstore.as_retriever(),
30 )

```

This block of code sets up the core components of the RAG pipeline. The `RecursiveCharacterTextSplitter` (line 2) is initialized with a chunk size of 1000 characters and an overlap of 200 characters, ensuring that the documents are split into manageable and contextually cohesive chunks. Note that `chunk_overlap` must be smaller than `chunk_size`. The `split_documents` method (line 8) is then used to apply this splitter to the loaded documents, effectively preparing them for downstream tasks. To handle embeddings, the `OpenAIEmbeddings` class (line 11) is instantiated using the `text-embedding-3-large` model. Other available embedding models include `text-embedding-3-small` and

text-embedding-ada-002.²³ These embeddings are stored in a FAISS vector store via the `FAISS.from_documents` method (line 14), enabling efficient similarity-based searches. The `ChatOpenAI` model (line 17) is then initialized with specific parameters, the same presented for `LlamaIndex` in the previous section. Finally, the `RetrievalQA.from_llm` method (line 25) creates a retrieval-based question-answering chain by linking the initialized language model (`llm`) and the FAISS vector store retriever (`vectorstore.as_retriever()`).

Query and response

```
1 user_query = input("Enter your query: ")
2 response = qa_chain.invoke(user_query)
```

In this final block the user query is processed using the `qa_chain.invoke` method (line 3). The functionality mirrors the equivalent steps in the `LlamaIndex` framework.

7.3.2.2 RAG with KG

Now we present an alternative implementation of a RAG pipeline with knowledge graphs.

Import

```
1 import os
2
3 from dotenv import load_dotenv
4 from neo4j import GraphDatabase
5
6 from langchain_openai import ChatOpenAI
7 from langchain.chains import GraphCypherQAChain
8 from langchain.graphs import Neo4jGraph
9 from langchain.prompts import PromptTemplate
```

This initial block imports the required libraries for integrating a knowledge graph with the RAG pipeline. It includes `Neo4j` as the graph database backend and `LangChain` modules such as `GraphCypherQAChain` and `Neo4jGraph` for

²³ More information on embedding models by OpenAI is available at <https://platform.openai.com/docs/guides/embeddings>.

interacting with the KG. The `PromptTemplate` is used to customize queries sent to the model, while `pprint` assists with structured output during debugging.

Configuration and KG loading

```

1  # Load environment variables from a .env file
2  load_dotenv()
3
4  # Retrieve API keys and database credentials from a .env file
5  openai_api_key = os.getenv("OPENAI_API_KEY")
6  neo4j_username = os.getenv("NEO4J_USERNAME")
7  neo4j_password = os.getenv("NEO4J_PASSWORD")
8
9  # Retrieve the Neo4j database URI and set authentication credentials.
10 URI = os.getenv("NEO4J_URI")
11 AUTH = (neo4j_username, neo4j_password)
12
13 # Initialize the Neo4jGraph connection with enhanced schema support.
14 graph = Neo4jGraph(
15     url=URI,
16     username=neo4j_username,
17     password=neo4j_password,
18     enhanced_schema=True,
19 )
20
21 # Define a Cypher query to load CSV data from a specific URL and create
22 ↪ nodes/relationships.
23 cypher_query = """
24 LOAD CSV WITH HEADERS FROM 'specific URL to load CSV file' AS row
25
26 // Replace the following MERGE clauses with your specific data modeling
27 ↪ logic.
28 // Use MERGE to create nodes and relationships based on the data in the
29 ↪ CSV.
30 // Add your domain-specific MERGE clauses here
31 """
32 graph.query(cypher_query)

```

This block configures environment variables for connecting to the Neo4j database and OpenAI services. Using `dotenv`, it retrieves and sets the necessary details, including the OpenAI API key, Neo4j URI, username, and password from the `.env` file (lines 5–11). Following this, the `Neo4jGraph` class (line 14) from `Langchain` is initialized using the Neo4j URI, username, and password, which are retrieved from the environment variables. This object provides a convenient interface to interact with the graph database. The `enhanced_schema` parameter (line 18) is set to `True` to enable advanced schema capabilities for more complex graph queries. The first part of the code allows you to interact with and upload an existing graph that is already present in Neo4j. The second part of the code allows you to execute a Cypher query (line 22) that performs an upload from a CSV file

into the Neo4j graph database. The `LOAD CSV` clause in Cypher loads data from the specified file, and the `MERGE` clauses within the query are used to create nodes and relationships based on the contents of the CSV file. The `insert specific URL to load CSV file placeholder` (line 23) should be replaced with the actual location of the CSV file to be uploaded.

Pipeline set-up

```

1  # Define a custom prompt template for user input, allowing future
   ↳ customization.
2  chat_prompt = PromptTemplate(
3  input_variables=["user_input"],
4  template="" # Template left intentionally blank for customization
5  )
6
7  # Initialize the OpenAI model with specific configurations for response
   ↳ generation
8  llm = ChatOpenAI(
9  model="gpt-4o",
10 temperature=0, # from 0 to 1
11 max_tokens=2000, # from 0 to infinity
12 top_p=0.8, # from 0 to 1
13 api_key=openai_api_key)
14
15 # Create a GraphCypherQAChain for querying the Neo4j graph using
   ↳ natural language.
16 chain = GraphCypherQAChain.from_llm(
17 llm,
18 graph=graph,
19 prompt=chat_prompt,
20 verbose=True,
21 return_intermediate_steps=True,
22 validate_cypher=True,
23 top_k=5,
24 allow_dangerous_requests=True
25 )

```

This block of code sets up a `GraphCypherQAChain` (line 16), enabling queries on a Neo4j graph using OpenAI's `gpt-4o` model. A `PromptTemplate` defines the input format for user queries, allowing for future customization as needed. The `ChatOpenAI` model (line 2) is configured with key parameters such as `temperature`, `max_tokens`, and `top_p`, while the OpenAI API key is provided for authentication. The `GraphCypherQAChain` connects the language model, the Neo4j graph instance, and the input prompt (line 17–19). Additional options include enabling verbose logging for detailed output, returning intermediate reasoning steps, validating Cypher queries before execution, limiting

the number of results with `top_k`, and permitting potentially unsafe requests via `allow_dangerous_requests` (line 20–24).

Query and response

```
1 user_query = input("Enter your query: ")
2 response = chain.invoke(users_query)
```

Finally, we can execute our query using `chain.invoke` (line 2).

7.3.3 RAG Evaluation

Evaluating RAG systems is crucial for ensuring their effectiveness in real-world applications. Unlike traditional generative models, RAG combines retrieval and generation components, which requires evaluation strategies that assess both aspects comprehensively. Earlier assessments of RAG systems often focused on task-specific metrics, such as F1 and EM scores [28, 33] for Question Answering or Accuracy for Fact-Checking [26]. However, modern evaluation frameworks now emphasize not only task performance but also the quality of retrieval and the fidelity of generation, making them more suited to assess the unique characteristics of RAG systems.

This section introduces two widely used frameworks for RAG evaluation: DeepEval²⁴ and RAGAS.²⁵ First, we will present the key evaluation metrics, which are divided into *retrieval evaluation metrics* and *generation evaluation metrics*, to offer a comprehensive understanding of how to measure the quality of retrieval and generated outputs. In this context, it is important to clarify that all of these metrics use LLMs as a critical component for assessing the quality of the retrieval and generation processes. For retrieval metrics such as *Contextual Precision* and *Contextual Recall*, the LLM evaluates the relevance of nodes in the retrieval context and determines their alignment with the input and expected output. For generation metrics like *Answer Relevancy* and *Faithfulness*, the LLM evaluates the factual accuracy and relevance of the generated output in relation to the input query and retrieval context. Additionally, the LLM provides reasoning for its judgments, which enhances the transparency and interpretability of the evaluation process. After introducing these metrics, we will provide implementation examples for DeepEval, demonstrating how to apply them in practice. Due to space limitations, detailed implementation examples for RAGAS will be included in GitHub.

²⁴ <https://docs.confident-ai.com/docs/getting-started>.

²⁵ <https://docs.ragas.io/en/stable/>.

7.3.3.1 Retrieval Evaluation Metrics

The following retrieval evaluation metrics are essential for assessing the performance of the retriever component in a RAG pipeline. These metrics evaluate the quality and relevance of the retrieved documents or nodes from the retrieval context in relation to the expected output.

Contextual Precision The *Contextual Precision* metric (CP) assesses the effectiveness of the retriever by measuring how well the retrieved nodes in the retrieval context match the relevance of the given input query. In particular, it evaluates whether relevant nodes are ranked higher than irrelevant ones. This metric employs a self-explaining LLM-based evaluation, where an LLM is used to assess the relevance of each retrieved node to the query, and justifies its relevance judgment. The CP score prioritizes nodes that appear earlier in the retrieval list, reflecting their higher importance. The CP is calculated using the following formula:

$$CP = \frac{1}{No. \text{ of Relevant Nodes}} \cdot \sum_{k=1}^n \left(\frac{No. \text{ of Relevant Nodes up to Position } k}{k} \cdot r_k \right)$$

where:

- k is the position of the k -th node in the *retrieval_context*
- n is the total number of nodes in the *retrieval_context*
- r_k is the binary relevance of the k -th node ($r_k = 1$ if relevant, $r_k = 0$ otherwise)

The resulting score is computed using the *Weighted Cumulative Precision* (WCP), which aggregates the relevance assessments while prioritizing nodes ranked higher in the retrieval list. The WCP increases the contribution of relevant nodes that appear earlier in the ranking, reflecting their greater importance in a retrieval context.

Contextual Recall The *Contextual Recall* metric (CR) evaluates the ability of the retriever to capture all the relevant information in the retrieval context with respect to the expected output. It measures how well the retrieval context aligns with the expected output by calculating the proportion of statements in the expected output that are supported by the retrieved context. Similar to CP, this metric also uses an LLM-based evaluation to assess how well each statement in the expected output is supported by the retrieval context. The CR is calculated as:

$$CR = \frac{Number \text{ of Attributable Statements}}{Total \text{ Number of Statements in Expected Output}}$$

where:

- *Attributable statements* are statements in the expected output that can be traced back to the nodes in the retrieval context

- *Total number of statements* refers to the total number of distinct statements identified in the expected output

The LLM plays a crucial role here by extracting and classifying the statements made in the expected output. It then determines which of those statements are attributable to the nodes in the retrieval context. The final score reflects the proportion of statements in the expected output that are adequately supported by the retrieved context.

7.3.3.2 Generation Evaluation Metrics

The following metrics focus on evaluating the quality of the generated output in a RAG pipeline, specifically assessing the coherence, relevance, and factual accuracy of the answers generated by the model.

Answer Relevancy The *Answer Relevancy* metric evaluates how relevant the actual output generated by the LLM is in relation to the input query. It measures the proportion of relevant statements in the generated output compared to the total number of statements. This metric is essential for understanding how well the generated answer addresses the user's query. The Answer Relevancy Metric (AR) is calculated as follows:

$$AR = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements in Actual Output}}$$

where:

- *Relevant statements* are those statements in the actual output that are directly related to the input query
- *Total number of statements* refers to the total number of distinct statements made in the actual output

The LLM is responsible for evaluating the relevance of each statement in the generated output. It provides the reasoning behind its classification, enabling transparency in the evaluation process.

Faithfulness The *Faithfulness* metric evaluates the degree to which the actual output aligns with the factual information contained in the retrieval context. It ensures that the generated answer does not introduce hallucinations or incorrect information, maintaining consistency with the retrieved context. This metric uses an LLM-based evaluation, where the LLM checks if the claims made in the generated output are truthful and supported by the retrieval context. The Faithfulness Metric (F) is calculated as:

$$F = \frac{\text{Number of Truthful Claims}}{\text{Total Number of Claims in Actual Output}}$$

where:

- *Truthful claims* are those claims in the actual output that are factually accurate and supported by the retrieval context
- *Total number of claims* refers to the total number of distinct claims made in the actual output

The LLM is used to verify the truthfulness of each claim in the generated output by comparing it to the information in the retrieval context. The reasoning behind the evaluation is provided, allowing for an explanation of which claims are considered truthful and which are not.

7.3.3.3 Practical Implementation with DeepEval

DeepEval - Import

```

1 # Importing the necessary evaluation metrics from the deepeval library
2 from deepeval.metrics import ContextualPrecisionMetric
3 from deepeval.metrics import ContextualRecallMetric
4 from deepeval.metrics import AnswerRelevancyMetric
5 from deepeval.metrics import FaithfulnessMetric
6 from deepeval.test_case import LLMTestCase
7 from deepeval import evaluate

```

In this first block of code, necessary library are imported, including those for the retrieval and generation evaluation metrics, namely, *Contextual Precision*, *Contextual Recall*, *Answer Relevancy*, and *Faithfulness* (lines 2–5). *LLMTestCase* and *evaluate* (lines 6–7) are fundamental for the actual execution of the evaluation process.

Metrics configuration and initialization

```

1 # Defining the environment variable for OpenAI API key
2 os.environ["OPENAI_API_KEY"] = "your_openai_api_key"
3
4 #Initializing test inputs
5 actual_output = " "
6 expected_output = " "
7 retrieval_context = []
8
9 # Initializing the ContextualPrecisionMetric
10 metric_CP = ContextualPrecisionMetric(
11     threshold = 0.7,
12     model = "gpt-4o-mini",
13     include_reason = True
14 )
15
16 # Initializing the ContextualRecallMetric
17 metric_CR = ContextualRecallMetric(

```

```

18 threshold = 0.7,
19 model = "gpt-4o-mini",
20 include_reason = True
21 )
22
23 # Initializing the AnswerRelevancyMetric
24 metric_AR = AnswerRelevancyMetric(
25     threshold = 0.7,
26     model = "gpt-4o-mini",
27     include_reason = True
28 )
29
30 # Initializing the FaithfulnessMetric
31 metric_F = FaithfulnessMetric(
32     threshold = 0.7,
33     model = "gpt-4o-mini",
34     include_reason = True
35 )

```

Here, the code sets up the api key (line 2) and the inputs for evaluation (lines 5–7). The `actual_output` (line 3) represents the generated response from the model, while the `expected_output` (line 4) is the ground truth or reference answer. The `retrieval_context` (line 5) is a list of strings retrieved by the system, which the model uses to generate its response. Moreover, all single metrics are initialized (lines 8, 15, 22, 29), setting specific parameters, such as a `threshold`, used to determine relevance, and the model to be evaluated, in this case, `gpt-4o-mini`. The `include_reason` parameter ensures that the reasoning behind the score is provided.

Executing test

```

1 # Creating a test case to evaluate contextual precision (CP)
2 test_case_PC = LLMTestCase(
3     input = "",
4     actual_output = actual_output,
5     expected_output = expected_output,
6     retrieval_context = retrieval_context
7 )
8
9 # Creating a test case to evaluate contextual recall (CR)
10 test_case_CR = LLMTestCase(
11     input = "",
12     actual_output = actual_output,
13     expected_output = expected_output,
14     retrieval_context = retrieval_context
15 )
16
17 # Creating a test case to evaluate answer relevancy (AR)
18 test_case_AR = LLMTestCase(

```

```

19 input = "",
20 actual_output = actual_output,
21 )
22
23 # Creating a test case to evaluate faithfulness (F)
24 test_case_CR = LLMTestCase(
25     input = "",
26     actual_output = actual_output,
27     retrieval_context = retrieval_context
28 )
29
30 # Measuring the evaluation score for the test case
31 metric_CP.measure(test_case_CP)
32 metric_CP.measure(test_case_CR)
33 metric_CP.measure(test_case_AR)
34 metric_CP.measure(test_case_F)
35
36 # Evaluating the test case with the metric and displaying results
37 evaluate([test_case_CP], [metric_CP])
38 evaluate([test_case_CR], [metric_CR])
39 evaluate([test_case_AR], [metric_AR])
40 evaluate([test_case_F], [metric_F])

```

Inputs defined in *Metrics configuration and initialization* block go into each LLMTestCase function (lines 2, 10, 18, 24). The `measure` method (lines 31–34) is then called to evaluate the test case, calculating the score based on the provided inputs. Finally, the evaluation is conducted using the `evaluate` function (lines 37–40), which processes and outputs the final evaluation results.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: international Semantic Web Conference, pp. 722–735. Springer (2007)
2. Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., Vidal, M.E.: Towards a knowledge graph for science. In: Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, pp. 1–6 (2018)
3. Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G.B., Lespiau, J.B., Damoc, B., Clark, A., et al.: Improving language models by retrieving from trillions of tokens. In: International Conference on Machine Learning, pp. 2206–2240. PMLR (2022)
4. Boykis, V.: What Are Embeddings. <https://doi.org/10.5281/zenodo.8015029> (2023)
5. Caballero, M.: A brief survey of question answering systems. Int. J. Artif. Intell. Appl. (IJAIA) **12**(5), (2021)
6. DeBellis, M., Duttav, N., Ginoc, J., Balajid, A.: Integrating ontologies and large language models to implement retrieval augmented generation (rag). Appl. Ontol. **1**, 1–5 (2024)
7. Devlin, J.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

8. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)* **48**(1-4), 2 (2016)
9. Ferrucci, D.A.: Introduction to “this is watson”. *IBM J. Res. Dev.* **56**(3.4), 1:1–1:15 (2012). <https://doi.org/10.1147/JRD.2012.2184356>
10. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023)
11. Gao, Y., Xiong, Y., Wang, M., Wang, H.: Modular rag: Transforming rag systems into lego-like reconfigurable frameworks. *arXiv preprint arXiv:2407.21059* (2024)
12. Graham, K.: Techmatters: What happened to jeeves, you ask. com? *LOEX Quart.* **33**(3), 3 (2006)
13. Gruber, T.: What is an ontology (1993)
14. Hogan, A., Blomqvist, E.: Knowledge graphs. *ACM Comput. Surv. (Csur)* **54**(4), 1–37 (2021)
15. Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y.: A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Networks Learn. Syst.* **33**(2), 494–514 (2021)
16. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**(12), 1–38 (2023). <https://doi.org/10.1145/3571730>
17. Krajewski, M., Marcum Charles, I.: Ask Jeeves: Servants as search engines. *Grey Room* (38), 6–19 (2010). <https://doi.org/10.1162/grey.2010.1.38.6>
18. Lee, C., Roy, R., Xu, M., Raiman, J., Shoeybi, M., Catanzaro, B., Ping, W.: Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428* (2024)
19. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **33**, 9459–9474 (2020)
20. Liu, Y.: Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* **364** (2019)
21. Mikolov, T.: Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* **3781** (2013)
22. Noy, N.F., Crubézy, M., Ferguson, R.W., Knublauch, H., Tu, S.W., Vendetti, J., Musen, M.A.: Protégé-2000: an open-source ontology-development and knowledge-acquisition environment: Amia 2003 open source expo. In: *Amia Annual Symposium Proceedings*, vol. 2003, p. 953 (2003)
23. Pellissier Tanon, T., Vrandečić, D., Schaffert, S., Steiner, T., Pintscher, L.: From freebase to wikidata: The great migration. In: *Proceedings of the 25th International Conference on World Wide Web*, pp. 1419–1428 (2016)
24. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543 (2014)
25. Reimers, N.: Sentence-bert: Sentence embeddings using Siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019)
26. Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., Chen, W.: Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. *arXiv preprint arXiv:2305.15294* (2023)
27. Tonmoy, S., Zaman, S., Jain, V., Rani, A., Rawte, V., Chadha, A., Das, A.: A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313* (2024)
28. Van Rijsbergen, C.J.: Foundation of evaluation. *J. Document.* **30**(4), 365–373 (1974)
29. Vaswani, A.: Attention is all you need. *Adv. Neural Inf. Process. Syst.* (2017)

30. Wang, C., Liu, X., Yue, Y., Tang, X., Zhang, T., Jiayang, C., Yao, Y., Gao, W., Hu, X., Qi, Z., Wang, Y., Yang, L., Wang, J., Xie, X., Zhang, Z., Zhang, Y.: Survey on factuality in large language models: Knowledge, retrieval and domain-specificity (2023). <https://arxiv.org/abs/2310.07521>
31. Xiao, J., Ding, L., Barry, J., Elkaref, M., De Mel, G., Han, J.: Orag: Ontology-guided retrieval-augmented generation for theme-specific entity typing. In: First Conference on Language Modeling (2024)
32. Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., Cui, B.: Retrieval-augmented generation for ai-generated content: A survey. arXiv preprint arXiv:2402.19473 (2024)
33. Zhu, K., Luo, Y., Xu, D., Wang, R., Yu, S., Wang, S., Yan, Y., Liu, Z., Han, X., Liu, Z., et al.: Rageval: Scenario specific rag evaluation dataset generation framework. arXiv preprint arXiv:2408.01262 (2024)

Chapter 8

Large Language Model Agents



Jerin George Mathew  and Jacopo Rossi 

Abstract Large language model (LLM) agents are transforming the landscape of artificial intelligence, enabling sophisticated interactions, reasoning, and autonomous decision. This chapter explores the theoretical and practical foundations of LLM agents and multi-agent systems, highlighting their architecture, communication strategies, and coordination mechanisms. The chapter also presents examples of LLM agent applications in software engineering, demonstrating their impact on tasks such as code generation, debugging, and automated software development. Finally, the chapter describes the programming paradigms for designing and deploying intelligent agents and multi-agent coordination frameworks. By summarizing key advancements and methodologies, this chapter provides a comprehensive guide to developing and understanding LLM agents and multi-agent systems in modern AI applications.

8.1 Introduction

Addressing complex and articulated tasks has always been one of the main challenges in computer science. Large language models (LLMs) have a significant impact on solving intricate problems. However, LLMs alone often have trouble with complex tasks that require planning, multi-step reasoning, or using external information. Expanding the context window can mitigate some limitations but does not fully address these challenges. One promising approach is the development of *LLM agents* [12], which augment LLMs with additional modules dedicated to memory, planning, and external tools.

In artificial intelligence, the concept of *agent* is fundamental, referring to an entity that perceives its environment, processes information, and takes actions to achieve specific goals. This notion is particularly relevant in reinforcement learning (RL) [1], where an agent interacts with an environment, learning through feedback

J. G. Mathew (✉) · J. Rossi
Sapienza Università di Roma, Rome, Italy
e-mail: mathew@diag.uniroma1.it; monti@diag.uniroma1.it; j.rossi@diag.uniroma1.it

to maximize cumulative rewards. Similarly, LLM agents enhance the ability to solve complex and articulated tasks. Such tasks often require the execution of multiple actions and steps, which must be managed with great care and handled by different specialized components. These solutions enable LLMs to interact with the external environment, further expanding their capabilities.

LLM agents operate in such situations by integrating various module components. These components collectively enable the system to handle complex tasks, including iterative learning from prior outcomes.

To understand the importance of LLM agents, we can analyze in detail two possible problems to solve:

- (a) Analyze the environmental impacts of two different cars.
- (b) Analyze the life cycle emissions of two different cars, including production, usage, and disposal. There is the need to generate a report summarizing key findings and create a chart showing the annual reduction in emissions for the two different models

As we can observe, the first problem (*a*) could be entirely resolved by an LLM when it has sufficient knowledge. If the LLM lacks the necessary information, a RAG (retrieval-augmented generation) system could be used, having access to customized documents that serve as the knowledge base.¹

Regarding the second problem (*b*), it is clear that the exclusive use of an LLM is not sufficient to solve the entire task. This is because the problem requires complex operations that cannot be managed directly in a single iteration. In this context, LLM agents come into play: the task can be divided into subtasks, which can be addressed using specific tools and following a flow of operations that leads to the desired final response. An effective approach might involve creating an LLM agent equipped with access to APIs and a database, enabling it to deliver relevant insights on environmental impact. Furthermore, the agent should incorporate a tool for generating visualizations, such as charts, to better interpret trends related to pollution. These components can be considered the core of a potential LLM agent; however, it is also crucial to consider other aspects, such as incorporating planning and memory modules, which are further discussed below.

8.2 LLM Agent Structure

LLM agents have several fundamental components that define their operational capabilities and performance. These elements are essential for information processing, task management, and interaction with external data sources. Specifically, three primary categories can be identified that define the functioning of LLM Agents (c.f. Fig. 8.1):

¹ For a detailed overview about RAG system, see Chap. 7.

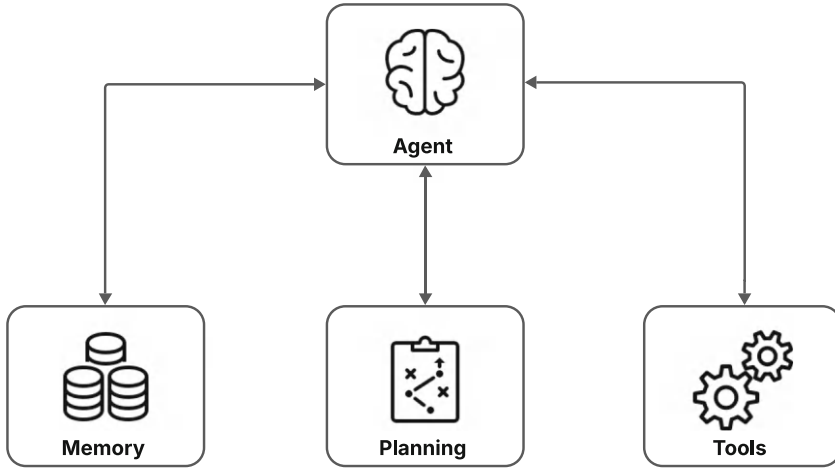


Fig. 8.1 Architecture of LLM agent

- **Memory:** divided into *short-term memory*, which allows to retain relevant information temporarily for immediate tasks, and *long-term memory*, which can be used to store extensive information in an external storage system.
- **Planning:** breaks down complex tasks into smaller, manageable subgoals, allowing for more efficient handling and organization. This technique is complemented by the agent's capacity for self-assessment, where it reflects on past actions, identifies mistakes, and refines its methods for future steps, enhancing overall performance.
- **Tools:** use external resources and applications to recover missing information, execute operations, or have access to proprietary data. This augments agent knowledge and real-time abilities.

8.2.1 Memory

The integration of memory modules in LLM agents is essential to make the system more precise and adaptable [23]. LLM agents have the ability to retain and organize information internally, enabling them to navigate complex, multi-step processes. They arrange data into distinct memory modules, each designed to address specific informational purposes. Memory modules act as a structured and dynamic system that allows agents to use stored information, improving decision-making, increasing adaptability and building the cognitive basis of the agent. This includes two different types of memory: short-term memory and long-term memory.

- **Short-term memory** is a temporary, immediate storage that holds information relevant to the agent's current task or session. Short-term memory enables

the agent to maintain context within a single interaction or trial, allowing it to give answer based on the ongoing context [23]. For instance, if an agent is having a conversation, short-term memory stores details from earlier in that conversation, helping it understand follow-up questions or references. It facilitates a coherent flow within a task by retaining recent actions, responses, or observations, making it essential for multi-step processes like guiding a user through a series of instructions or tasks in a single session (e.g., a chatbot for customer support). Since its focus is on the current context, short-term memory enables the agent to retrieve relevant information quickly, keeping interactions efficient and responsive. Short-term memory also communicates with long-term memory systems, acting as the core integrative layer that links the different functions of a language agent [17].

- Long-term memory enables agents to accumulate and store knowledge and insights over time. Various methodologies are employed to manage data in long-term memory, allowing for the distinction between different types of memory [23]. The use of cross-trial information includes details from historical data related to past iterations and their outcomes. This approach optimizes future decisions for specific tasks but has some limitations in its dependency on past interactions between the agent and its environment. Closely related to these methods is also the concept of episodic memory, which stores data from earlier decision iterations. There are also approaches that incorporate external knowledge rather than relying solely on the agent's direct experiences. This information can be stored as parameters (encoded in model through fine-tuning or training), textual data (readily interpretable), or external vector databases and support tasks that require historical or contextual understanding. Some methods draw from general sources like Wikipedia or other Website [19, 25], while others utilize specialized repositories [11, 21]. In this context, the semantic memory is also important, which stores world knowledge, enabling agents to integrate new insights through reasoning.

8.2.2 Planning

Planning for LLM agents represents an ambitious frontier. Numerous studies aim to integrate LLMs with planning methods. Their main goal is to enable agents to make complex decisions and perform tasks across a wide range of domains. *Task Decomposition*, *Multiple Plan Selection*, *External Planner-Assisted Planning*, and *Reflection and Refinement* are key approaches that address distinct challenges in planning [7]. Together, these strategies enhance the robustness, flexibility, and scalability of LLM agents.

- Task Decomposition allows agents to break tasks into manageable sub-tasks. In LLM agents, this involves the model's ability to sequentially identify and plan these sub-tasks, improving clarity and reducing cognitive load for each

planning step. We can distinguish two different methods. *Decomposition-first* method involves breaking down sub-tasks at the beginning, allowing planning for each sub-task in isolation. *Interleaved decomposition method* offers a dynamic approach, allowing interleaved task decomposition and sub-tasks planning where each decomposition identifies only one or two immediate sub-tasks.

- **Multiple Plan Selection.** LLMs agents can generate multiple solutions to solve complex tasks. This process involves two key steps: Multiple Plan Generation and Multiple Plan Selection. *Multiple Plan Generation* employs diverse strategies to produce a set of candidate plans, ensuring coverage of potential approaches. *Multiple Plan Selection* mitigates variability by evaluating these candidates through an optimal selection mechanism, guaranteeing that the most effective plan is executed. Together, these steps enhance robustness and reliability in task resolution by systematically exploring and refining solutions.
- **Planning Assisted by External Planners.** Integrating LLMs with external planners, both symbolic (rule-based systems that use predefined logic) and neural (data-driven models that learn patterns), can significantly enhance feasibility and efficiency, especially in dynamic or complex environments. *Symbolic planners* improve task feasibility by following strict rules and structured reasoning, ensuring reliability in predictable settings. However, they struggle in unfamiliar scenarios where rules are missing or incomplete. *Neural planners*, on the other hand, adapt well to specific tasks by learning from examples, but they falter when faced with situations outside a specific context.
- **Reflection and Refinement.** LLMs benefit significantly from reflection and refinement mechanisms. These techniques mirror principles of reinforcement learning but focuses on refining outputs through prompt-based feedback rather than parameter updates, enabling continuous improvement without retraining. Reflection and refinement are critical for overcoming challenges like hallucinations (generating incorrect or nonsensical outputs) and reasoning gaps (incomplete or flawed logic). By systematically analyzing failures and adapting strategies, LLM agents improve errors tolerance, adaptability, and overall robustness in planning tasks.

8.2.3 Tools

Equipping LLMs with external tools enhances their ability to interact with external environments, enabling them to execute complex workflows and obtain necessary information beyond their inherent capabilities.

The entire process of utilizing tools to assist LLM agents can be handled in different ways. Here we introduced an iterative and structured pipeline that can be summarized in four distinct steps [20]: (a) task planning, (b) tool retrieval, (c) tool selection, and (d) tool execution.

Before executing the pipeline, it is crucial to clearly define the target of the LLM's action to identify the specific task to be solved. Once the intent is understood,

the LLM requires an action plan to achieve the predefined goal. After formalizing the plan (*a*), often aided by the planning module (c.f. Sect. 8.2.2), the next step is to translate the plan into concrete tasks and align these tasks with the available tools for solving the problem (*b*). Subsequently, the most appropriate tool is selected (*c*) based on its description, and all the parameters for its execution are configured. After the tool execution (*d*), the model collects feedback on the results obtained. If a tool fails, the system can activate a fallback mechanism. This involves selecting an alternative tool from the tool-retrieval module (*b*) and continuing the workflow from that point.

By adopting the described pipeline, the agent can handle disparate categories of tools.

In this context, agents simulate human behavior in orchestrating and coordinating tools. It is also worth highlighting how task execution methods can be categorized into code-based and UI-based approaches [12], each offering distinct capabilities and facing unique challenges. Code-based methods rely on generating code to interact with APIs, databases, or other components, often employing LLMs for tasks like slot-filling and program synthesis. These methods are effective for structured and predefined API-driven tasks but encounter limitations in handling complex scenarios, scaling to large toolsets, and adapting to unseen domains. On the other hand, UI-based approaches focus on simulating human interactions with device interfaces, enabling versatile automation without depending on APIs, through text-based representations like HTML or multimodal representations combining visual and textual inputs. Both Code-based and UI-based methods can be categorized by different functionalities [26]. These include *text tools*, which extract relevant information from text documents; *database tools*, which manage and query tabular data; *math tools*, which handle computations ranging from basic arithmetic to complex operations; *graph tools*, which analyze graph structures and attributes; *code tools*, which execute code and enable interoperability between tools; and *system tools*, which finalize task results based on execution feedback. This classification highlights the diverse features required for effective task execution and emphasizes the importance of selecting the appropriate tools to match the specific needs of executing a task.

The use of tools introduce also significant challenges [16]. Deciding when to invoke a tool is very important, since doing it in the wrong way can slow down the process and increase resource consumption, while an absence of invocation can lead to incomplete or inaccurate results. Another challenge is related to selecting the most appropriate tool. LLMs often have a wide range of tools at their disposal, but choosing the right one is not always straightforward. Even when the right tool is selected, the LLM must know how to invoke it correctly [9]. Each tool has its own interface and requires specific parameters. Errors in formatting parameters or understanding the API can lead to failures. For example, an API that requires a date in ISO format may fail if the model provides a different format. A critical aspect is to ensure the robustness of the process, avoiding errors accumulating [2]. Because complex tasks require multiple steps, an initial failure can have a cascading effect on all subsequent steps.

8.3 LLM-Based Multi-agents (LLM-MA)

Due to the impressive results of using LLMs as autonomous agents to do many tasks automatically, the idea of using multiple LLMs to accomplish complex problem-solving tasks has been proposed to leverage the collective intelligence, specialized profiles, and skills of multiple LLM-based agents.

Compared to systems using a single LLM-based agent, multi-agent systems offer advanced capabilities by specializing LLMs into various distinct agents, each with different capabilities, and enabling interactions among these diverse agents to simulate complex real-world environments effectively [4]. In this context, multiple autonomous agents collaboratively participate in planning, discussions, and decision-making, similarly to the cooperative nature of human group work in problem-solving tasks. The interaction among the agents leverages the communication capability of LLMs, which are able to comprehend and generate texts of disparate domains.

In particular, in a multi-agent system, each agent is assigned a distinct role and relevant expertise, making it specifically responsible for different tasks; in addition, the agents can communicate with each other and share the progress/information as the task proceeds. Typically, agents can work collaboratively (i.e., by working on different sub-tasks to achieve a final goal) or competitively (i.e., by working on the same task while debating adversarially).

The main operational framework of these types of systems focuses on four key aspects: the environment interface, the profiling, the communication, and the capability acquisition [4]. These aspects are represented in Fig. 8.2, which depicts

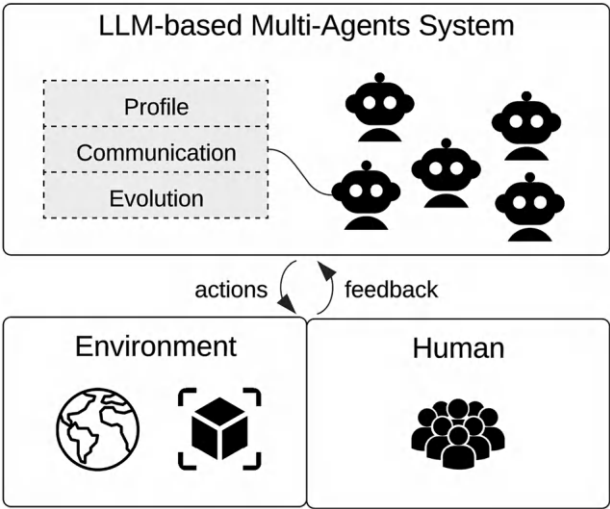


Fig. 8.2 LLM-based multi-agents architecture

the general architecture of this type of system and will be further described in the sections below.

8.3.1 *Environment Interface*

The environment interface defines the setting in which the LLM-MA system perceives and interacts, shaping agent behavior and decision-making. The authors in [4] categorize these interfaces into three types: *Sandbox*, *Physical*, and *no specific external environment*.

The *Sandbox* environment refers to a simulated or virtual space where agents interact freely and experiment with various actions and strategies without real-world consequences. The *Physical* environment involves real-world interactions, where agents engage with tangible entities and operate under physical constraints. The third category, *no specific external environment*, applies to settings where agents primarily interact with each other rather than with an external system.

These environments are applied across various domains depending on the nature of agent interaction. The *Sandbox* environment is widely used in software development, gaming, and experimental simulations, enabling agents to explore different strategies in controlled settings. Examples include game-based learning environments and AI-driven software engineering platforms. The *Physical* environment includes robotics and industrial automation, where agents interact with sensors, actuators, and other physical components to perform real-world tasks. In contrast, scenarios without a specific external environment include debating agents [8] and simulated social interactions [14], where agent behavior is defined entirely through internal knowledge exchange and inter-agent communication.

8.3.2 *Profiling*

The agents interacting in a LLM-MA system are defined by their roles (or profiles) specifying their characteristics, capabilities, behavior, and constraints, which are personalized to meet specific goals. The profiles of the agents are crucial, as they define the interactions and effectiveness in their respective environments. The authors in [4] classify agent profiling methods into three main types: *pre-defined profiles*, *model-generated profiles*, and *data-derived profiles*.

Pre-defined profiles consist of manually designed roles where agents have fixed characteristics, responsibilities, and constraints. This approach is useful for structured environments where agent behavior needs to be explicitly defined, such as in software engineering or policy simulations. *Model-generated profiles* allow LLMs to dynamically generate agent attributes based on context and requirements. This enables flexible role assignments and allows agents to adapt their behavior based on

real-time interactions and external changes. *Data-derived profiles* rely on existing datasets to infer agent characteristics, facilitating domain-specific behaviors.

Beyond role assignment, profiling also defines inter-agent relationships and communication structures. LLM-MA systems can be designed with different hierarchical structures, ranging from decentralized peer-to-peer networks to centralized agent coordination models. These structures, which will be further discussed in the next section, impact collaboration efficiency and decision-making processes within the system [4].

Additionally, agents within LLM-MA can exhibit cognitive evolution, meaning their profiles can be updated over time based on experience, interactions, and learning feedback. This allows for a more adaptive and evolving multi-agent ecosystem, where agents refine their expertise and improve task performance through iterative learning cycles [10].

8.3.3 Communication

Inter-agent communication is a fundamental aspect of multi-agent collaboration, enabling agents to exchange information, coordinate actions, and resolve conflicts. LLM-MA employ different communication paradigms depending on the nature of their interactions and the complexity of their tasks [4].

Communication within these systems can take *cooperative*, *debate-based*, or *competitive* forms, each serving a distinct role in agent coordination.

Cooperative communication involves agents working together toward shared objectives, where information is exchanged to improve collective decision-making. This paradigm is particularly useful in collaborative problem-solving scenarios, such as software development or scientific research, where multiple agents contribute distinct expertise to achieve a common goal.

Debate-based communication allows agents to engage in structured argumentative discussions, evaluating multiple perspectives before reaching a consensus. This approach is frequently utilized in policy simulation and automated negotiation systems, where diverse viewpoints must be reconciled.

Competitive communication fosters adversarial interactions, where agents pursue individual strategies that may be in conflict with others. This is commonly seen in gaming and strategic planning applications.

These communication paradigms align with different structural models of agent interaction [10], as depicted in Fig. 8.3. In a *single-agent* setting, an LLM interacts with external tools but does not communicate with other agents. This model is suited for tasks where a single agent can operate autonomously while leveraging external resources. In a *centralized* setup, a single agent oversees coordination and decision-making, managing communication with subordinate agents. This structure is effective in cooperative environments where efficiency and control are prioritized. A variation of this model, *centralized (as tools)*, treats multiple agents as external tools controlled by an LLM, balancing modularity with oversight while allowing

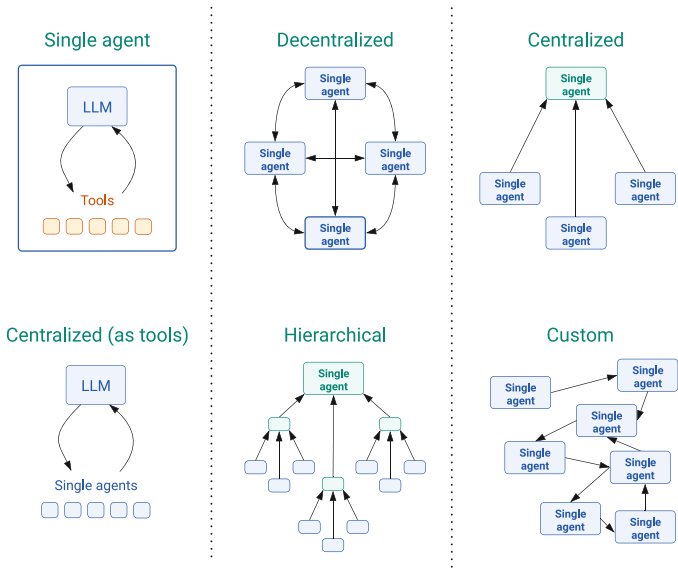


Fig. 8.3 Different types of communication in LLM-MA systems

specialization within a structured framework. *Decentralized* architectures distribute communication across multiple agents interacting in a peer-to-peer manner, without a central coordinator. This approach supports debate-based and competitive communication, enabling greater adaptability and resilience in distributed environments. A more structured variation is the *hierarchical* model, where high-level agents oversee lower-level agents, combining structured oversight with task-specific autonomy. This is particularly useful in complex systems requiring multi-level decision-making. Finally, *custom* architectures combine elements of these models, adapting communication structures to specific use cases where hybrid interaction strategies are necessary.

8.3.4 Evolution

Agent evolution in LLM-MA refers to the ability of agents to dynamically improve, adapt, and refine their behavior over time based on experience, interactions, and external feedback. This process enhances multi-agent collaboration by allowing agents to learn from prior interactions, optimize their strategies, and develop new capabilities autonomously [4].

The evolution process can be categorized into three primary mechanisms: *memory-based learning*, *self-adaptation*, and *dynamic agent generation*.

Memory-based learning allows agents to store past interactions and retrieve relevant experiences to inform future decisions, ensuring contextual consistency and improved response efficiency. *Self-adaptation* enables agents to refine their strategies based on ongoing interactions, modifying their behavior in real time to enhance coordination and task execution. *Dynamic agent generation* introduces new agents as required, adjusting system complexity and specialization levels to optimize performance in evolving environments [10]. Reinforcement learning techniques [18] and iterative training mechanisms [13] further contribute to the evolution of LLM-based agents. These techniques enable agents to continuously improve their problem-solving abilities, adapt to new challenges, and fine-tune decision-making processes. The integration of feedback loops (e.g., [22]), both from other agents and external sources, ensures a more robust and scalable multi-agent system that can handle complex, real-world tasks effectively.

8.4 Applications of LLM Agents in Software Engineering

This section reviews some applications of LLM agent-based systems in software engineering. These systems have been applied across various stages of software development, including requirements analysis, system design, code generation, testing, and documentation.

Software engineering typically involves complex, interdependent tasks that traditionally require extensive human oversight. LLM-MA systems can address these challenges by structuring workflows, enabling agents to collaborate, validate outputs, and refine software artifacts iteratively. The following sections explore different frameworks, starting with structured systems that enforce predefined workflows, followed by agile-oriented approaches that enable dynamic adaptation. The discussion concludes with a look at autonomous multi-agent frameworks and their potential for scalable, self-sufficient software development.

Frameworks for Software Engineering

Several frameworks that leverage multi-agent collaboration to enhance software development have been recently proposed in the literature.

MetaGPT [6] is one such framework that introduces a structured approach to software engineering by assigning distinct roles to different agents. Instead of treating the development process as a single monolithic task, MetaGPT organizes it into multiple specialized roles, such as a product manager for gathering requirements, an architect for designing the system, an engineer for writing the code, and a QA engineer for testing and debugging. A defining feature of MetaGPT is its use of predefined workflows that guide each agent's responsibilities. This structure prevents agents from engaging in unstructured interactions that could lead to inconsistencies or unnecessary repetition. The development process follows a structured sequence, where each agent builds on the output of the previous one. For example, the product manager agent first produces a requirements document that serves as the foundation for system design. The architect then refines this document into technical

specifications, which the engineer uses to generate code. The QA engineer later tests the implementation against the specifications to identify any issues. This ensures that agents exchange information in a structured manner, potentially minimizing misunderstandings and preventing errors from propagating through the pipeline. Unlike multi agent frameworks that rely on free-form conversation between agents, MetaGPT enforces a step-by-step approach where each agent's output adheres to a predefined format. This method reduces ambiguities, maintains consistency across development phases, and ensures that generated software adheres to well-defined specifications.

Another framework, ChatDev [15], takes a different approach by modeling a virtual software company where agents communicate through structured dialogues. The development process is divided into four sequential phases: designing, coding, testing, and documenting. Each phase involves agents with distinct roles, such as architects, programmers, and testers, who engage in multi-turn interactions to refine requirements, generate implementations, and validate outputs. ChatDev enables agents to verify each other's contributions, reducing the risk of errors in generated software. If a programmer agent produces an implementation, a peer reviewer checks the code for correctness before passing it to a tester for validation. The system follows a structured dialogue model where agents justify their decisions, propose modifications, and iterate on solutions as needed. This collaborative verification process helps mitigate hallucinations in LLM-generated code by ensuring that multiple agents reach a consensus before finalizing an output.

Agile Software Engineering with LLM Agents

Some frameworks focus on integrating multi-agent coordination into agile software engineering methodologies. CogniSim [3] is an example of a multi-agent system designed to enhance agile workflows, particularly in the Scaled Agile Framework (SAFe). The system consists of autonomous agents that participate in sprint planning, backlog refinement, and iterative software development. Each agent contributes specialized expertise at different stages of the development cycle, supporting tasks such as prioritizing backlog items, optimizing workload distribution, and tracking project progress. The use of agents allows for continuous learning and adaptation. Instead of relying on static decision-making rules, these agents analyze past development cycles, evaluate performance metrics, and adjust their strategies accordingly. A backlog refinement agent, for instance, may learn from previous sprints to improve its prioritization of user stories, while a sprint planning agent can modify workload assignments based on real-time team capacity. These adaptations can help agile teams manage changing requirements more effectively and optimize resource allocation throughout the development process.

From Multi-agent Coordination to Fully Autonomous Software Engineering

While existing frameworks have demonstrated significant progress in structuring and optimizing software development workflows, full autonomy remains an open challenge. Current systems, such as MetaGPT and ChatDev, rely on predefined workflows and structured communication, while agile-oriented frameworks like CogniSim introduce adaptability through iterative refinement. However, these

approaches might still require human intervention to define objectives, interpret ambiguous requirements, and address unexpected failures. Moving toward fully autonomous software engineering requires advances in three key areas, as outlined in the survey in [5]: *independent decision-making*, *adaptive planning*, and *long-term learning*.

Independent decision-making remains a challenge for LLM agent systems. While current frameworks distribute tasks across agents, decisions about ambiguous or conflicting requirements might still require human oversight. Achieving full autonomy requires mechanisms that allow agents to debate, negotiate, and reach consensus without explicit human input.

Adaptive planning is another critical factor in autonomous software engineering. Existing frameworks follow structured workflows, which, while effective, may struggle when encountering novel scenarios that require deviating from predefined processes. Future research must focus on enabling agents to dynamically adjust their workflows based on real-time feedback to allow LLM agent systems to generalize beyond their initial training data.

Long-term learning and knowledge retention are also essential for achieving sustained autonomy. Fully autonomous systems must incorporate mechanisms for continuous learning, where agents retain and refine knowledge over multiple projects, enabling incremental improvements in software development capabilities. Techniques such as lifelong learning [24] may play a crucial role in this transition.

Achieving fully autonomous software engineering will require further advancements in multi-agent reasoning, uncertainty handling, and ethical considerations in AI-driven decision-making. While current LLM agent systems offer structured workflows and iterative adaptation, future developments will need to address independent goal setting, dynamic problem-solving, and self-improvement mechanisms to reduce reliance on human oversight.

8.5 Programming LLM Agents

In this section, we explore the implementation of LLM agents and compare various approaches. While foundational libraries like Hugging Face Transformers² and LangChain³ facilitate building agents from scratch, recent frameworks have emerged to facilitate the development of LLM agents and complex workflows involving LLM-MA. Notable examples include *LangGraph*,⁴ *CrewAI*,⁵ and *OpenAI Swarm*.⁶

² <https://huggingface.co/docs/transformers>.

³ <https://www.langchain.com/langchain>.

⁴ <https://www.langchain.com/langgraph>.

⁵ <https://www.crewai.com>.

⁶ <https://github.com/openai/swarm>.

LangGraph is part of the LangChain ecosystem and offers a robust framework for constructing structured multi-agent workflows. It models interactions as directed graphs, with nodes representing agents or tasks and edges defining the flow of information between them. LangGraph excels in managing complex decision-making processes and supporting parallel task execution, making it particularly useful for applications requiring precise control over agent orchestration, such as dynamic task pipelines or modular workflows.

CrewAI adopts a human-centric approach to multi-agent systems, focusing on collaboration through the concept of “crews,” i.e., specialized teams of agents working collectively toward a shared objective. With features like role-based management, task delegation, and seamless inter-agent communication, CrewAI simplifies development by aligning agent interactions with familiar organizational paradigms. Its design is especially effective for projects requiring intuitive role assignment and efficient coordination between agents.

OpenAI Swarm is an experimental, educational framework designed to explore lightweight multi-agent orchestration. It focuses on making agent coordination and execution controllable and testable through two primary abstractions: agents and handoffs. An agent encompasses instructions and tools and can choose to hand off a conversation to another Agent. These primitives enable the construction of real-world solutions while avoiding a steep learning curve.

In the following steps, we will first implement an LLM agent at a low level using Hugging Face Transformers to understand the foundational concepts and mechanics. Then, we will leverage the high-level interfaces of the previously mentioned high-level libraries, specifically LangGraph, to ease the implementation of LLM agents and LLM-MA.

8.5.1 Implementing a Simple LLM Agent from Scratch Using Hugging Face Transformers

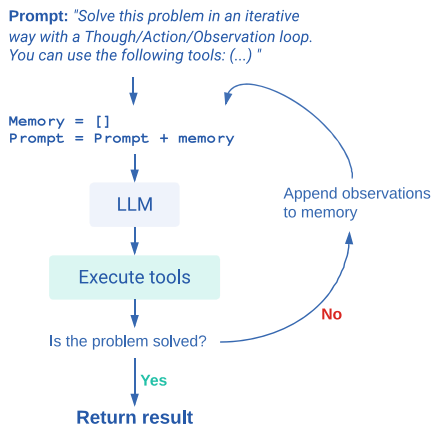
In this section, we will implement a simplified ReAct agent [19] from scratch using Hugging Face Transformers. A ReAct agent operates by iteratively alternating between *reasoning* and *acting* to accomplish a task.

During the *reasoning* stage, the agent analyzes the current state of the environment, processes the context, and formulates a plan based on its understanding of the task. This stage focuses on logical deduction and decision-making. During the *acting* stage, the agent executes the planned actions, interacts with the environment, and collects feedback, which it uses to refine subsequent *reasoning* steps. We provide a summary of the inner workings of a ReAct agent in Fig. 8.4.

For simplicity, this implementation will use a single tool: a calculator, allowing the agent to perform basic mathematical operations as part of its actions.

We will start by importing the libraries needed for this implementation of an LLM agent and setting up the device (either CPU or GPU) for inference.

Fig. 8.4 Overview of the inner workings of ReAct



Import the required libraries

```

1 from transformers import LlamaTokenizer, MistralForCausalLM,
  ↳ StoppingCriteria, StoppingCriteriaList
2 import json
3 import re
4 import torch
5
6 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

```

Next, we define the LLM to use as the agent and its tokenizer. We will use OpenHermes-2.5-Mistral-7B, which is a fine-tuned variant of the Mistral architecture trained on additional code datasets, which is known for its agent capacities.

Define the model and its tokenizer

```

1 tokenizer = LlamaTokenizer.from_pretrained(
2     "teknium/OpenHermes-2.5-Mistral-7B",
3     trust_remote_code=True
4 )
5
6 model = MistralForCausalLM.from_pretrained(
7     "teknium/OpenHermes-2.5-Mistral-7B",
8     torch_dtype=torch.float16,
9     device_map=device,
10    load_in_8bit=False,
11    load_in_4bit=True,
12    use_flash_attention_2=False,
13    low_cpu_mem_usage=True
14 )

```

The model is loaded with quantization and memory optimization parameters to balance efficiency and performance. By using `torch_dtype=torch.float16`

(line 8), weights are stored in 16-bit floating-point format, reducing memory usage while maintaining precision. The `load_in_4bit=True` (line 11) enables 4-bit quantization, significantly lowering the memory footprint, making it suitable for resource-constrained environments. Additionally, `low_cpu_mem_usage=True` (line 13) minimizes memory use during loading, and `device_map=device` (line 9) ensures the model is loaded onto the most capable hardware (GPU or CPU).

The ReAct agent uses tools to perform specific tasks. For simplicity, we include a single tool: a calculator. This tool evaluates mathematical expressions and returns the result. We provide a simple (and naive) implementation of a calculator function below:

Define the calculator tool

```

1 def calculator(expression):
2     try:
3         result = eval(expression)
4         return f"The result is {result}."
5     except Exception as e:
6         return f"Error in calculation: {e}"
7
8 TOOLS = {
9     "Calculator": {"function": calculator, "parameters":
10         ↪ "expression (a mathematical expression to evaluate)"}
11 }
12
13 TOOL_NAMES = list(TOOLS.keys())

```

We will now define the ReAct agent as a Python class.

Define the React Agent

```

1 class ReActAgent:
2     def __init__(self, model, tokenizer, tools):
3         self.model = model
4         self.tokenizer = tokenizer
5         self.tools = tools

```

We will now define below a simple variant of the ReAct prompt to allow the LLM to generate structured reasoning and use tools to solve tasks. The `format_prompt` method below is used to generate the prompt that will be fed to the model.

Define the React Agent prompt

```

1 def format_prompt(self, question):
2     """Construct the exact prompt template with tool descriptions"""
3     tools_description = "\n".join(
4         [f"- {tool}: {desc['parameters']}" for tool, desc in
5         ↪ self.tools.items()]
6     )

```

```

6  system_prompt = f"""Answer the following questions as best you can. You
   ↪ have access to the following tools:
7
8  {tools_description}
9
10 The way you use the tools is by specifying a json blob.
11 Specifically, this json should have a `action` key (with the name of
   ↪ the tool to use) and a `action_input` key (with the input to the
   ↪ tool going here).
12 The only values that should be in the "action" field are: {TOOL_NAMES}
13 The $JSON_BLOB should only contain a SINGLE action and MUST be
   ↪ formatted as markdown, do NOT return a list of multiple actions.
   ↪ Here is an example of a valid $JSON_BLOB:
14 ```
15 {{
16         "action": $TOOL_NAME,
17         "action_input": $INPUT
18 }}
19 ```
20 Make sure to have the $INPUT in the right format for the tool you are
   ↪ using, and do not put variable names as input if you can find the
   ↪ right values.
21 ALWAYS use the following format:
22 Question: the input question you must answer
23 Thought: you should always think about one action to take. Only one
   ↪ action at a time in this format:
24 Action:
25 ```
26 $JSON_BLOB
27 ```
28 Observation: the result of the action
29 ... (this Thought/Action/Observation can repeat N times, you should
   ↪ take several steps when needed. The $JSON_BLOB must be formatted as
   ↪ markdown and only use a SINGLE action at a time.)
30 You must always end your output with the following format:
31 Thought: I now know the final answer
32 Final Answer: the final answer to the original input question
33
34 Now begin! Reminder to ALWAYS use the exact characters `Final Answer:`
   ↪ when you provide a definitive answer.
35 Question: {question}"""
36
37 return system_prompt

```

This prompt defines how the ReAct agent should structure its reasoning. The ReAct framework transforms the output of a language model from simple text generation into a process of structured reasoning and action. The structured reasoning in ReAct is achieved through three primary elements: Thought, Action, and Observation. These elements form a loop that the agent cycles through until it reaches a final answer. The “Thought” encapsulates the agent’s reasoning, the

“Action” specifies a concrete step to execute, and the “Observation” reflects on the result of that step to guide the next reasoning process.

Let’s explore this through an example of solving a mathematical problem: “What is the result of $5 + 2$?” When presented with this task, the agent begins by analyzing the problem and thinking how to decompose the problem into smaller problem. In this simple case, the problem can be solved in a single step by calling the calculator tool.

Reasoning/thinking in ReAct

Question: What is the result of $5 + 2$?

Thought: To solve this, I need to calculate the value of $5 + 2$ using
↪ the calculator.

With its plan in place, the agent decides to execute the calculation for $5 + 2$. The ReAct prompt ensures that the agent specifies this action using a JSON blob. This structured representation of the action communicates the agent’s intent to use the calculator tool:

Generating the action

```
Action:
...
{
    "action": "Calculator",
    "action_input": "5 + 2"
}
...
```

This explicit format is not just for the agent’s internal reasoning; it is designed for easy integration with external tools or APIs, ensuring that the system understands and executes the action correctly. The calculator then processes the input and returns the result, 7. The agent then observes this output and incorporates it into its reasoning:

Generating the final response

Observation: The result of the calculation is 7.

Thought: I now know the final answer.

Final Answer: 7

Here, the agent completes its reasoning process by explicitly stating the final answer using the “Final Answer” marker.

Moving on with the implementation of the ReAct agent, we implement the `generate_response` method that invokes the model to generate text based on the formatted prompt.

Prompting the model to generate a response

```

1 def generate_response(self, prompt, stop_criteria):
2     inputs = self.tokenizer(prompt, return_tensors="pt")
3     input_ids = inputs.input_ids.to(device)
4     attention_mask = inputs.attention_mask.to(device)
5     outputs = self.model.generate(
6         input_ids,
7         attention_mask=attention_mask,
8         max_new_tokens=750,
9         temperature=0.8,
10        pad_token_id=tokenizer.eos_token_id,
11        repetition_penalty=1.1,
12        do_sample=True,
13        stopping_criteria=stop_criteria
14    )
15    response = self.tokenizer.decode(outputs[0], skip_special_tokens=True)
16    return response

```

The `generate` method is called with several parameters that control the behavior of the model during text generation. The `max_new_tokens` parameter (line 8) limits the number of tokens the model can generate, ensuring responses remain concise and efficient. To balance creativity and coherence, the `temperature` parameter (line 9) is set to 0.8, introducing some randomness to the output without sacrificing relevance. The `repetition_penalty` (line 11) discourages repetitive phrases by penalizing the reuse of tokens, enhancing the diversity of the generated text.

The last parameter is the `stopping_criteria` (line 13), which we use to prevent the model from going all the way through the final answer without calling the calculator function. In this case, we want to stop generating text right after the model generates the action. A simple approach consists of stopping the generation once the model generates the “Observation:” sequence, which is generated right after the action. The code below implements general stopping criteria based on a specific text sequence:

LLM agent stopping criteria implementation

```

1 class StopOnObservation(StoppingCriteria):
2
3     def __init__(self, target_sequence, prompt):
4         self.target_sequence = target_sequence
5         self.prompt = prompt
6         self.last = None
7         self.new_token = ''
8
9     def __call__(self, input_ids, scores, **kwargs):
10        generated_text = tokenizer.decode(input_ids[0])
11        if self.last:
12            self.new_token += generated_text.replace(self.last, '')
13        if self.target_sequence in self.new_token:

```

```

14 return True
15 self.last = generated_text
16 return False

```

We now explore how the ReAct framework prompts the model to iteratively solve a query. The process is implemented in the `interact` method, which guides the agent through reasoning, action, and observation loops until a final answer is reached. Below is the code for this method:

Implementation of the ReAct workflow

```

1 def parse_json_blob(self, response):
2     """Extract the JSON blob related to an action from the model's
   ↳ response."""
3     # Locate the "Action:" keyword in the response
4     match = re.search(r"Action:\s*(?:``)?\s*({.*?})\s*(?:``)?", response,
   ↳ re.DOTALL)
5     if match:
6         json_str = match.group(1).strip() # Extract the JSON object string
7         try:
8             return json.loads(json_str)
9         except json.JSONDecodeError as e:
10            print(f"JSON parsing error: {e}")
11            print(f"Malformed JSON: {json_str}")
12            return None
13            print("No valid action JSON found in response.")
14            return None
15
16
17 def interact(self, question):
18     """Answer the question iteratively using the structured ReAct
   ↳ process."""
19     # generate the initial prompt
20     initial_prompt = self.format_prompt(question)
21
22     while True:
23         # generate response
24         stop_criteria = StopOnObservation("Observation:", self.tokenizer)
25         response = self.generate_response(initial_prompt, stop_criteria)
26
27         # mask the initial prompt from the current generation
28         new_content = response.replace(initial_prompt, "").strip()
29
30         # process the new content line by line and find the final answer or
   ↳ action
31         has_action = False
32         final_answer = None
33         for line in new_content.splitlines():
34             if line.startswith("Action:"):
35                 has_action = True
36             elif line.startswith("Final Answer:"):

```

```

37 final_answer = line[len("Final Answer:"):].strip()
38
39 # handle final answer immediately
40 if final_answer:
41     return final_answer
42
43 # parse the action
44 if has_action:
45     try:
46         action_json = self.parse_json_blob(new_content)
47     except json.JSONDecodeError:
48         return "I am unable to answer the question."
49     else:
50         return "I am unable to answer the question."
51
52 if action_json is None:
53     return "I am unable to answer the question."
54
55 # execute the action
56 tool_name = action_json.get("action")
57 tool_input = action_json.get("action_input")
58
59 if tool_name in self.tools:
60     result = self.tools[tool_name]["function"](tool_input)
61     new_content += f"{result}"
62 else:
63     new_content += "\nFinal Answer: I am unable to answer the question."
64     return "I am unable to answer the question."
65
66 # update the initial prompt to include the latest reasoning
67 initial_prompt += f"\n{new_content}"

```

The `interact` method begins by generating an initial prompt using the `format_prompt` function (line 20), which structures the user's question alongside the context needed for reasoning. This prompt is then passed to the `generate_response` function (line 25), which uses the model to produce a response adhering to the ReAct framework.

The response generated by the model can take one of two forms, depending on whether the `StopOnObservation` stopping criteria (line 24) is triggered or not: it may contain a final answer, signaling that the reasoning process is complete, or it may include an action, indicating that further steps are needed to resolve the query. If the response includes a final answer, the method immediately terminates, returning this answer to the user. This ensures that the agent stops reasoning as soon as the task is fully resolved.

If the response contains an action, the agent proceeds to parse it (line 46) using the `parse_json_blob` method (line 1). This step extracts the necessary details from the JSON representation of the action, including the tool to be used and its input. For example, if the response suggests using the calculator tool to compute $5 + 2$, the method extracts the tool name (Calculator) and the input ($5 + 2$). The agent then

executes the action by invoking the appropriate tool (line 60), such as the calculator, which returns the result of the operation.

Once the tool has provided its result, the agent updates its prompt by appending the new reasoning, action, and observation. This updated prompt forms the basis for the next iteration, as it provides the model with a more detailed context to refine its reasoning further. The process of reasoning, acting, and observing continues iteratively, with the agent building upon its previous steps until it arrives at a final answer.

With the ReAct agent constructed, we can now process user queries interactively, using the following code:

ReAct agent instantiation and execution

```
1 if __name__ == "__main__":
2     agent = ReActAgent(model, tokenizer, TOOLS)
3     print("ReAct Agent Initialized. Type 'exit' to quit.\n")
4
5     while True:
6         question = input("Your Question: ")
7         if question.lower() == "exit":
8             print("Exiting...")
9             break
10
11     answer = agent.interact(question)
12     print(f"Agent's Answer: {answer}\n")
```

While this simple implementation of the ReAct agent provides an intuition of how LLMs can use iterative reasoning and external tools, it has notable limitations that highlight the need for higher-level frameworks. A key challenge is the explicit stopping of text generation to identify tool invocations, execute actions, and then manually resume reasoning with an updated prompt. Additionally, the stopping criteria rely on hardcoded sequences, such as detecting “Observation:” in the output, which can be unreliable and may fail if the model generates unexpected variations. Moreover, tools must be defined manually with specific input and output structures, placing a significant burden on developers to design and integrate new functionalities. These limitations can make scaling the agent to more complex tasks or diverse toolsets challenging.

High-level abstractions, such as those offered by LangChain and LangGraph, address these issues by offering an ecosystem of diverse tools, agentic frameworks (e.g., ReAct), and even the ability to design complex flows involving LLM-MA.

8.5.2 Implementing an LLM Agent Using Langchain

We will now discuss how to implement a ReAct agent with the high-level abstraction provided by the Langchain and the Langgraph libraries. We will leverage their

ecosystem of tools to create an LLM Agent that can search the Web and Wikipedia to answer questions.

We will first start by implementing the ReAct agent using LangChain. We will start by importing the necessary libraries.

Import the required libraries

```
1 from langchain import hub
2 from langchain.agents import AgentExecutor, create_react_agent
3 from langchain.agents import load_tools
4 from langchain_openai import ChatOpenAI
5 import os
6
7 if "OPENAI_API_KEY" not in os.environ:
8     os.environ["OPENAI_API_KEY"] = "your-openai-key"
9
10 if "SERPAPI_API_KEY" not in os.environ:
11     os.environ["SERPAPI_API_KEY"] = "your-serapi-key"
```

The `hub`, `AgentExecutor`, `create_react_agent`, and `load_tools` modules from LangChain are used to manage prompts, construct the agent, and load tools. LangChain's ecosystem supports a wide variety of tools, such as Web search, Wikipedia, calculators, and APIs, enabling seamless integration with external resources. Additionally, we import `ChatOpenAI` from `langchain_openai` to instantiate our LLM.

We will use OpenAI's GPT-4o-mini as our agent in this example. To set up the model, we need to set the `OPENAI_API_KEY` for accessing the model. We will also set the `SERPAPI_API_KEY` for using `SerpApi`, which is an API to access Google search results, which we will expose as one of the tools for our agent.

The next step consists of loading the tools. As mentioned before, we will load two tools for this example: `SerpApi`, for searching the Web, and Wikipedia. Both tools are integrated into the Langchain ecosystem and can be defined using the `load_tools` function from Langchain, which conveniently wraps them as tools to be used by an LLM agent:

Define the list of tools

```
1 tools = load_tools(["serpapi", "wikipedia"])
```

We then define the prompt that will guide the agent's reasoning process.

Pull the ReAct prompt from the LangChain Hub

```
1 prompt = hub.pull("hwchase17/react")
```

The LangChain ecosystem includes a repository called the LangChain Hub, where prompts, models, and other reusable resources are stored and shared. Using

the `hub.pull` method, we retrieve the react prompt designed for the ReAct framework.

The LLM is selected next. We use the `ChatOpenAI` class to instantiate a model, specifying `gpt-4o-mini` as the version.

Define the model

```
1 llm = ChatOpenAI(model="gpt-4o-mini")
```

With the tools, prompt, and LLM in place, we construct the ReAct agent using the `create_react_agent` function.

Create the ReAct agent

```
1 agent = create_react_agent(llm, tools, prompt)
```

Finally, we create the `AgentExecutor`, which serves as the execution engine for the agent. Passing the constructed agent and tools to the `AgentExecutor` and setting `verbose=True`, we enable detailed logging of the agent's intermediate steps. This is useful for debugging and understanding how the agent reasons through tasks, as it provides a complete view of the reasoning, action, and observation loop.

Wrap the agent for subsequent execution

```
1 agent_executor = AgentExecutor(agent=agent, tools=tools, verbose=True)
```

The ReAct agent can then be queried using the `.invoke` method of `AgentExecutor`:

Run the agent

```
1 agent_executor.invoke({"input": "who is Massimo Mecella?"})
```

Here's an excerpt of the output from the execution of the ReAct agent:

Reasoning/thinking in ReAct

```
I should start by looking up Massimo Mecella on Wikipedia to gather
↪ relevant information about him.
```

```
Action: Wikipedia
```

```
Action Input: Massimo Mecella
```

```
... (continue)
```

```
There doesn't seem to be relevant information about Massimo Mecella
↪ directly from the initial search results on Wikipedia. I might need
↪ to look deeper or consult another source.
```

```
Action: Search
Action Input: Massimo Mecella

... (continue)

I found relevant information about Massimo Mecella, including his
↳ educational background, research interests, and current position.

Action: Search
Action Input: Massimo Mecella Sapienza University of Rome profile

... (continue)

I now have a comprehensive profile of Massimo Mecella, detailing his
↳ education, current position, and research interests.

Final Answer: Massimo Mecella is an associate professor at Sapienza
↳ University of Rome, where he specializes in Information Systems
↳ Engineering. He was born in Rome, Italy, in 1973 and holds a PhD in
↳ Computing Science and Engineering from the same university. His
↳ research interests include business process management, software
↳ engineering, and various information systems applications.
```

The agent first attempts to look up details about a person named Massimo Mecella on Wikipedia but finds no relevant results. It then issues a general Web search to gather information, retrieving details about a person named Massimo Mecella who works at Sapienza University of Rome. To refine its understanding, the agent formulates a more specific search query targeting his professional profile at the university. After retrieving this additional information, it synthesizes the results into a final answer about Massimo Mecella, including his academic background, research interests, and current position.

Compared to the previous implementation (cf. Sect. 8.5.1) where we had to explicitly handle passing the prompt to the model, pausing generation to detect tool usage, calling the tools, and then resuming the reasoning process, the combined use of `create_react_agent` and `AgentExecutor` abstracts away much of this complexity. In fact, the `create_react_agent` function automatically integrates the tools and the prompt with the LLM. The `AgentExecutor` instead transparently manages the process of stopping generation when an action (e.g., a tool invocation) is detected and then resumes reasoning seamlessly after the output from the tool is obtained.

8.5.3 Implementing LLM-MA with LangGraph

We will now discuss how to implement a LLM-MA system with LangGraph. LangGraph is a Python library part of the LangChain ecosystem and is designed

to facilitate the orchestration of workflows involving stateful interactions between agents. Its foundation is the *state graph*, a directed graph where nodes represent tasks or agents and edges define the transitions between them. The state graph integrates both the logic of individual tasks and the flow of data, making it ideal for scenarios involving complex task coordination. The core components of LangGraph include:

- *Nodes*: represent tasks or agents that modify or act upon the shared state.
- *Edges*: define how tasks connect and transition from one to another. Conditional edges allow the workflow to adapt dynamically based on intermediate outcomes.
- *State*: a mutable object shared across tasks, carrying the data that evolves through the workflow.
- *Special nodes*: START and END are predefined nodes marking the beginning and conclusion of the workflow.

This graph-based abstraction provides enough flexibility to cover a variety of different profile, communication, and evolution settings, allowing LLMs to interact conditionally and adaptively and enabling precise coordination in even the most complex workflows.

We will use LangGraph in this section to create an automated research assistant that uses multiple agents to streamline the process of finding and filtering academic research papers. The workflow starts with a search agent that queries Google Scholar through the `scholarly` library. This library provides a programmatic way to interact with Google Scholar, retrieving metadata for academic papers such as titles, authors, and publication venues. Once the initial papers are retrieved, a filtering agent evaluates their relevance to the initial query. This agent analyzes the content of each paper's metadata and determines whether it aligns closely with the original search query. A supervisor agent then reviews the filtered papers. If the number of relevant papers is sufficient, the workflow concludes by providing the user with the title, authors, publication year (if available), and venue (if available) of the relevant papers. However, if the relevant papers are too few, the supervisor triggers a query refinement process. A dedicated query refinement agent then generates a more precise search query, and the process is repeated to improve the results.

We will start by importing the relevant libraries first and setting the environment variables since we will use GPT 4o-mini in this example:

Import the required libraries

```
1 import os
2 from scholarly import scholarly
3 from langchain_openai import ChatOpenAI
4 from langchain.prompts import ChatPromptTemplate
5 from langgraph.graph import StateGraph, START, END
6
7 if "OPENAI_API_KEY" not in os.environ:
8     os.environ["OPENAI_API_KEY"] = "your-openai-key"
```

Then we define the first agent, `search_agent`, which queries Google Scholar for academic papers.

Define the search agent

```

1 def search_agent(state):
2     query = state.get("query", "applications of AI in education")
3     search_results = scholarly.search_pubs(query)
4     papers = []
5     for i in range(5): # Limit to 5 results for simplicity
6         try:
7             paper = next(search_results)
8             papers.append({
9                 "title": paper['bib']['title'],
10                "authors": paper['bib']['author'],
11                "pub_year": paper['bib'].get('pub_year', 'Unknown'),
12                "venue": paper['bib'].get('venue', 'Unknown'),
13                "abstract": paper['bib'].get('abstract', 'Unknown')
14            })
15         except StopIteration:
16             break
17     state["papers"] = papers
18     return state

```

This function extracts the query (line 2) from the shared state (defaults to “applications of AI in education” if no query is provided) and uses `scholarly` to retrieve search results (line 3). The first five results are stored in a structured format (lines 8–13) containing the paper’s title, authors, publication year, and venue. The results are added to the state (line 17) for subsequent processing.

Then, the `filter_agent` evaluates the relevance of the retrieved papers by interacting with OpenAI’s GPT 4o-mini.

Define the filter agent

```

1 def filter_agent(state):
2     papers = state.get("papers", [])
3     if not papers:
4         state["filtered_papers"] = []
5         return state
6
7     llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)
8     filtered_papers = []
9
10    for paper in papers:
11        title = paper['title']
12        prompt = ChatPromptTemplate.from_messages([
13            ("system", "You are a helpful academic assistant."),
14            ("user", f"Is this paper titled '{title}' relevant to the query
15             ↪ '{state['query']}'?. Here is the abstract:\n{paper['abstract']}. \n
16             ↪ Reply with 'Yes' or 'No'."))

```

```

15 ]])
16 response = llm(prompt.format_messages())
17 if "Yes" in response.content:
18     filtered_papers.append(paper)
19
20 state["filtered_papers"] = filtered_papers
21 return state

```

This agent retrieves the list of papers from the state (line 2) and evaluates each one using GPT 4o-mini. A prompt is created for each paper (line 12), asking the model if it is relevant to the query. Papers identified as relevant are added to the filtered list (lines 17–18), which is stored in the state. If the filtered results are inadequate, the `query_refinement_agent` refines the query to improve search results, as shown in the code below:

Define the query refinement agent

```

1 def query_refinement_agent(state):
2     query = state.get("query", "")
3     feedback = state.get("feedback", "No relevant papers found.")
4     llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)
5     prompt = ChatPromptTemplate.from_messages([
6         ("system", "You are a helpful research assistant."),
7         ("user", f"Refine this query: '{query}' to improve search results.
8         ↳ Feedback: {feedback}")
9     ])
10    response = llm(prompt.format_messages())
11    state["query"] = response.content.strip()
12    return state

```

This function refines the query by using feedback and GPT 4o model. The refined query is added back to the state (line 10) for another iteration of the search process. The `supervisor_agent` decides whether the workflow should finalize or refine the query.

Define the supervisor agent

```

1 def supervisor_agent(state):
2     filtered_papers = state.get("filtered_papers", [])
3     if len(filtered_papers) <= 1: # trigger refinement if 0 or 1 paper is
4         ↳ found
5         state["decision"] = "refine"
6         state["feedback"] = f"Only {len(filtered_papers)} relevant paper(s)
7         ↳ found. Refining the query."
8     else:
9         summary = f"Found {len(filtered_papers)} relevant papers:\n"
10        for idx, paper in enumerate(filtered_papers, 1):
11            summary += f"{idx}. {paper['title']} by {' '.join(paper['authors'])}"
12        if paper['pub_year'] and paper['pub_year'] != "NA":

```

```

11 summary += f" ({paper['pub_year']})"
12 if paper['venue'] and paper['venue'] != "NA":
13     summary += f" in {paper['venue']}"
14     summary += "\n"
15     state["decision"] = "finalize"
16     state["summary"] = summary
17
18 return state

```

If the number of relevant papers is greater than one, the agent finalizes the workflow by summarizing the papers (lines 7–16), including their titles, authors, publication years, and venues. Otherwise, it triggers query refinement by updating the state with feedback (lines 4–5).

These four agents are combined into an LLM-MA system using the graph-based abstraction offered by LangGraph, with nodes representing each agent and edges defining their transitions, as shown below:

Graph-based representation of the LLM-MA system with LangGraph

```

1 graph_builder = StateGraph(State)
2 # nodes (agents)
3 graph_builder.add_node("SearchAgent", search_agent)
4 graph_builder.add_node("FilterAgent", filter_agent)
5 graph_builder.add_node("SupervisorAgent", supervisor_agent)
6 graph_builder.add_node("QueryRefinementAgent", query_refinement_agent)
7 # edges (transitions)
8 graph_builder.add_edge(START, "SearchAgent")
9 graph_builder.add_edge("SearchAgent", "FilterAgent")
10 graph_builder.add_edge("FilterAgent", "SupervisorAgent")

```

Conditional edges are added to enable dynamic routing based on the supervisor agent's decisions:

Routing logic

```

1 def supervisor_routing(state: State) -> str:
2     if state.get("decision") == "refine":
3         return "QueryRefinementAgent"
4     elif state.get("decision") == "finalize":
5         return END
6     else:
7         raise ValueError(f"Unexpected decision: {state.get('decision')}")
8     # other edges (transitions)
9     graph_builder.add_conditional_edges("SupervisorAgent",
10                                         ↪ supervisor_routing)
11 graph_builder.add_edge("QueryRefinementAgent", "SearchAgent")

```

Finally, the workflow is compiled and executed with an initial query:

Run the LLM-MA system

```

1 graph = graph_builder.compile()
2
3 initial_input = {"query": "service composition roman model"}
4
5 final_state = graph.invoke(initial_input)
6
7 if final_state.get("decision") == "finalize":
8     print(final_state.get("summary"))
9 else:
10    print("Refined Query Suggested:")
11    print(final_state.get("query"))

```

The system processes the query, either producing a list of relevant academic papers or refining the query and rerunning the system to improve the results. The output of this process is displayed below:

Sample output

```

1. Automatic Service Composition and Synthesis: the Roman Model. by D
   ↳ Calvanese, G De Giacomo, M Lenzerini (2008) in IEEE Data Eng. Bull
2. Automated service composition based on behaviors: The roman model by
   ↳ G De Giacomo, M Mecella, F Patrizi (2013) in Web services
   ↳ foundations
3. Best Service Synthesis in the Weighted Roman Model. by D Calvanese,
   ↳ A Santoso (2012) in ZEUS
4. Automatic Composition of e-Services: The "Roman" way by D Berardi

```

LangGraph also allows the visualization of the constructed workflow graph, providing a clear representation of how the nodes (agents) and edges (transitions) interact, thus helping in understanding the flow of tasks and possibly debugging the logic of the LLM-MA system, as shown in the code below:

Generating a diagram of the LLM-MA system

```

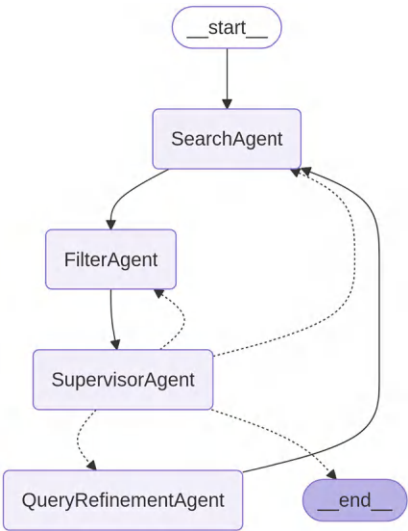
1 from IPython.display import Image, display
2 from langchain_core.runnables.graph import MermaidDrawMethod
3
4 graph_png = graph.get_graph().draw_mermaid_png(
5     draw_method=MermaidDrawMethod.PYPETEEER)
6 display(Image(graph_png))

```

The resulting graph is shown in Fig. 8.5. The `__start__` node marks the entry point, leading directly to the `SearchAgent`, which initializes the workflow. The `__end__` node signifies the conclusion of the process when sufficient results are found.

Two types of edges connect the nodes: solid edges, such as those between the `SearchAgent` and `FilterAgent`, represent the default sequential flow of

Fig. 8.5 Graph representation of the LLM-MA system built with LangGraph



tasks like searching and filtering papers. Dotted edges, such as the one from the `SupervisorAgent` to the `QueryRefinementAgent`, are conditional and reflect the dynamic nature of the workflow, triggered when the supervisor determines that query refinement is needed. The loop from the `QueryRefinementAgent` back to the `SearchAgent` ensures that the refined query is used for another search iteration.

8.6 Concluding Remarks

This chapter presented an in-depth exploration of LLM agents and multi-agent systems, highlighting their role in addressing complex and articulated tasks. While traditional LLMs are powerful in language processing, their limitations in planning, multi-step reasoning, and external interaction necessitate the development of agent-based architectures. By integrating memory, structured decision-making, and external tool usage, LLM agents enhance problem-solving capabilities and extend the applicability of AI-driven systems. Additionally, we examined the broader context of multi-agent systems, where multiple intelligent agents collaborate to achieve shared objectives. Furthermore, we explored the application of LLM agents and multi-agent systems in software engineering, demonstrating how they can assist in different tasks like code generation or testing. Finally, a practical guide on how to program LLM agents and multi-agent systems was discussed, providing insights into their implementation and deployment. As research in this area continues to evolve, overcoming challenges related to efficiency, robustness, and ethical considerations will be crucial for realizing the full potential of LLM agents in real-world applications.

References

1. Canese, L., Cardarilli, G.C., Di Nunzio, L., Fazzolari, R., Giardino, D., Re, M., Spanò, S.: Multi-agent reinforcement learning: A review of challenges and applications. *Appl. Sci.* **11**(11), 4948 (2021)
2. Chen, Z., Zhou, K., Zhang, B., Gong, Z., Zhao, W.X., Wen, J.R.: Chatcot: Tool-augmented chain-of-thought reasoning on chat-based large language models. *arXiv preprint arXiv:2305.14323* (2023)
3. Cinkusz, K., Chudziak, J.A.: Towards llm-augmented multiagent systems for agile software engineering. In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 2476–2477 (2024)
4. Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O., Zhang, X.: Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680* (2024)
5. He, J., Treude, C., Lo, D.: Llm-based multi-agent systems for software engineering: Literature review, vision and the road ahead. *ACM Trans. Software Eng. Methodol.* **34**(5), 1–30 (2025)
6. Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., et al.: Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023)
7. Huang, X., Liu, W., Chen, X., Wang, X., Wang, H., Lian, D., Wang, Y., Tang, R., Chen, E.: Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716* (2024)
8. Khan, A., Hughes, J., Valentine, D., Ruis, L., Sachan, K., Radhakrishnan, A., Grefenstette, E., Bowman, S.R., Rocktäschel, T., Perez, E.: Debating with more persuasive llms leads to more truthful answers. In: *Forty-first International Conference on Machine Learning* (2024)
9. Li, M., Zhao, Y., Yu, B., Song, F., Li, H., Yu, H., Li, Z., Huang, F., Li, Y.: Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244* (2023)
10. Li, X., Wang, S., Zeng, S., Wu, Y., Yang, Y.: A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinearth* **1**(1), 9 (2024)
11. Li, Y., Li, Z., Zhang, K., Dan, R., Jiang, S., Zhang, Y.: Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus* **15**(6), e40895 (2023). <https://doi.org/10.7759/cureus.40895>
12. Li, Y., Wen, H., Wang, W., Li, X., Yuan, Y., Liu, G., Liu, J., Xu, W., Wang, X., Sun, Y., et al.: Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459* (2024)
13. Motwani, S.R., Smith, C., Das, R.J., Rybchuk, M., Torr, P.H., Laptev, I., Pizzati, F., Clark, R., de Witt, C.S.: Malt: Improving reasoning with multi-agent llm training. *arXiv preprint arXiv:2412.01928* (2024)
14. Park, J.S., Popowski, L., Cai, C., Morris, M.R., Liang, P., Bernstein, M.S.: Social simulacra: Creating populated prototypes for social computing systems. In: *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–18 (2022)
15. Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., Sun, M.: Communicative agents for software development. *arXiv preprint arXiv:2307.07924* **6** (2023)
16. Shen, Z.: Llm with tools: A survey. *arXiv preprint arXiv:2409.18807* (2024)
17. Sumers, T.R., Yao, S., Narasimhan, K., Griffiths, T.L.: Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427* (2023)
18. Xu, Z., Yu, C., Fang, F., Wang, Y., Wu, Y.: Language agents with reinforcement learning for strategic play in the werewolf game. In: *Forty-first International Conference on Machine Learning* (2024)
19. Yao, S., Zhao, J., Yu, D., Du, N., Shafraan, I., Narasimhan, K., Cao, Y.: React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022)
20. Yuan, S., Song, K., Chen, J., Tan, X., Shen, Y., Kan, R., Li, D., Yang, D.: Easytool: Enhancing llm-based agents with concise tool instruction. *arXiv preprint arXiv:2401.06201* (2024)

21. Zhang, K., Li, J., Li, G., Shi, X., Jin, Z.: Codeagent: Enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges. *arXiv preprint arXiv:2401.07339* (2024)
22. Zhang, Y., Yang, S., Bai, C., Wu, F., Li, X., Wang, Z., Li, X.: Towards efficient llm grounding for embodied multi-agent collaboration. *arXiv preprint arXiv:2405.14314* (2024)
23. Zhang, Z., Bo, X., Ma, C., Li, R., Chen, X., Dai, Q., Zhu, J., Dong, Z., Wen, J.R.: A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501* (2024)
24. Zheng, J., Qiu, S., Shi, C., Ma, Q.: Towards lifelong learning of large language models: A survey. *arXiv preprint arXiv:2406.06391* (2024)
25. Zhu, X., Chen, Y., Tian, H., Tao, C., Su, W., Yang, C., Huang, G., Li, B., Lu, L., Wang, X., et al.: Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144* (2023)
26. Zhuang, Y., Yu, Y., Wang, K., Sun, H., Zhang, C.: Toolqa: A dataset for llm question answering with external tools. *Adv. Neural Inf. Process. Syst.* **36**, 50117–50143 (2023)

Chapter 9

Applications of Large Language Models in Information Systems



Jerin George Mathew  and Flavia Monti 

Abstract The integration of large language models (LLMs) into information systems is revolutionizing data management and business process automation. This chapter presents two complementary approaches that leverage LLMs to enhance intelligent data processing and workflow execution. By enabling the seamless integration of heterogeneous data and increasing the adaptability of business process management systems, these approaches highlight the potential of LLMs to optimize and advance information systems. Implementation examples provide practical guidelines for developing LLM-based applications for real-world scenarios, illustrating how LLMs can drive efficiency, automation, and intelligent decision-making within information systems.

9.1 Introduction

The increasing adoption of large language models (LLMs) in various domains has transformed how data is processed, retrieved, and utilized. In the context of industrial applications and business process automation, LLMs offer significant potential for enhancing the efficiency and adaptability of available application systems, also referred to as information systems.

This chapter presents two complementary works that leverage LLMs for data synthesis and process execution. The first work, *COSMADS* (COMposing SMART Data Services) [5], addresses the challenge of data accessibility and integration in industrial settings. Manufacturing environments generate vast amounts of heterogeneous data, often siloed across different systems. *COSMADS* facilitates data retrieval through the dynamic composition of information extraction pipelines in response to natural language queries. An LLM agent and a structured repository of data services and past pipelines enable efficient on-demand data processing, bridging the gap between operators and digital resources. The second work,

J. G. Mathew (✉) · F. Monti
Sapienza Università di Roma, Rome, Italy
e-mail: mathew@diag.uniroma1.it; monti@diag.uniroma1.it

NL2ProcessOps [8], focuses on business process automation by extracting structured process operations from natural language process descriptions. Traditional business process modeling techniques require predefined workflows, which may lack flexibility in dynamic environments. *ProcessOps* employs an LLM-based approach to translate textual process descriptions into executable scripts, ensuring rapid process adaptation. Tool descriptions and context-aware retrieval mechanisms support the generation of Python-based process workflows that can be executed within business process management systems. The chapter presents such works focusing on the functioning of the developed LLM-based solutions; if the reader wants to get more details on the evaluation, please refer to the related full papers.

Together, these works showcase the potential of LLMs in facilitating intelligent data management and process automation. *COSMADS* emphasizes data retrieval and synthesis in industrial settings, while *ProcessOps* extends these capabilities to business process execution, demonstrating how LLMs can drive automation across different operational domains.

9.2 Data-on-Demand in the Industrial Domain with LLMs

In modern manufacturing environments, the ability to access and synthesize data on demand is increasingly vital for maintaining operational efficiency and agility. From shop floors to administrative offices, manufacturing companies generate vast amounts of heterogeneous data, including sensor readings, production metrics, and machine statuses. However, this data often resides in siloed systems, limiting its accessibility and utility. In this section, we introduce *COSMADS*—*CO*mposing *SM*Art *DA*ta *S*ervices [5]—a tool that synthesizes information extraction pipelines, in the form of Python scripts, starting from natural language queries and a documented *codebase* consisting of available data services and possibly other, previously defined, information extraction pipelines. Such pipelines, in particular, can be either manually defined or formerly generated with *COSMADS*.

9.2.1 The *COSMADS* Architecture

Figure 9.1 depicts the architectural components of *COSMADS*. A new execution is spawned as soon as a human operator specifies a natural language query ① to retrieve information from the ongoing manufacturing process. A query can be parametric, meaning that it can provide a set of input arguments (e.g., a time range or a specific kind of defect to be monitored). The reader can imagine the human operator to have little or zero knowledge about the available data services. As a consequence, it can be supposed that the query only expresses the required information without technical details on how to compute it.

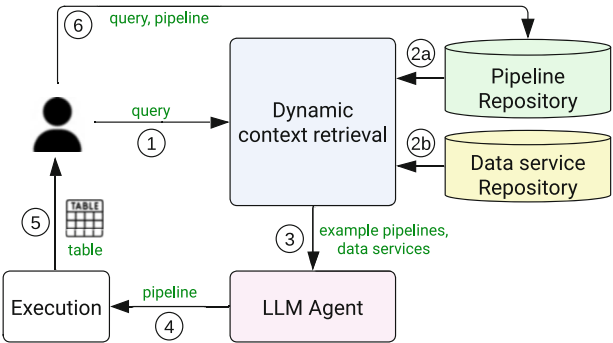


Fig. 9.1 COSMADS high-level architecture [5]

The core of the architecture is represented by an *LLM agent* that instructs a pre-trained LLM by feeding a query-specific prompt, which is built according to the output of the *dynamic context retrieval* component. This module analyzes the query and retrieves \mathcal{Q} , a set of example pipelines from a *pipeline repository*, and \mathcal{D} , a set of data services from a *data service repository* that can be used to answer the query.

Data services exposed by manufacturing assets in a factory can range from operational/actuating services (e.g., turning on the camera) to services that generate data (e.g., getting the current speed of the die cutting machine using its embedded chip). Data services can be accessed using different paradigms and communication protocols, but for simplicity, the authors assume they can be called through function calls wrapping the actual calling mechanism. A data service can expect a set of parameters and returns an output, which can be either structured or unstructured. For each data service, the *data service repository* contains a *documentation*, i.e., a textual description of the functioning and usage of a data service.

The *pipeline repository* consists of all the pipelines already available. These pipelines can be manually defined or obtained from previous executions of COSMADS. Each pipeline is associated with the query it fulfills. In general, pipelines can be defined using various modeling formalisms, such as programming languages and scientific workflow scripting languages [9], among others. In COSMADS, pipelines are software (Python) scripts that (i) produce a table as an output and (ii) make use of data services.

Examples of pipelines and documentation of relevant data services are fed ③ to the *LLM Agent* together with the original query. Such input data is incorporated into a prompt, according to a specific prompt template. The output pipeline generated by the *LLM agent* is then sent ④ to the execution module (e.g., the Python interpreter).

The execution of the pipeline finally ⑤ produces a table answering the query. If the human operator thinks the produced pipeline can be helpful as a future reference for future queries, the pipeline together with the originating query can be stored ⑥ in the pipeline repository.

9.2.2 Implementing COSMADS

A Python-based prototype of COSMADS¹ has been implemented using the LangChain framework. The base LLM model utilized for the LLM agent relies on GPT-4 (gpt-4-turbo) by OpenAI.²

9.2.2.1 Dynamic Context Retrieval

The contextual information required for the prompt of the LLM agent consists of (i) relevant previous queries with corresponding pipelines to be used as *few-shot examples* and (ii) the set of data services needed for solving the input query. As discussed in Sect. 9.2.1, this information can be obtained from the *pipeline repository* and the *data service repository*, respectively.

The *pipeline repository* is implemented as a vector store containing the vector representations of queries already solved. Each vector contains the embedding of the query and some metadata, including the path to the Python script containing the pipeline that solves that query. An example of a query that is embedded into a dense vector is provided below.

Example of stored query and pipeline metadata

```

1 {
2     "query": "Consider the next 5 carboard of the first diecutter.
3     ↳ Generate a table containing: (i) the number of cardboards
4     ↳ with no defects, (ii) those with errors, (iii) how many
5     ↳ fold errors, and (iv) how many hole errors.",
6     "metadata": {
7         "pipeline": "pipelines/04_07_2024-21_43_24.py"
8     }
9 }
```

The embedding is computed by applying the `text-embedding-ada-002` model by OpenAI. For the vector store, the authors rely on DocArray's DocIndex,³ which is well integrated into LangChain and allows us to efficiently access stored data. The implementation details of the pipeline repository are provided below, omitting or simplifying some methods for the sake of clarity.

¹ <https://github.com/jermathew/COSMADS>.

² The base LLM model for the LLM agent can be any LLM model available in the literature. Some changes in the prompt may be necessary if another model is selected.

³ Cf. <https://github.com/docarray/docarray>.

Pipeline repository implementation

```

1  from langchain_community.vectorstores import DocArrayInMemorySearch
2  from langchain_core.documents import Document
3  import json
4
5
6  class PipelineStore():
7
8  def __init__(self):
9      json_setup = json.load(open('queries_pipelines.json'))
10     queries = json_setup.keys()
11     self.docs = []
12     for q in queries:
13         self.docs.append(
14             Document(
15                 page_content = json_setup[q]['query'],
16                 metadata = {"pipeline": json_setup[q]['pipeline']}
17             )
18         )
19
20     def embed_docs(self, embedding_function):
21         self.embedding_function = embedding_function
22
23     self.db = DocArrayInMemorySearch.from_documents(documents=self.docs,
24         ↪ embedding=embedding_function)
25
26     def search(self, query):
27         best_result = None
28         best_match = self.db.similarity_search(query)
29         best_result = best_match[0]
30         return {'api_name': self.__class__.__name__, 'input': query, 'output':
31             ↪ best_result, 'exception': None}

```

The vector store is implemented as a Python class called `PipelineStore`. During instantiation (lines 8–18), the `PipelineStore` reads a JSON file containing the list of available pipelines and the query from which they stem. Each query-pipeline pair in the JSON file is formatted similarly to the example provided above. Each query is then wrapped into a `Document`, a built-in class from the `LangChain` library that provides a unified interface for managing and manipulating documents (e.g., generating an embedding).

The `embed_docs` method (line 20) contains the logic for embedding each `Document` parsed from the JSON file during initialization. The method takes an `embedding_function` parameter, i.e., a function that maps a string to a dense vector using an embedding model and uses it to embed each query and store it into `DocArray`, using the `DocArrayInMemorySearch` class. The latter represents a wrapper class for `DocArray` that allows easy storage of embeddings and performing queries.

The retrieval procedure is implemented in the `search` method (line 25). The core part is the `similarity_search` function, which takes a query as a parameter

and (i) transparently embeds it into a dense vector and (ii) returns a sorted list of similar queries in descending order of cosine similarity. Particularly, given a natural language query, the *dynamic context retrieval* component computes its embedding and retrieves the top- K similar queries from the vector store.

The PipelineStore can be initialized and the text-embedding-ada-002 model by OpenAI can be specified as the embedding function using the OpenAIEmbeddings class, as shown in the code snippet below:

Specify an embedding function for DocArray

```

1 from langchain_openai import OpenAIEmbeddings
2
3 self.pipeline_store = PipelineStore()
4 openai_key = None # replace with your OpenAI API key
5 embedding_function = OpenAIEmbeddings(model="text-embedding-ada-002",
6   ↪ api_key=openai_key)
7 self.pipeline_store.embed_docs(embedding_function)

```

To simplify interaction with the PipelineStore and allow for convenient configuration, the system introduces a wrapper class, PipelineManagerDB.

The PipelineManagerDB wrapper class

```

1 class PipelineManagerDB:
2
3     def __init__(self, openai_key):
4         self.pipeline_store = PipelineStore()
5         embedding_function = OpenAIEmbeddings(model="text-embedding-ada-002",
6           ↪ api_key=openai_key)
7         self.pipeline_store.embed_docs(embedding_function)
8
9     if __name__ == '__main__':
10         openai_key = None # replace with your OpenAI API key
11         tools_manager = PipelineManagerDB(openai_key=openai_key)

```

This wrapper simplifies the initialization of the vector store by enabling direct specification of the embedding function, which, in our case, is OpenAI's text-embedding-ada-002 model.

The set of data services needed for solving the input query is stored in the *data service repository*. A data service is used to retrieve historical or online data. Online data is often associated with the execution of some operation (e.g., taking a picture). As a consequence, any asset of the company exposing services to support its operational functionalities can be considered a data service. Referring to the example case study, an example of data service is the service related to the camera asset that captures frames (i.e., DS3). In COSMADS, each data service is realized as a Python class having two main components: (i) a function wrapping the existing service of the asset and representing the actual execution logic and (ii) a class

variable containing the documentation. The LLM agent relies on the documentation of the data services, which summarizes their capabilities, how they need to be used, a one-shot example, and a specification of the input and output parameters. The data service documentation of the camera asset capturing frames is provided below. Notably, this documentation corresponds to what in traditional service composition was referred to as service description.

Data service contextual information

```

1 "brief_description": "Data service that, given the id of a camera1,
  ↳ provides a frame captured from that camera1.",
2 "detailed_description":
3 """"Data service that, given the id of a camera1, provides a frame
  ↳ captured from that camera1.
4 In general instances of camera1 point downwards to a conveyor belt of a
  ↳ specific production line that transports single cutout cardboards
  ↳ produced by a specific die machine.
5 The data service takes a single parameter, namely the id of the camera1
  ↳ (an integer) and returns a frame captured from that camera1 as a
  ↳ numpy matrix.
6 The matrix is a 2D array having a shape of (1080, 1920, 3) where 1080
  ↳ is the height, 1920 is the width and 3 is the number of channels
  ↳ (RGB).
7 Example usage:
8 - If the id of the camera1 is 123, then the data service would be
  ↳ called as follows:
9 camera1_id = 123
10 frame = GetFrameFromCamera1.call(camera1_id=123)
11 # assuming the frame is a numpy matrix
12 print(frame.shape) # (1080, 1920, 3)
13 Things to keep in mind:
14 - The refresh rate of the camera is 1 second, i.e. the frame is updated
  ↳ every second, so if the data service is called multiple times
  ↳ within a second, it will return the same value.
15 - The frame is a numpy matrix, so avoid trying to access it as a
  ↳ dictionary.""",
16 "input_parameters": ["camera1_id:int"],
17 "output_values": ["frame:np.matrix"],
18 "module": "camera1"

```

Noteworthy, while the *dynamic context retrieval* selects only K example pipelines to be included in the prompt, all data services are considered. This approach is justified by the assumption that the number of services in the entire repository remains relatively stable, as the codebase of a company typically grows slowly and is also relatively small compared to the maximum prompt length allowed by the LLM. Conversely, the number of pipelines is expected to grow more significantly over time, as new pipelines are added to the repository, either through manual definition or automatic generation by COSMADS. Also, this excludes the possibility an incorrect pipeline is generated, simply because information about needed data services is not available.

9.2.2.2 LLM Agent

The LLM agent leverages the ICL ability of LLMs [2]. The quality of the output though is strongly dependent on the quality of the provided prompt [13]. For *COSMADS*, in particular, a *prompt template* is designed to be filled with the output of the *dynamic context retrieval* module, which follows the most common best practices [14]. These include (i) expressing the goal task clearly, (ii) including contextual information, (iii) providing demonstrations, and (iv) utilizing model-friendly format style. The prompt template is provided below:⁴

Prompt template

```
<LLM AGENT EXPERTISE>
Query: {query}
<GOAL DESCRIPTION>
{data_services}
<DATA SERVICES DOCUMENTATIONS STRUCTURE>
<GUIDELINES>
Here examples of pipelines that may help you in generating a new
↪ pipeline:
Query: {example_query}
Pipeline: {example_pipeline}
...other examples...
Answer:
```

The invariable parts of the prompt include (i) a system header describing the skills of the agent, (ii) a description of the specific goal to be fulfilled, (iii) the description of the structure of the documentation of the data services, and (iv) a set of guidelines the output of the agent must respect. It also contains the *dynamic context retrieval* output, i.e., the set of data services the pipeline can call, and the example queries. Finally, the input natural language query to be answered is also provided.

The implementation of the LLM agent is provided below.

Implementation of the LLM Agent

```
1 from langchain.prompts import ChatPromptTemplate
2 from langchain.schema import BaseOutputParser
3 from langchain_openai import ChatOpenAI
4
5 class PipelineGeneratorAgent:
6     """The agent that designs the pipeline."""
7
8     def __init__(self, openai_key):
```

⁴ The full prompt is available at https://github.com/jermathew/COSMADS/blob/main/src/pipeline_chain.py.

```

9      """Initialize the agent."""
10     # define the prompt
11     prompt_template = TEMPLATE
12     self.prompt = ChatPromptTemplate.from_template(prompt_template)
13     # define the LLM
14     self.llm = ChatOpenAI(model="gpt-4-turbo",
15                             api_key=openai_key,
16                             temperature=0.0)
17     # define the output parser
18     self.output_parser = CustomOutputParser()
19
20     def get_chain(self):
21         # generate the python function
22         agent_chain = self.prompt | self.llm | self.output_parser
23         return agent_chain
24
25
26     class CustomOutputParser(BaseOutputParser):
27         """The output parser for the LLM."""
28
29         def parse(self, text: str) -> str:
30             text = text.strip("\n")
31             text = text.strip()
32             # count how many ``` are in the text
33             back_count = text.count("```")
34             if back_count != 2:
35                 print(text)
36                 raise ValueError("The string should contain exactly two triple
37                                     ↳ backticks")
38             code = text.split("```")[1]
39             code = code.strip()[len("python"):].strip()
40             return code

```

The LLM agent is implemented as a class called `PipelineGeneratorAgent`, and its inner logic is defined in the `get_chain` method (line 20). This method implements the chain as a linear sequence of tasks, consisting of (i) passing the prompt to the model, (ii) generating the pipeline, and (iii) parsing its output. This sequence of steps is defined in line 22 using LCEL. The final part of the pipeline involves a custom output parser, which is used to parse the output generated by the LLM agent. The prompt, the LLM, and the output parser are defined in the constructor of the `PipelineGeneratorAgent` (lines 8–18). The prompt is set up in line 11 according to the previously described prompt structure and is then wrapped into a `ChatPromptTemplate`, a utility class to format the prompt for OpenAI's GPT models. The constructor also creates a `ChatOpenAI` object, which is a general wrapper for OpenAI models and is set up with `gpt-4-turbo`. Finally, a `CustomOutputParser` object (defined in lines 26–39) is initiated, which strips off character delimiters from the Python code generated by the LLM agent.

9.2.2.3 The COSMADS LCEL Chain

We have already seen an example of a chain in the `PipelineGeneratorAgent` class:

The chain used in `PipelineGeneratorAgent`

```

1 def get_chain(self):
2     # generate the python function
3     agent_chain = self.prompt | self.llm | self.output_parser
4     return agent_chain

```

This simple composition demonstrates how LangChain chains can be built using modular components. The LangChain Expression Language (LCEL) extends this idea by structuring more complex chains that involve multiple sequential and parallel operations. It introduces components such as `RunnableLambda` for single-step transformations and `RunnableParallel` for concurrent execution, enabling efficient and flexible task orchestration. LCEL also allows for the dynamic composition of LLM-driven workflows, making them well suited for integrating retrieval, generation, and execution steps, as seen in the *COSMADS* system.

We now present the high-level functioning of *COSMADS* through its LCEL chain. The main implementation is encapsulated in the `COSMADS` class, detailed below.

The `COSMADS` class

```

1 import sys
2 from pathlib import Path
3 import dotenv
4 import os
5 from langchain.schema.runnable import Runnable, RunnableLambda,
6     ↳ RunnableParallel, RunnablePassthrough
7
8 # append the path to the parent directory to the system path
9 import sys
10 sys.path.append(str(Path(__file__).parent.parent.parent))
11
12 from pipeline_manager_db import PipelineManagerDB
13 from pipeline_chain import PipelineGeneratorAgent
14 from runner_chain import PipelineRunner
15
16 INTERMEDIATE_RESULTS_FILEPATH = Path(__file__).parent /
17     ↳ "temp_pipeline.py"
18
19 class COSMADS:
20     def __init__(self):
21         dotenv.load_dotenv()
22         OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
23
24     self.pipeline_manager = PipelineManagerDB(OPENAI_API_KEY)

```

```

23 self.generator = PipelineGeneratorAgent(OPENAI_API_KEY)
24 self.runner = PipelineRunner()
25
26
27
28 def get_chain(self) -> Runnable:
29     generator_chain = self.generator.get_chain()
30     runner_chain = self.runner.get_chain()
31
32     generator_chain_output = {
33         "pipeline": generator_chain,
34         "inputs": RunnablePassthrough()
35     }
36
37     runner_chain_output = {
38         "output": runner_chain,
39         "inputs": RunnablePassthrough()
40     }
41
42     chain = ...
43
44     # return the chain
45     return chain
46
47
48 if __name__ == "__main__":
49     llm = COSMADS()
50     query = "" # your query here
51     result = llm.get_chain().invoke(query)
52     print(result["output"])

```

The *COSMADS* class includes a `get_chain` method that consists of a `Runnable` object implementing the *COSMADS* processing flow depicted in Fig. 9.1. The chain can be called using the `invoke` method (line 54) by passing an input query and will execute the *COSMADS* system, including the generation of the Python script and its execution.

The execution of the generated pipeline is managed by the `PipelineRunner` class, which is responsible for running the generated Python script and processing its results. We provide the implementation details of the `PipelineRunner` class below:

PipelineRunner implementation

```

1 class PipelineRunner:
2
3     def run_pipeline(self, pipeline_filepath: str) -> dict:
4         execution_ok = False
5
6         cwd = Path.cwd()
7         os.chdir(Path(pipeline_filepath).parent)

```

```

8  execution_result = os.system(f"python {pipeline_filepath}")
9
10 if execution_result == 0:
11     execution_ok = True
12
13 os.chdir(cwd)
14 return execution_ok
15
16 def parse_pipeline_result(self, pipeline_result_filepath: str) -> dict:
17     with open(pipeline_result_filepath, "r") as f:
18         result = json.load(f)
19
20     result = pd.DataFrame(result)
21     result = tabulate(result, headers='keys', tablefmt='psql')
22     return result
23
24 def get_chain(self) -> Runnable:
25     runner_chain = (
26         RunnableLambda(lambda x: {
27             "execution_ok": self.run_pipeline(x["pipeline_filepath"])
28         })
29         | RunnableBranch(
30             (lambda x: x["execution_ok"], RunnableLambda(lambda x:
31                 ↪ self.parse_pipeline_result(
32                 str(PIPELINE_RESULT_FILEPATH)
33             )))
34             (RunnableLambda(lambda x: "The pipeline did not run successfully"))
35         )
36     )
37     return runner_chain

```

The PipelineRunner class consists of three core methods. The `run_pipeline` method (line 3–14) executes the pipeline script in its designated directory and returns a Boolean indicating whether the execution was successful. If the execution completes without errors, the output of the pipeline is stored in a JSON file. The `parse_pipeline_result` method (line 16–22) loads the result into a Pandas DataFrame and formats it as a table for display. The PipelineRunner is integrated into the COSMADS LCEL chain through its `get_chain` method, which defines the execution workflow. The chain consists of a `RunnableLambda` that triggers the pipeline execution, by invoking the `run_pipeline` method, followed by a `RunnableBranch` that checks if the execution was successful. If the pipeline runs successfully, the output is parsed using the `parse_pipeline_result` method and returned; otherwise, an error message is displayed.

The implementation details of the whole COSMADS chain are provided below. Additionally, Fig. 9.2 provides its visual explanation.

LCEL implementation of COSMADS processing flow

```

1 chain = (
2   # Step 1: Retrieve similar pipelines from the Pipeline repository
3   RunnableLambda(lambda x: {
4       "query": x,
5       "pipeline_search":
6           ↪ self.pipeline_manager.pipeline_store.search(x),
7   })
8   # Step 2: Retrieve the the most relevant data services
9   # for this query from the Data Service repository
10  | RunnableLambda(
11  lambda x: {
12      "query": x["query"],
13      "example": self.get_example(x["pipeline_search"]["output"]),
14      "data_services": self.get_data_services()
15  })
16  )
17  # Step 3: Parse the retrieval results from both repositories
18  | RunnableLambda(
19  lambda x: {
20      "query": x["query"],
21      "data_services": x["data_services"][0],
22      "data_services_list": x["data_services"][1],
23      "example_query": x["example"][0],
24      "example_pipeline": x["example"][1],
25  })
26  )
27  # Step 4: Generate the pipeline using the LLM Agent
28  | generator_chain_output
29  # Step 5: Store the pipeline into a temporary Python script file
30  | RunnableParallel(
31  gen = RunnableLambda(lambda x: {
32      "query": x["inputs"]["query"],
33      "data_services": x["inputs"]["data_services"],
34      "example_query": x["inputs"]["example_query"],
35      "example_pipeline": x["inputs"]["example_pipeline"],
36      "pipeline": x["pipeline"]
37  }),
38  exe = RunnableLambda(lambda x:
39  self.save_intermediate_result_to_json(x["pipeline"],
40  ↪ x["inputs"]["data_services_list"])
41  )
42  | RunnableLambda(lambda x: {
43      "inputs": x,
44      "pipeline_filepath": str(INTERMEDIATE_RESULTS_FILEPATH)
45  })
46  # Step 6: Execute the pipeline (i.e. the temporary file)
47  | RunnableParallel(
48  inputs = RunnableLambda(lambda x: {

```

```

49     "query": x["inputs"]["gen"]["query"],
50     "data_services": x["inputs"]["gen"]["data_services"],
51     "example_query": x["inputs"]["gen"]["example_query"],
52     "example_pipeline": x["inputs"]["gen"]["example_pipeline"],
53     "pipeline": x["inputs"]["gen"]["pipeline"],
54 },
55 output = runner_chain_output
56 )
57 # Step 7: Parse the pipeline output
58 | RunnableLambda(lambda x: {
59     "query": x["inputs"]["query"],
60     "data_services": x["inputs"]["data_services"],
61     "example_query": x["inputs"]["example_query"],
62     "example_pipeline": x["inputs"]["example_pipeline"],
63     "pipeline": x["inputs"]["pipeline"],
64     "output": x["output"]["output"],
65 })
66 )

```

The COSMADS LCEL chain is structured as a sequence of `RunnableLambda` and `RunnableParallel` components, the two foundational abstractions in LangChain. `RunnableLambda` is designed for single-step transformations or computations. It wraps a callable function or lambda, providing a flexible way to define and execute individual tasks within a chain. For example, it can retrieve data, transform inputs, or parse outputs all while maintaining a clear and isolated scope for each operation. In the context of COSMADS, `RunnableLambda` is

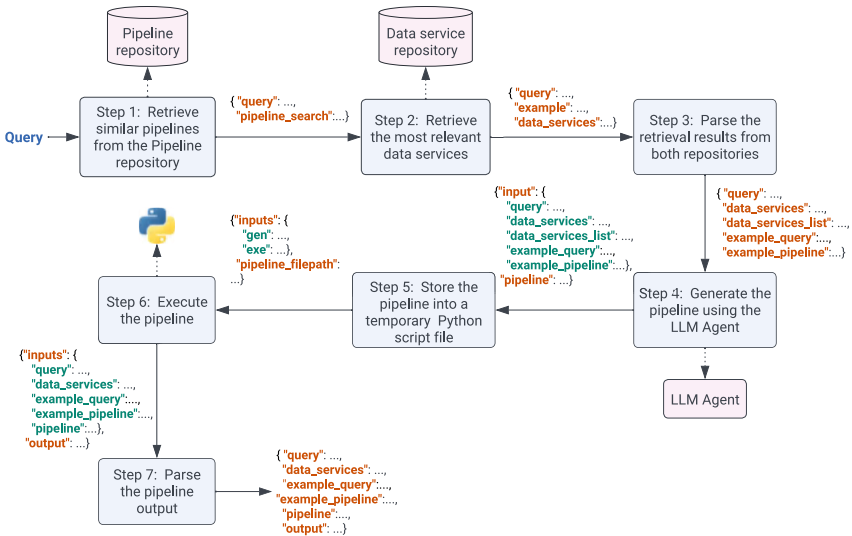


Fig. 9.2 COSMADS LCEL chain explained

used to perform tasks like querying repositories and parsing retrieval results. `RunnableParallel`, on the other hand, facilitates the concurrent execution of multiple independent tasks. It is particularly useful in scenarios where different processes can run simultaneously without interdependencies, enhancing the efficiency of the workflow.

The chain begins with a `RunnableLambda` (line 3) that queries the pipeline repository to retrieve pipelines relevant to the input query. This step leverages existing knowledge by identifying similar pipelines to use as examples in the generation process. Following this, another `RunnableLambda` (line 10) fetches data services from the data service repository that are most relevant to fulfilling the query. These two steps ensure that the *COSMADS* system gathers the necessary contextual information for creating a new pipeline.

Once the repository outputs are retrieved, a third `RunnableLambda` (line 18) parses and organizes the results, wrapping the query, example pipelines, and data services into a structured format. This parsed information is then fed into the LLM agent, which is implemented as part of a `RunnableLambda` (line 28). The agent uses the provided context to generate a Python script that represents the pipeline for fulfilling the query.

The generated pipeline is then stored using a `RunnableParallel` component (line 30), which handles both writing the pipeline to a temporary Python file and saving metadata about the inputs and selected data services. Subsequently, another `RunnableParallel` component (line 42) executes the stored pipeline while capturing its outputs concurrently.

Finally, a `RunnableLambda` (line 47) processes the output of the executed pipeline, structuring it into a relational table or other desired formats based on the query. The chain concludes with this step, delivering the final result to the user.

9.3 Extracting Process Operations with LLMs

The complexity of modern business processes often demands rapid and efficient solutions for translating textual descriptions into executable process operations. Traditional approaches to business process management rely heavily on predefined models, making them inflexible to evolving requirements. In this section, we describe `NL2ProcessOps` [8], an LLM-based approach for generating a process script from a textual process description. In practice, the script is a Python code containing invocations to external tools for executing the tasks of the given process. Each tool is characterized by a description that provides a textual representation of operations offered to execute the tasks. Tool documentations are prompted to the LLM to generate the script. To overcome the limitation of the input context length of LLMs, which cannot incorporate too much information, we consider the most appropriate tool descriptions and fit them within the prompt length.

9.3.1 The NL2ProcessOps Architecture and Pipeline

The goal of NL2ProcessOps is to provide a solution for process operations (ProcessOps), aiming to simplify the development and deployment of processes, similar to how DevOps simplifies the development and deployment of general software. Specifically, once a new process definition is available, the involved operations include (a) extracting a process model from the description, defining the legal execution traces similarly to a program, (b) binding of each task to a software module implementing that task, and (c) defining the data flow, i.e., how data artifacts are manipulated by the tasks.

The generation of the *process script* from a textual process description is broken down into multiple *stages* supported by LLMs and chained together as follows: (i) extraction of tasks and control flow from the textual process description (operation (a) of ProcessOps), (ii) retrieval of relevant tools corresponding to the extracted tasks (operation (b) of ProcessOps), and (iii) generation of the process script implementing the process (operation (c) of ProcessOps). Figure 9.3 illustrates the components of NL2ProcessOps and their interactions. The numbers in the circles represent the order of the performed operations.

Stage (i) consists of a textual process description given as input ① to the *Tasks-Model extractor*. This component is an LLM prompted to extract the tasks and the control flow of the process and generates the model representation as a Mermaid.js [3, 4]. The list of tasks paired with the textual process description (i.e., [proc_desc, tasks]) is input ②a to the *Tasks pre-processing* component. Concurrently, the process model and the textual process description (i.e., [proc_desc, model]) are input ②b to the *Code generator* component.

Stage (ii) (depicted in gray in Fig. 9.3) is inspired by the RAG concept to retrieve the relevant tools for the particular textual process description. The *Tasks pre-processing* component employs an LLM to refine the descriptions of the extracted tasks based on the textual process description. The refined list of tasks is then processed ③ by the *Tools retriever* component. This component interacts ④ with the vector database *Tools DB* and retrieves the most similar embedded tools offering the most suitable operation for each embedded task. *Tools DB* stores vectors consisting of the embeddings of the descriptions of the tools.

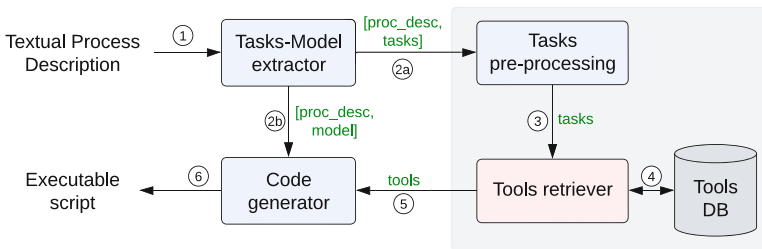


Fig. 9.3 Architecture of the NL2ProcessOps [8]

The list of retrieved tools is fed ⑤ into the *Code generator* component to initiate stage (iii), the *Code generator* LLM. The *Code generator*, given the textual process description, process model, and the list of tools (and their operations) implementing the process tasks, generates ⑥ a Python code—*process script*—embedding the control and data flows and implementing the process.

The integration of LLMs within the proposed approach is essential for several reasons. LLMs excel at processing natural language, making them ideal for extracting tasks and control flow from textual process descriptions. This first stage is critical as it forms the basis for the subsequent stages of the pipeline. Without accurate extraction of tasks and control flow, subsequent stages would lack the necessary information for generating meaningful outputs. The retrieval of relevant tools highly relies on the quality of the extracted tasks. An incorrect set of tools would affect the generation of the process script, leading to incorrect data and control flows. Finally, LLMs excel in generating high-quality code from a description. In this case, the textual process description guides the code generation, supported by the control flow and tools information derived from the previous stages.

9.3.2 A Running Example

Let us consider a real-world example in the Smart Manufacturing domain. Smart Manufacturing is a modern trend where cutting-edge technologies such as Industrial Internet of Things (IIoT) and artificial intelligence (AI) play pivotal roles in enabling quality enhancement, optimization, and automation of production processes [10]. In this domain, the integration of the proposed solution, paired with PEE and enterprise systems like Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP), enables the orchestration and execution of specific processes in a quick and efficient way [12].

Example The automatic calibration process of cardboard production consists of continuously capturing a photo of the cardboard being produced. Each photo is analyzed to check if all the markers identified are ok. If markers are not ok, the calibration process continues. If the markers are ok, the speed of the die-cutting machine is set to 10,000 RPM and the process ends.

The *Example* describes the automatic calibration process in cardboard production. Cardboard production is a manufacturing process that involves a die-cutting machine for the transformation of raw cardboard into printed cut-out cardboard sheets for the packaging industry. When starting a new order, calibration is needed to guarantee quality before proceeding with the production.

Figure 9.4 depicts the input-output of each of the NL2ProcessOps components over the *Example*. The numbers in circles are tightly connected to those in Fig. 9.3. Figure 9.4 reports the process model represented as Mermaid.js (green colored) and the list of (refined) tasks (blue and red colored) extracted from *Example*, the list of tools operations (purple colored) retrieved, and the process script—Python code (black colored). All the artifacts are available at the provided link.⁵

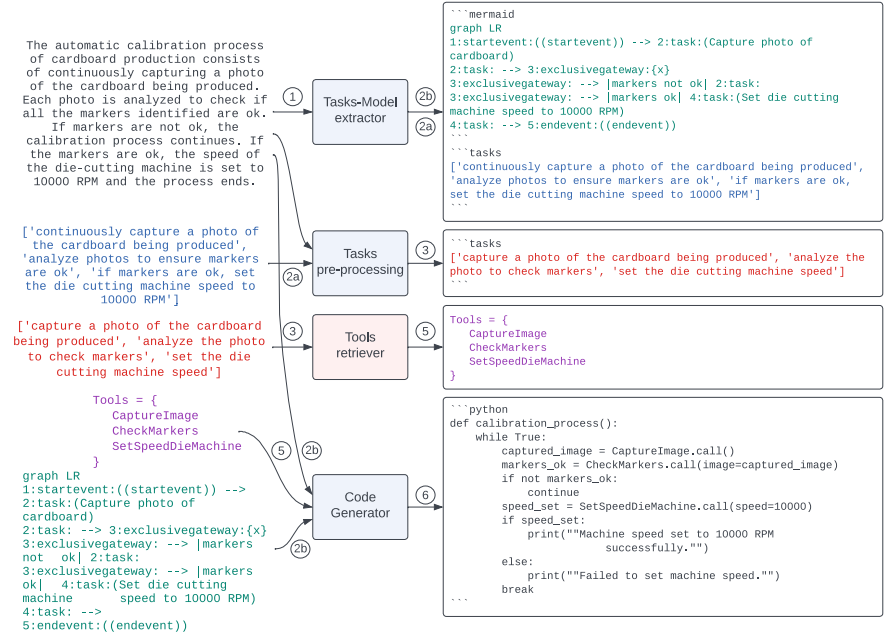


Fig. 9.4 Input-output of each component of NL2ProcessOps on *Example*

9.3.3 Realization

The authors have realized a prototype⁵ of NL2ProcessOps [7]. The prototype is developed in Python and built on top of LangChain,⁶ a framework for constructing LLM-powered applications facilitating the creation, management, and chaining of prompts. The base LLM utilized for the three LLM-based components is GPT-4 (gpt-4-0125-preview) from OpenAI [1] with temperature set to 0, implying a more deterministic LLM mode.

9.3.3.1 Tools Retriever

Tools are central to such approach as they support task execution in a service-oriented way, i.e., offering operations for specific tasks. As proposed in [6], the LLM relies solely on the documentation of these tools, which outlines their

⁵ Cf. <https://github.com/iaiamomo/NL2ProcessOps>.

⁶ Cf. <https://www.langchain.com/>.

capabilities, usage instructions, and outcomes. An excerpt from the documentation for the `SetSpeedDieMachine` tool of the *Example* is provided below. It consists of a general description of its unique operation, parameters details, and outcome description.

Example of tool documentation

```

1 SetSpeedDieMachine = {
2     "description": "Set the speed of the die-cutting machine.",
3     "more_details": "It takes as input the speed. It returns a
4     ↪ boolean value, True if the speed has been set, False
5     ↪ otherwise.",
6     "input_parameters": ["speed:int"],
7     "output_parameters": ["speed_set:bool"],
8     "tool": "die_machine" }

```

The *Tools DB* is a vector database containing the vector representations of the descriptions of the available tools. Each vector constitutes the embedding of the following information: `tool_name` and `description`, where the `description` is extracted from the tool documentation. An embedding is a sequence of numbers that represent information and enable better comprehension of relationships between similar information. The `text-embedding-ada-002` model from OpenAI⁷ is utilized to compute these vectors. The embeddings are usually managed in vector DBs, which enable a set of capabilities, including indexing, distance metrics, and similarity search. As vector DB, ChromaDB,⁸ which is open source and well integrated with LangChain, is employed. ChromaDB enables the implementation of RAG-based approach basing similarity search on cosine distance (where a lower score indicates better similarity).

In the *Example*, for the refined extracted task description “*set the speed of the die cutting machine*”, the similarity search identifies `SetSpeedDieMachine` tool operation as the most similar. Consequently, the *Die machine* tool is the most suitable for task execution.

The similarity search outputs a set of most similar tools with corresponding score results. The module guarantees that the useful tools are among those retrieved. In practice, given a task, the *Tool retriever* selects one or more tools based on their score values. The set of tools is then processed by the *Code generator* LLM that considers only those necessary for the specific case.

The tools retriever is implemented as a Python class responsible of managing tools, which are implemented as Python classes themselves. An example of tool is provided below:

⁷ Cf. <https://platform.openai.com/docs/models/embeddings>.

⁸ Cf. <https://www.trychroma.com/>.

Sample tool implementation

```

1 class ConfigureAssemblyLine:
2     description = {
3         "description": "Configure the assembly line.",
4         "more_details": "This tool takes no input and does not return
5         ↪ anything.",
6         "input_parameters": [],
7         "output_parameters": [],
8         "actor": "assembly_line"
9     }
10
11 def call():
12     return

```

A simplified version of the class implementing the *Tool Retriever* is provided below.

Embedding tools in NL2ProcessOps

```

1 from langchain_community.vectorstores.chroma import Chroma
2
3 class ToolStore():
4
5     def embed_tools(self, embedding_function):
6         self.embedding_function = embedding_function
7         self.db = Chroma.from_texts(self.tools, embedding_function)
8
9     def search(self, keywords):
10         best_match = self.db.similarity_search_with_score(keywords)
11         for i, match_elem in enumerate(best_match):
12             if i == 0 and match_elem[1] >= 0.5:
13                 break
14             elif i > 0 and match_elem[1] <= 0.3:
15                 tool_name = match_elem[0].page_content.split(' ')[0]
16                 file_name = match_elem[0].page_content.split(' ')[1]
17                 api_info = self.extract_input_output(tool_name, file_name)
18                 list_match.append(api_info)

```

The `embed_tools` method (line 5) makes usage of the `Chroma.from_texts` method (line 7), which takes a list of tool descriptions and converts them into embeddings, which are then indexed in ChromaDB for similarity-based searches.

The search method (line 9) performs retrieval by calculating the cosine similarity between the query embeddings and the stored tool embeddings. It uses ChromaDB's `similarity_search_with_score` function to fetch tools whose descriptions closely match the query. If the similarity score exceeds predefined thresholds, the corresponding tools are added to the result set.

In this realization, embeddings serve as a bridge between textual process descriptions and tool metadata. For example, a textual task description like “adjust machine speed to 10,000 RPM” is embedded and compared against stored tool descriptions.

Table 9.1 Prompt information of the LLM-based components

| | Tasks-model extractor | Tasks pre-processing | Code generator |
|-----|--|---|---|
| (a) | BPM expert | BPM expert | – |
| (b) | Extract the control flow in terms of a process model and the list of tasks | Rephrase the tasks descriptions | Generate a Python code |
| (c) | Description of BPMN elements | – | Tools descriptions and guidelines |
| (d) | Yes | Yes | No |
| (e) | Set of custom rules for the process model representation | – | Python program structure |
| (f) | Textual process description | Textual process description and extracted tasks | Textual process description and process model |

The similarity search may retrieve a match like the `SetSpeedDieMachine` tool, which provides the exact operation required.

9.3.3.2 Prompt Engineering

The prompt is a guide for the model, instructing it on relevant information and desired output formatting. The quality of the LLM output directly correlates with the quality of the provided prompt [13].

The proposed approach consists of three different LLM-based components, each specialized in a particular task, i.e., extraction of the process tasks and model, pre-processing of the extracted tasks, and generation of Python code. Each component is characterized by a specific prompt. Each prompt is characterized by all (or some) of the following parts: (a) the role the LLM plays that helps in controlling the output style [11], (b) a clear description of the task to be performed, (c) additional information (context) to aid the LLM in generating better responses, (d) few examples to teach the LLM, (e) type and format of the desired output, and (f) the input data used by the LLM to compute the response. Table 9.1 presents detailed information regarding each part of the prompt for the three LLM-based components. All the prompts are available at the provided link.⁵

9.3.3.3 NL2ProcessOps Implementation

The implementation of NL2ProcessOps leverages LangChain to integrate LLMs into a pipeline for generating process scripts from natural language descriptions. This section explains the key components of the implementation, focusing on how tasks, tools, and code generation are orchestrated.

The `ProcessLLM` class implements the core logic of `NL2ProcessOps`. It integrates multiple components, including task modeling, code generation, and tool management, by leveraging LangChain's `Runnable` utilities and custom LLM modules.

The `ProcessLLM` class

```

1 class ProcessLLM:
2     def __init__(self, model="gpt-3.5-turbo", openai_key=None,
3         ↪ temperature=0.0):
4         self.model_tasks_llm = MermaidLLM(model, openai_key,
5             ↪ temperature=temperature)
6         self.task_llm = TaskRetrieverLLM(model, openai_key,
7             ↪ temperature=temperature)
8         self.code_llm = CodeLLM(model, openai_key, temperature=temperature)
9
10    embedding_function = OpenAIEmbeddings(model="text-embedding-ada-002",
11        ↪ api_key=openai_key)
12    self.tools_store = ToolStore(openai_key)
13    self.tools_store.embed_tools(embedding_function)

```

The `ProcessLLM` class initializes various components, including (i) `MermaidLLM`, which extracts process models from descriptions; (ii) `TaskRetrieverLLM`, which identifies tasks within the process; and (iii) `CodeLLM`, which generates Python code based on tasks and tools. These classes are implemented as standard LangChain pipelines, adhering to the standard chain structure of `self.prompt | self.model | self.output_parser`. The `ProcessLLM` class also sets up ChromaDB (line 8) for semantic retrieval of the tools through the `ToolStore` class.

The `get_chain` method, which is provided below, defines the overall workflow of the `ProcessLLM` class. It orchestrates the sequence of operations using LangChain's `Runnable` utilities, including `RunnableLambda`, `RunnablePassthrough` (which directly propagates inputs without modification), and `RunnableBranch` (enabling conditional branching based on predicate functions).

The `ProcessLLM` chain

```

1 def get_chain(self):
2     model_tasks_llm_chain_output = self.model_tasks_llm_parser()
3     task_llm_chain_output = self.task_llm_parser()
4     code_llm_chain_output = self.code_llm_parser()
5
6     general_chain = (
7         RunnableLambda(lambda x: {
8             "model": x["model"],
9             "tools": self.tools_prompt_parser(x["tasks"]),
10            "input": x["input"],
11        })

```

```

12 | code_llm_chain_output
13 | RunnableBranch(
14 | (lambda x: not x["error_python"], RunnableLambda(lambda x:
15 |     ↪ self.parse_output(x))),
16 | (lambda x: "There are some errors in the python code.")
17 | )
18
19 chain = (
20 model_tasks_llm_chain_output
21 | task_llm_chain_output
22 | RunnableLambda(
23 |     lambda x: {
24 |         "tasks": x["tasks"],
25 |         "has_tasks": self.is_list_of_tasks(x["tasks"]),
26 |         "input": x["input"],
27 |         "model": x["model"],
28 |     }
29 | )
30 | RunnableBranch(
31 | (lambda x: x["has_tasks"], general_chain),
32 | (lambda x: "Your process description does not contain any task.")
33 | )
34 | )
35 | return chain

```

This method begins by defining three parsing subchains for task modeling (line 2), task retrieval (line 3), and code generation (line 4). The `model_tasks_llm_parser` chain processes the input to extract tasks, the `task_llm_parser` chain retrieves additional task details, and the `code_llm_parser` chain generates Python code based on the tasks and tools. These subchains are constructed using LangChain's `RunnableLambda` for inline computations and `RunnablePassthrough` to propagate inputs.

The heart of the workflow lies in the two branches created using `RunnableBranch`. The first branch (lines 30–33) checks whether tasks are present in the process description. If tasks are found, the `general_chain` is executed, which:

1. Integrates task descriptions, tools, and process inputs.
2. Generates Python code using the `code_llm_parser` (line 12).
3. Validates the generated code with the second branch (lines 13–16). It either parses the generated code for output or returns an error message if issues are detected.

If no tasks are found, the branch directly outputs a message stating, “Your process description does not contain any task.”

Finally, the `ProcessLM` provides a CLI-based interactive interface for users to input natural language process descriptions, generate Python code based on these descriptions, and optionally execute the generated code. Below is a simplified version of the main function:

Running NL2ProcessOps (simplified)

```

1  if __name__ == "__main__":
2      # Load environment variables and retrieve the OpenAI API key
3      dotenv.load_dotenv()
4      OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
5      model = "gpt-4-0125-preview" # Define the LLM model to use
6
7      # Instantiate the ProcessLLM pipeline
8      llm = ProcessLLM(model, OPENAI_API_KEY)
9
10     while True:
11         # Prompt the user to input a process description
12         input_text = input("Enter a process description (or press Enter to
13         ↪ skip): ")
14         if input_text == "":
15             continue
16
17         # Process the input and generate Python code
18         result = llm.get_chain().invoke({"input": input_text})
19         print(f"Generated Code:\n{result}")
20
21         # Ask if the user wants to execute the process or provide another
22         ↪ description
23         user_choice = input("Execute the process? (y)\nEnter a new description?
24         ↪ (p)\nQuit? (q): ")
25         if user_choice == "y":
26             try:
27                 # Validate and execute the generated Python code
28                 print("Executing the process...")
29                 if os.system("python -m py_compile llm_process_code.py") != 0:
30                     print("The generated code contains syntax errors.")
31                 else:
32                     os.system("python llm_process_code.py")
33                     print("Process executed successfully.")
34                 except Exception as e:
35                     print(f"Error during execution: {e}")
36                 elif user_choice == "p":
37                     continue # Loop back for a new process description
38                 else:
39                     print("Exiting...")
40                     break

```

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)

2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *NeurIPS 2020* (2020)
3. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modelling: State of the art, applications, and implications in practice. In: *BPM 2023 Forum* (2023)
4. Klievtsova, N., Benzin, J.V., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modeling: Can generative ai empower domain experts in creating and redesigning process models? *arXiv preprint arXiv:2304.11065* (2024)
5. Mathew, J.G., Monti, F., Firmani, D., Leotta, F., Mandreoli, F., Mecella, M.: Composing smart data services in shop floors through large language models. In: *International Conference on Service-Oriented Computing*, pp. 287–296. Springer (2024)
6. Mialon, G., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al.: Augmented language models: a survey. *arXiv preprint arXiv:2302.07842* (2023)
7. Monti, F., Leotta, F., Mangler, J., Mecella, M., Rinderle-Ma, S.: *NL2ProcessOps* (2024). <https://doi.org/10.5281/zenodo.11219809>
8. Monti, F., Leotta, F., Mangler, J., Mecella, M., Rinderle-Ma, S.: *NI2processops*: Towards llm-guided code generation for process execution. In: *BPM*. Springer (2024)
9. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A., Li, P.: Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics* **20**(17), 3045–3054 (2004)
10. Popkova, E.G., Ragulina, Y.V., Bogoviz, A.V.: *Industry 4.0: Industrial Revolution of the 21st Century*, vol. 169. Springer (2019)
11. Shanahan, M., McDonnell, K., Reynolds, L.: Role play with large language models. *Nature* **623**, 493–498 (2023)
12. Thalmann, S., Mangler, J., Schreck, T., Huemer, C., Streit, M., Pauker, F., Weichhart, G., Schulte, S., Kittl, C., Pollak, C., et al.: Data analytics for industrial process improvement a vision paper. In: *CBI 2018*. IEEE (2018)
13. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382* (2023)
14. Zhao, W.X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al.: A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023)

Chapter 10

Ethics and Governance of Large Language Models



Gabriele Rendace Presta  and Andrea Paoletta 

Abstract Artificial Intelligence is reshaping the ontological structure of reality. The analysis examines AI's role in the re-ontologization process, emphasizing the implications of digital tools on human existence and showing how ethical concerns also emerge from this ontological shift. To face these AI ethical issues, this chapter proposes a Heideggerian approach that looks at ethics as care of the human stay (*Aufenthal*). It is an attempt to define a critical effort able to re-evaluate the human role in AI development and use, accompanying the already proposed ethical principles and the law-making processes (especially in Europe). The analysis focuses then on the global AI political and regulative scenario, with an overview observing China, Russia, EU, and the USA. The text shows the common points and differences between these approaches, offering a structured understanding of main governmental actions on AI (from development strategies to direct regulative actions). Finally, the overview on regulations will be crucial to reflect on censorship, AI openness, freedom, and rights.

10.1 Preliminary Ontology

10.1.1 Introduction

In this first introductory section, the reader will focus on the ontological issues related to digital tools and AI. According to a Heideggerian vision, ontology is here intended as the analysis of the deep composition of the meaning to be of reality [53], looking what means for something to be and not just why something is. In fact, digital tools, and artificial intelligence more than anything else, are reshaping the ontology not just of single objects but of reality as a whole [42, 50]. Every aspect of the world—actions, relations, basic shapes, fundamental elements, etc.—is changed by these new technologies. This condition, of course, puts under strict

G. R. Presta (✉) · A. Paoletta
Sapienza Università di Roma, Rome, Italy

attention the way AI is used and which are the principle effects it has: here ethical concerns emerge, along with all the possible political, juridical, and philosophical answers [40, 42]. This passage will be essential to enlighten how the ethical and ontological problems are deeply related and why this previous section on ontology will be necessary to understand the subsequent sections on ethics and rules.

Two examples about this process of re-ontologization are then shown into the following paragraphs of the introduction. Both deal with the phenomenon of prosuming (progressive merging of the two figures of producer and consumer [18]), trying to show how processes and relations are transformed by the use of digital tools. The examples will be the chance to open a reflection on the role of human beings in an informational society, where the world humans live in is adapted for the coexistence with artificial digital agents.

However, at the end of this section, a new idea enters the discourse: the possibility to design critically optimized political and juridical tools to achieve the right solutions for the progressive emergence of ethical concerns related to the AI re-ontologization of reality.

10.1.2 The Ontological Problem Behind the Ethical Approach to AI

To achieve a real comprehension of what artificial intelligence is and how it works from a philosophical perspective, we must understand the inner composition of the world it has been built to be used in. This is not an approach that seeks a philosophical answer to the deep structures of artificial intelligence algorithms and models—it would not be different to ask a misleading question about what a machine is¹—but a more complex analysis of the way AI is used and the way the world is arranged to allow its efficient use. A first point that must be clear and has to be accurately considered is that the ontology we are discussing is not the one of AI machines but the one of contemporary world, which is rearranged in its deeper composition and structure by these machines.

However, this kind of research brings us exactly to the point where we must understand why and how we use AI to the question “what should we do?”, a question that, in a more radical and essential approach, turns into the following deeply complex interrogation: “How should we think?”² Nevertheless, this way

¹ As Turing already noticed, it would be a question leading to no solution, putting our research on the wrong path [42].

² It's essentially a Heideggerian philosophical interrogation that enshrines an ethical approach into a firstly ontological one: “Dies alles vermogen wir nur, wenn wir vor der anscheinend immer nachsten und allein als dringlich erscheinenden Frage: was sollen wir tun, dies bedenken: Wie müssen wir denken? Denn das Denken ist das heigentliche Handeln, wenn Handeln heißt, dem Wesen des Seins an die Hand gehen” *eng.*: “We can only achieve all this if, in the face of the seemingly ever-present and only urgent question: “What should we do?”, we also consider: “How

of thinking should constitute an always-necessary approach when construing a possible political or ethical path in our contemporary society. Specifically, new technologies, as artificial intelligence is, represent the main tools involved in the progressive re-ontologization of the world [42]. AI is reshaping the way things get done, and this condition of course leads to an interrogation about how societies and organizations should use these technologies: here the ethical issue emerges, along with all the possible political answers.

To give the reader a more practical vision of the just suggested re-ontologization process, everyday examples on how digital technologies and tools rebuild our reality and restructure the world could be useful. Among these cultural and ontological evolutions, a quite famous one is prosuming, often discussed also in new media theory.³ It consists in the progressive merging of two different figures operating in the cultural and economic fields: the consumer and the producer.⁴ In the last decades, a growing number of young adolescents from every country of the world started to turn the stories they read on books they had bought into new collective narratives shared on blogs and Web sites, thus transforming themselves from simple readers, or passive consumers, to readers-writers, at the same time passive readers and active producers involved in the expansion of a fictional world, becoming new digital entities that we would call prosumers [18]. We can make now an example that would resonate a little bit more in our mind, contemplating how prosuming operates also in the sphere of new generative AI. The owner of a creative agency who needs to create textual contents for Web sites could hire a copywriter, give her or him some indications, and obtain a result, good and specific contents that will add to the Web sites. In this case, there's a consumer asking for a product. Instead, if the creative agency owner does not ask to a copywriter, an expert in the field of textual content creation, but start prompting indications to a LLM obtaining as a result the contents for the Web sites, the owner is turning somehow into a prosumer. The owner is self-producing (producer) the contents that will be used for the final aim (consumer).

These are two very different examples regarding the power of re-shaping and re-ontologize reality digital technologies have. They change actions, identities, and relations, establishing a new ontological perimeter. It must be underlined that it is not just a matter of why and how we choose to do something, but it deals also with the way we think and understand reality.⁵

must we think?" [52]. As we will see further in the analysis of ethical approaches to AI, the question about the way we should think becomes a general critical approach to technological developments that precedes and envelopes law-making and legislative directions.

³ Important contribution has been given by researchers in the fields of new media phenomena. Fanfictioning has probably been one of the most explored [18].

⁴ A good report on the historical evolution of this term and its meaning can be found in Floridi, 2022 [42].

⁵ To understand better, the following passage can be useful: "il digitale non è semplicemente qualcosa che potenzia o aumenta una realtà, ma qualcosa che la trasforma radicalmente, perché crea nuovi ambienti e nuove forme di agire con cui interagiamo. [...] ho utilizzato l'espressione

It is important to recognize that ethical concerns and the political and juridical regulations that will follow them cannot be detached from a philosophical and cultural detection directed to explore the mechanisms of contemporary thinking. This is clearly a purpose linked with a human-oriented approach that will focus on the essential articulation of how we think and then of how we construe our reality. It is important to put at the center of the analysis this peculiar point: firstly, to recognize its importance in the field of AI development and use and, as second instance, to insert it in this discourse as a filter of analysis to observe which path we have decided to walk through.

Nowadays, it is necessary to evaluate if there is a real human-centered approach to AI. The progressive transformation of society has led to a digital technology environment where we are just one part, but not the main constitutive element. Human beings therefore appear to live a condition of simple means in the new reality constituted by the encounter of online and offline worlds.⁶

Our artificial agents, currently constitutive components in every aspect of our life and in every space we walk through, exist because we're building an environment where these objects⁷ can live and prosper. We build an envelope where digital entities, including AI of course, can work autonomously and realize a product or give any kind of service [42]. This works on a small scale and in a more expanded and complex one. Robots involved in manufacturing cars or moving boxes in distribution centers, for example, can complete their tasks only in the perimeter of the factory, a small environment completely arranged to guarantee a successful accomplishment of digital entities tasks. Smart cities are examples on a large scale of this envelopment process.⁸ So, there is a cultural and material movement of reality toward a digitized status or, better said, toward a merged condition where

re-ontologizzazione per fare riferimento a una radicale forma di re-ingegnerizzazione, che non consiste soltanto nel disegnare, costruire o strutturare un sistema [...] in modo nuovo, ma nel trasformare fondamentalmente la sua natura intrinseca, vale a dire la sua ontologia. [...] Attraverso la re-ontologizzazione della modernità, per dirlo in breve, il digitale sta anche ridefinendo dal punto di vista epistemologico la mentalità moderna, cioè molte delle nostre concezioni e idee consolidate." *eng.*: "The digital is not simply something that enhances or increases a reality, but something that radically transforms it, because it creates new environments and new forms of action with which we interact. [...] I have used the term re-ontologization to refer to a radical form of re-engineering, which does not consist merely in designing, building or structuring a system [...] in a new way, but in fundamentally transforming its intrinsic nature, that is, its ontology. [...] Through the re-ontologization of modernity, in short, the digital is also redefining the modern mentality from an epistemological point of view, that is, many of our established conceptions and ideas" [42].

⁶ A world we can easily call with the widespread expression *infosphere* [40].

⁷ The word "object" for digital entities can be not exactly the most fitted one. South Korean philosopher Byung-Chul Han talks about *Undinge*, non-objects, because of digital incapability to produce real negativity and establish the perimeter of reality [50]. This has consequences on how we make choices, build relations, work, and so on. Of course, it can represent also a good point to start a discussion where digital ethics and esthetics are profoundly interlinked.

⁸ We'll see that one of the first countries involved in the development of AI policies and ethical discussion, China, is also the most advanced in the creation of smart cities and the subsequent data collection [87].

digital informational reality results combined with the material and physically experienced world: the *onlife* of the *infosphere* [40]. Well, this is the starting point to understand the re-ontologization of reality. In these new spaces, things are as they've never been before, and humans are not the measure of this process but simple guests into an environmental project progressively more oriented to and fit for machines and digital entities [42].

However, humans still have the possibility to imagine and draw a world shaped by artificial intelligence, and our ability to design will probably be the aspect that will determine the quality of the answers we will give to political and ethical concerns deriving from AI development [42]. Future challenges will lie not only in the way we drive digital and AI innovation but also and overall in the way we will govern it through a critical designing approach.

Key Points of Preliminary Ontology for Artificial Intelligence

- Understanding AI requires analyzing the world it is built to function in.
- The focus is not just on AI models but on how AI reshapes reality.
- Ethical concerns lead to fundamental questions:
 - “*What should we do?*” → “*How should we think?*” (Heideggerian radical approach).
- AI and digital technologies restructure reality at its core.
 - Example: *Prosuming* (producer and consumer roles merge):
 - New Media Theory on fanfictions.
 - AI tools like LLMs turning content consumers into a hybrid of content producers-consumers (prosumers).
- AI changes actions, identities, and relationships.
- The digital world does not merely enhance reality but fundamentally transforms the real world merging with it.
- *Re-ontologization* is the name of this radical process of reconstruction (Floridi).
- Concept of the *Infosphere*: merging of online and offline worlds.
- A central issue regards the question on centrality of human beings and their transformation into mere means inside a merged digital-material environment.
- AI operates in enveloped environments designed for its efficiency (factories, smart cities), and the enveloping process is progressively involving every aspect of reality.
- AI is shaping the world, but humans can still direct this transformation.
- Ethical and political decisions will determine the future of AI governance.
- *Key challenge*: Not just driving AI innovation but designing it with a critical approach.

10.2 Ethics

10.2.1 Introduction

After the introductory first section centered on the relation between ontology and ethics, this second part will take a further look on how an ethical approach to AI could be imagined and arranged.

The starting point will be an attempt to describe how ethics could be interpreted, looking at the topic employing a semantic analysis of the Ancient Greek word *ἦθος* (*ēthos*), from which the modern term ethics comes from.

In fact, the ancient Greek word *ἦθος* could embrace a double meaning: a first one related to the idea of “habit” and another one linked to the concept of “stay.” Starting from this ambiguous semantic nature, it is possible to intercept two philosophical interpretations that can operate as useful tools to develop an ethical approach to AI:

1. The first approach derives from the Aristotelian tradition and links *ἦθος* to the role that the laws of the state play in shaping the virtues of citizens [6];
2. The second interpretation has instead its roots in Heraclitus and Heidegger’s way of thinking, connecting ethics to the condition of care humans should direct toward the place they live every day, aside from any constriction deriving from the external action of law and government [51].

The attempt is to show how these two approaches can co-operate, defining a more complex scenario where laws are accompanied by the personal development of an articulated critical thinking.

Then, the analysis will focus on the *Unified Framework of Five Principles for AI* proposed by Floridi and Cowls. It is the first fundamental systematic and organized definition of an ethical path for the development and use of artificial intelligence digital tools [43].

The *Unified Framework* is composed of four principles inherited in AI ethical studies from the research already developed in the field of bioethics. They are beneficence, non-maleficence, autonomy, and justice. To them the two researchers add another one: explicability. This last principle underlines how the processes and data exploited by these technologies must be clear and open for the users, who must be aware of the entity that is accountable for the potential dangers and threats deriving from the use of AI technology [43].

Finally, the analysis will introduce three fields—education, climate change, and work—where AI has produced interesting questions for ethical research. The action of AI in relation to these three areas represents a great interest for international political organizations, especially for the European Union, and at the same time, these three fields are good examples of how AI can generate ambiguous and controversial effects in everyday aspects of society [7, 25, 47, 72, 85]. This last reflection thus proposes to the reader a first practical idea of what an AI ethical concern can be and leads to the next part of this chapter, where different countries’ strategies and legislative actions to face these issues are introduced.

10.2.2 *Where the Meaning of Ethics Comes From*

Starting from a point of view able to merge philosophy, linguistics, and semantics, the first passage of this second section will be related to a wider and articulated understanding of the term “ethic.” It derives from the ancient Greek word ἦθος (ēthos),⁹ and the meaning it had for Greeks can open our reflection to a vast comprehension of what ethics is.

Aristotle in the *Nicomachean Ethics* introduces the concept of virtue and distinguishes between two different kinds of virtue: the dianoetic virtue and the ethical virtue. The first one can be reached *through learning* (ἐκ διδασκαλίας), while the second one is derived *from habit* (ἐξ ἔθους).¹⁰

According to Aristotle, ethical behavior is not related just to a natural disposition, but it is obtained by also nurturing our inner nature through the continuous smoothing action of the habits. He affirms that someone can become virtuous and pursue an ethical path by only accomplishing good actions: we can learn to be good people through the repetition of these good actions, confirming our natural disposition with practice [6]. In Aristotle’s discourse then enters one of the main political and cultural protagonists of ancient Greek world, the πόλις (polis), the city-state. In this passage, Aristotle affirms that the political leaders of the city-state, the legislators, guide the citizens toward virtuous behaviors creating good habits they can follow to become just.¹¹ Now we can achieve a deeper understanding of Aristotelian thinking: ethical behavior—the practical and moral one, related to action and to the question “what should we do?”—is shaped by habits, and the laws of the State are arranged to create good habits that will lead the citizens to the achievement of ethical virtue.

To summarize with a sentence: here ἦθος enters the semantic area of the habit, dialogues with this concept, and, according to Aristotle, interlaces consequently itself with the legislative bodies.

However, we must look at the wider semantic horizon the word ἦθος opens in the ancient Greek language. It is particularly interesting to understand what happens when this word assumes other possible meaning. In fact, ἦθος does not relate just to the semantic area of the habit, but it contemplates also the idea of the dwelling, the

⁹ The etymology of the English term can be easily found on the https://www.oed.com/dictionary/ethics_n?tl=true, Oxford English Dictionary.

¹⁰ “Virtue being, as we have seen, of two kinds, intellectual and moral, intellectual virtue is for the most part both produced and increased by instruction, and therefore requires experience and time; whereas moral or ethical virtue is the product of ‘habit’” [6].

¹¹ “This truth [of learning virtues through habit] is attested by the experience of states: lawgivers make the citizens good by training them in habits of right action—this is the aim of all legislation, and if it fails to do this it is a failure; this is what distinguishes a good form of constitution from a bad one” [6].

one of the home.¹² Something ethical, for ancient Greeks, could be just an object or a place you are accustomed to. Heidegger stimulates a profound philosophical reflection starting exactly from an ancient Greek sentence attributed to Heraclitus [51]. The sentence is extrapolated from Heraclitus Fragment 119: ἦθος ἀνθρώπων δαίμων (ēthos anthrōpō daimōn).¹³

Heidegger denotes how translators, biased by their modern view of ethics and ethical thinking, often render inaccurately this passage: *Seine Eigenart ist dem Menschen sein Dämon* (in English, “His own character is for the human his demon”) [51]. According to Heidegger, this is a partial translation of ἦθος as “character,” unable to capture the real essential meaning of the word ἦθος; he proposes instead to translate the word ἦθος with the German term *Aufenthalt*, which we could turn into English as “stay” or “residence” [51]. Well, we can see how here the radical meaning of ἦθος as the accustomed place, as home, comes back and somehow takes place again in the realm of contemporary ethics.

The German philosopher then tells a short story about Heraclitus.¹⁴ The ancient Greek savior would have been seen close to an oven, warming his cold hands and body, by some visitors who expected to admire the famous Heraclitus thinking in a special or extraordinary pose. However, the philosopher understands their disappointment and invites them to come closer to the oven, saying that *Götterwesen auch hieran* (in English, “Gods are here too”) [51]. He was trying to communicate to his visitors that also in the usual and most common stay, human beings can find the essence of things and the essence of themselves too. That’s why the Heideggerian translation of the Heraclitus sentence “ἦθος ἀνθρώπων δαίμων” will be “*Der (geheure) Aufenthalt ist dem Menschen das Offene für die Anweisung des Gottes (des Un-geheuren)*” (in English, “The [usual] stay is for the human the opening for the coming of the gods [of the un-usual]”) [51].

So, if we consider the word ἦθος according to its fundamental meaning, ethics and ethical thinking must be linked directly to the idea of the stay of the human being. Then, the main task of ethics should be to nurture and take care the essence of the stay. This leads back ethics directly to an ontological dimension.¹⁵

¹² A simple dictionary can be helpful in understanding properly the multiplicity of different semantic uses of the word ἦθος, (e.g., <https://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0057:entry=h=qos>, Liddell, Scott, 1940).

¹³ This fragment Heidegger starts his reflection from has been reported by ancient author Stobaeus in his *Anthology* [74].

¹⁴ The story Heidegger tells has been reported by Aristotle in his *De Partibus Animalium* [5].

¹⁵ “Soll nun gemäß der Grundbedeutung des ἦθος der Name Ethik dies sagen, daß sie den Aufenthalt des Menschen bedenkt, dann ist dasjenige Denken, das die Wahrheit des Seins als das anfängliche Element des Menschen als eines eksistierenden denkt, in sich schon die ursprüngliche Ethik. Dieses Denken ist aber dann auch nicht erst Ethik, weil es Ontologie ist (in English, “If, according to the fundamental meaning of ἦθος, the name ‘ethics’ is to signify a consideration of human existence, then that thinking which conceives the truth of being as the primordial element of human existence, as that which exists, is in itself already the original ethics. Such thinking, however, is not first ethics and then ontology”) [51].

In a political and social environment influenced by AI, where challenging ethical concerns frequently enter the cultural debate, it is important to learn to design and think about ethical solutions that overcome a simple regulatory base—the Aristotelian $\eta\theta o\varsigma$ of laws and bureaucratic protocols cannot be sufficient—a model that has the bravery to embrace not just a radical redesign of tech tools but also a redefinition of the human environment and society, welfare, education, economic relations, and care, they will be used in, thus reactivating the Heraclitus-Heideggerian $\eta\theta o\varsigma$, which looks at the essence of the stay of human beings that pones itself before the simple action.¹⁶ It is true what Floridi has already theorized about the creation and expansion of the *Infosphere* [40], but this reontologization [39, 42] theory works overall on a superficial plan. An informational interpretation of the world does not contemplate the entire and complex sphere of human action, an aspect that we must consider, as said before, when we talk about ethics. It is necessary to propose again the Heideggerian return of ethics to the realm of ontology, trying to understand how we can construe a world that can host properly human action, and not just a world reontologized by informational machines on which we try to impose laws and constrictions. A radical solution could be necessary when a game-changing technology is introduced in human society. Informational tools are redefining the human relation with the world [40]; they are however *Undinge* (non-things) [50], and in a reality where objects are not able to create negativity and mediate, there is a high ethical risk: the impossibility of action [4].

This comparison between the two meaning of the word $\eta\theta o\varsigma$ could represent an important premise to enter the discourse of AI ethics trying to develop a critical approach, but it has to meet also the philosophical prodromes that influenced contemporary debate on the ethical use of digital tools and AI.

¹⁶ “Es muß nämlich gefragt werden: wenn das Denken, die Wahrheit des Seins bedenkend, das Wesen der Humanitas als Eksistenz aus deren Zugehörigkeit zum Sein bestimmt, bleibt dann dieses Denken nur ein theoretisches Vorstellen vom Sein und vom Menschen, oder lassen sich aus solcher Erkenntnis zugleich Anweisungen für das tätige Leben entnehmen und diesem an die Hand geben? Die Antwort lautet: dieses Denken ist weder theoretisch noch praktisch. Es ereignet sich vor dieser Unterscheidung” (*In English*, “One must ask: if thinking, in considering the truth of being, determines the essence of humanity as existence from its belonging to being, does this thinking then remain merely a theoretical conception of being and of man, or can directives for practical life be derived from such knowledge and given to it? The answer is: this thinking is neither theoretical nor practical. It occurs prior to this distinction”) [51].

10.2.3 *The Unified Framework of Five Principles for AI in Society*

It is crucial, for whoever wants to enhance his or her knowledge in the field of digital ethics, to know what the *Unified Framework of Five Principles for AI in Society* is.¹⁷ It represents the attempt, led by Floridi and Cows, to manage the chaotic scenario of ethical initiatives launched by several organizations to adopt AI technologies that could benefit society.¹⁸

The two researchers then identified also the emergence of five principles that could be adopted to evaluate and to create new artificial intelligence tools [43].

The first four principles were already widely spread in the field of bioethics: beneficence, non-maleficence, autonomy, and justice.

The *beneficence* principle regards the creation of AI tools that are directed to benefit the existence of human beings, and it is one of the most widespread principles among the documents realized in the field of AI ethics and analyzed by Floridi and Cows. Linked to the idea of beneficence are the concepts of well-being, common good, human dignity, and sustainability [42, 43].

The *non-maleficence* principle could seem not so different from the previous one of beneficence, or at least logically consequent, but there is instead an important semantic nuance we must look at. Non-maleficence principle reminds us that overuse and misuse of AI tools must be avoided in order to preserve human beings from important and irreversible issues caused by these technologies. Several concerns emerge when artificial intelligence deals with our personal data or when it enters the military perimeter and helps the development of new technologically advanced weapons.¹⁹ One interesting point is also that from the documents, it is not so clear whether the people who use them or the technologies themselves should be subjected to this principle of not harming [42, 43].

Strictly related to this philosophical question is the third principle, the one of *autonomy*. In fact, when we use AI tools, we give part of our decision-making power to machines, which operate autonomously from our direct will. So, it is necessary to find the right balance between the possibility of human beings to act on the processes and outputs machines give and the power to operate autonomously we completely delegate to these technologies. If we do not succeed to reach this

¹⁷ Of course, research on AI and debates on AI ethics do not begin in the contemporary era with this paper published at the end of the last decade but at the beginning of the second half of the twentieth century [69, 88, 101].

¹⁸ Floridi and Cows analyze six documents, elaborated by organizations, authorities, and researchers, that represented various ethical approaches to AI. These initiatives were (1) *The Asilomar AI Principles*, (2) *The Montreal Declaration for Responsible AI*, (3) *Ethically Aligned Design: A Vision for Prioritizing Human Well-being with Autonomous and Intelligent Systems*, (4) *The Statement on Artificial Intelligence, Robotics and 'Autonomous' Systems*, (5) *AI in the UK: ready, willing and able?*, and (6) *The Tenets of the Partnership on AI* [43].

¹⁹ Further in this chapter, it will be clear how war and AI are profoundly interlinked (e.g., in the case of Russia).

balance, the potential threat could be the imposition of machine autonomy over human autonomy. AI technologies should instead improve overall human autonomy. Human beings, not machines, should decide if they want to leave part of their will and power to other autonomous agents. It is necessary also to adopt a sort of decide-to-delegate model, where human beings decide which tasks in a decision-making process they want to delegate to AI—e.g., they could delegate to machines in order to obtain more efficacy. Nonetheless, human agents should always have the chance to take again power over the parts of the process they delegated, in a sort of “deciding to decide again” action [42, 43].

The fourth principle is the one of *justice*. It deals with the idea of AI as profitable and positive for each member of human society. Artificial intelligence should not be discriminatory and promote diversity in order to remove any potential threats for justice itself and for citizens. The first action should be to work on the datasets we use for the training of our AI, removing biased data that can give the machine an inheritance of discriminatory or prejudicial information,²⁰ and as a consequence, it leads to potentially dangerous outputs. Justice principle promotes also the idea of a shared prosperity and shared benefits coming from AI²¹ [42, 43].

Floridi and Cowles denote another important element about the world of artificial intelligence: a small bunch of people, experts, are really conscious of how these technologies work, while a great mass of other human beings do not really know the fundamentals of AI, but it is involved in the profound changes that are spread by these new tools and systems [43]. They underline a condition all the documents they analyzed contributed to make clearer: there is something unintelligible that’s invisibly working behind these technologies; it is accessible only by few people, and this situation is creating a really unavoidable concern [42, 43]. That’s why they added to the other four principles another one: *explicability*. AI technologies must be intelligible, they must answer to questions linked to the basic structure of their functioning, and at the same time they must be accountable so we can know who has the responsibility of possible dangerous or threatening AI outputs.²²

The *Unified Framework of Five Principles for AI in Society* has been crucial for the ethical and legal debate in Europe [44] and for further studies all over the

²⁰ Dependent from the quality of data, from their robustness, is the whole infrastructure of AI systems, including the potential biased outputs [94].

²¹ Artificial intelligence can lead to the progressive diffusion of real threats for historically disadvantaged groups and reinforce existing power hierarchies spreading economic inequality [60].

²² “The addition of the principle of ‘explicability,’ incorporating both the epistemological sense of ‘intelligibility’ (as an answer to the question ‘how does it work?’) and in the ethical sense of ‘accountability’ (as an answer to the question ‘who is responsible for the way it works?’), is the crucial missing piece of the AI ethics jigsaw. It complements the other four principles: for AI to be beneficent and non-maleficent, we must be able to understand the good or harm it is actually doing to society, and in which ways; for AI to promote and not constrain human autonomy, our ‘decision about who should decide’ must be informed by knowledge of how AI would act instead of us; and for AI to be just, we must know whom to hold accountable in the event of a serious, negative outcome, which would require in turn adequate understanding of why this outcome arose”[43].

world.²³ So, it is a fundamental tool for whoever wants to understand the ethical consequences of developing AI tools, but, as said before, we must consider also the wider challenge—to look for critical instruments to take care of the human stay, and not just to respect rigid protocols and laws²⁴—ethical thinking involves.²⁵

10.2.4 *Education, Climate Change, Work*

In this section, as already suggested before, the attempt will be directed to give the reader the chance to face the major issues ethical research on AI has faced in specific fields, trying to describe a first practical idea of how AI is redefining the way things get done. A process that produces at the same time the concerns and challenges contemporary societies must deal with.

The three fields that will receive a deeper analysis are education, climate change, and work. They have been chosen because they are considered by international organizations, such as the United Nations and the European Union, crucial and game-changing elements for an equal and sustainable development of worldwide societies [7, 25, 47, 72, 85].

Moreover, they are also specific fields that can help the reader understand clearly how AI controversies emerge:

1. Artificial intelligence makes education more targeted and personalized, helping students learn, but at the same time it can be a potential threat for their privacy [10, 11].
2. AI model use has a negative impact on the environment, but the data analysis and the outputs obtained through these models can help in the fight against climate change [40, 42].
3. These new technologies promise to boost economy and enhance the quality of life, but nowadays workers are still concerned for the possibility of losing their jobs [20].

These are just few examples that will be then understood more deeply and analyzed in their wide complexity.

As just suggested, artificial intelligence, especially generative AI, has profoundly contributed to give a great boost to transform the field of education [19]. This process of redefinition is already in progress all over the world and will define a great change in the way people study and learn. AI is contributing to several strategic roles of contemporary education: instructional and teaching assistants,

²³ For example, the interesting attempts to evaluate governmental regulation of AI starting from the principles extrapolated by Floridi and Cowls [89].

²⁴ We talked about this before when introducing the two different meaning of $\eta\theta\omicron\varsigma$.

²⁵ The *Framework* can of course help us during the phases of development and deployment, but important concerns are still present when we talk about the whole life cycle of these AI tools [99].

helping with bureaucratic duties and the production of materials for lessons and evaluation, parent assistants, and administrator assistants [11]. Artificial intelligence can combine data and sophisticated statistical tools to monitor school climate and act when necessary to improve the well-being of students [11] or to give teachers the chance to understand better the different levels of their classrooms and create a more personalized educational path for each student [10].

However, along with these stunning new possibilities, artificial intelligence arises also numerous issues regarding the ethical consequences its use determines. Firstly, AI can contribute to spread cheating habits (e.g., doing homework for students). This could lead to a diminished interest for a real learning of what they are studying and to a lack of skills when they will end their path in schools and universities [11]. Another important issue regards the potential bias present in datasets AI systems are trained by. Students could be exposed to potentially racist or discriminatory ideas, which could be a real obstacle for the development of ethical and social abilities fundamental for civil coexistence in our modern societies. At the same time, these AI biases can lead the machine to favor students belonging to a certain socioeconomic class or with some specific characteristics [11].

Other problems are linked to potential diffusion of private data. Students can expose themselves chatting with AI bots, letting their privacy become the source for further analysis and extraction of value. At the same time, teachers should be careful when they input into AI systems potential personal data and information regarding the school environment, including colleagues and students [1, 10, 11].

Finally, some challenges can be directly related to the area of social connections and equity issues. Relevant in education is, of course, the chance to meet the other and establish a profound connection that can lead to the activation of cognitive and learning skills. This happens when an educator meets her or his students, but rarely when a young pupil alone chats with conversational bots. Social connections are central in the discourse of education, and we must answer to the possible lack of this link not only between the student and teacher but also of course between students [11, 49]. Meanwhile, inequality in access to AI systems and models represents a potential threat to the development of the necessary skills to face the hard challenges of life. This imbalanced access is primarily influenced by economic and social gaps between students and can contribute to widen these socioeconomic lags.²⁶

The second field that we will now be giving deeper attention to is climate change. According to Floridi [42], in relation to climate change, artificial intelligence has a sort of double face. On one hand, these models are giving a fundamental help in contrasting the sudden environmental changes occurring all around the world. AI, in fact, is directly contributing to the design of the fundamental tools employed in the attempt to answer to these new challenges—from neural networks able to forecast climatic trends [56] or the water demand under climate change [92] to systems that provide more sustainable solutions in the optimization of energy [33].

²⁶ We must consider that AI systems, if ethically directed and able to consider different intersections, can instead be a resource to fight inequality and delete privileges [61].

On the other hand, artificial intelligence models, in terms of emissions, impact directly on climate change when they are developed, during training, and when they are implemented by final users.²⁷ There is no doubt that one important concern is computing power. Deep Learning, a high-energy-requiring type of machine learning, needs a lot of computing power to accomplish its tasks, and of course, this has a relevant impact also on the emissions of these models [42]. To estimate the ecological footprint of AI, we must consider different elements: the hardware used, how long a session of training lasts, the number of neural networks trained, when during the day the model is trained, how much memory it uses, and the kind of resources the electric network providing energy employs²⁸ [54].

These models are trained by researchers that often do not give sufficient information on the training conditions of their models and on the attempts done before they obtain publishable results. A lot of computing power is exploited to reach just a small progress in models' accuracy, forgetting to invest part of the research in enhancing energy efficiency. This approach creates a huge entry barrier for developing countries that do not have enough resources to start the AI development process [42].

The debate about the impact of AI on climate change is still open and complex,²⁹ it would deserve a deeper analysis, including not just an ethical approach but also a political and social one. However, Floridi's thirteen recommendations for an ethical approach to AI and climate change issues can be a useful starting point. They can be useful for a practical approach to AI research and governance; at the same time, they can lead to the development of a first critical approach to the relationship between AI and climate change challenges.³⁰

²⁷ As <https://openai.com/index/ai-and-compute/>, Amodei and Fernandez, 2018 report, the required energy to train models has been increasing constantly and in an incredibly rapid way since 2012.

²⁸ To train the GPT-3 model, it has been estimated that every single training session would have to produce 223.920 kg of $C O_2$. An average car in the USA emits every year almost 4600 kg of $C O_2$ [42].

²⁹ An important discussion deals also with the idea of human hyperagency and the boost AI could give to the dangerous development of Anthropocene [22].

³⁰ (1) "Incentivise a world-leading initiative (Observatory) to document evidence of AI being used to combat climate change around the world [...]; (2) develop standards of quality, accuracy, privacy, relevance and interoperability for data to be included in the forthcoming Common European Green Deal data space; identify aspects of climate action for which more data would be most beneficial; and explore, in consultation with domain experts and civil society organisations, how this data could be pooled in a common global climate data space; (3) incentivise collaborations between data providers and technical experts in the private sector with domain experts from civil society, in the form of 'challenges' [...]; (4) incentivise the development of sustainable, scalable responses to climate change that incorporate AI technology [...]; (5) develop mechanisms for ethical auditing of AI systems [...]; (6) develop greener, smarter and cheaper data infrastructure (e.g., European research data centres) for researchers and universities across the EU; (7) assess AI and its underlying infrastructure (e.g., data centres) when formulating energy management and carbon mitigation strategies [...]; (8) develop carbon assessment and disclosure standards for AI to help the field align on metrics, increase research transparency, and communicate carbon footprints effectively via methods such as adding carbon labels to AI-based technologies and models listed

The third and last area of interest for our introductory dissertation on ethical approaches to AI is work. Our personal and social identity, our self-esteem, and the way we approach the social world is often influenced by our job³¹ [42]. Artificial intelligence has yet to begin and will be a game-changer in the field of work, restructuring the way most of our jobs are done³² but also undermining the necessity of many job skills [20]. It has been calculated that 40% of the workforce worldwide faces significant changes due to the rise of AI, and it's interesting to see how advanced economies are more exposed to this process (60%) because its workforce is prevalently composed by professionals whose tasks are increasingly being influenced or reshaped by AI's growing capabilities [20]. So, more educated workers seem to be more exposed, and often they are women. This adds another critical point: AI-led job transformation will bring a prevalent exposure of women, potentially widening the already-existing gender gap. At the same time, highly educated workers have clearly shown to be able to adapt to these changes and shift to jobs with high AI complementarity [20]. Nonetheless, there are still some critical points we must observe carefully; among them, of course, is the risk of labor displacement, with a consequent reduction³³ of labor income for workers.

in online libraries, journals, and leaderboards; (9) incentivise diverse research agendas by funding and rewarding projects that diverge from the current trend of compute-intensive AI research to explore energy-efficient AI; (10) incentivise energy-efficient and green research by making EU funding conditional on applicants measuring and reporting their estimated energy consumption and GHG emissions. [...]; (11) develop conference and journal checklists that include the disclosure of, *inter alia*, energy consumption, computational complexity, and experiments (e.g., number of training runs, and models produced) to align the field on common metrics [...]; (12) assess the carbon footprint of AI models that appear on popular libraries and platforms, such as PyTorch, TensorFlow and Hugging Face, to inform users about their environmental costs; (13) incentivise the development of efficiency metrics for AI research and development (including model training) by promoting efficiency improvements and objectives in journals, conferences and challenges" [32].

³¹ Often work is also linked to a condition of alienation and loss of identity and well-being. This Marxist point of view [68] has been a theoretical base for the critical contemporary philosophic approach to work. Franco Berardi, in 1970, published in Italy his famous work *Contro il lavoro*, which, using Marxist theoretical tools, opens a wider reflection on the roles of machines not seen anymore as producers of value but as simple mediators of the worker "lavoro vivo" (human work), the only kind of work that gives real value to the final product. Berardi says that capitalists will always need workers and "lavoro vivo" to create this value: according to Berardi, to destroy capitalism, it should be necessary to leave production only to machines [14], a point of view that leads to some critical points but able at the same time to start a new philosophic discourse in the era of AI and spread automation.

³² As said before, we must remember that AI is, at its core, leading a reality re-ontologization process, and work is not less affected by these technological transforming pressure. The ontology of actions, their essence, would be moving with the input of AI toward a smart condition. Actions are somehow becoming smart actions. So, according to this vision, artificial intelligence would not be a new kind of intelligence but a new kind of action, a smart action [42].

³³ However, according to Cazzaniga [20], this reduction could be offset by gains in productivity. This position is not always shared by other researchers who instead criticize it and denounce a polarization that "widen income and wealth disparities" [37].

Then, another problem is represented by the emergence of strong differences linked to the way advanced economies can deal with the AI transformation of work and how instead the so-called low-income countries do not seem to be enough prepared to face this new challenge. The situation described here, of course, can bring as consequence the growth of the gaps between the former economies and the latter ones, leading to a condition of diffused inequality [20].

Policymakers have to answer to this AI-led transformation of work that will disrupt the actual scenario and potentially create social unrest. The political action must consider to lead an equitable and ethical integration of AI, granting the adequate training of the next generation and a general reskilling of workers already part of the existing economies and threatened by AI disruption. As seen before, AI can contribute to spreading inequalities on the worldwide scenario, widening gaps and lags between countries. An answer is absolutely required and must consider the adoption of policies of international cooperation [20].

One practical solution is represented by universal basic income (UBI). It is considered a radical idea, but its popularity and experimental use are progressively spreading all around the world [42]. UBI is really considered a good option from different thinkers—Martin Luther King and Milton Freedman among them—and this popularity exists because universal basic income has a wide range of interpretation. Liberals look at UBI as a possible additional income related to a still existing wage, so alone it would not be enough to grant someone a comfortable life. Another version of UBI is related to the removal of the domain of unconditionality in welfare services, such as education, health, and housing. So, it would be possible to receive UBI but no more free welfare. This is considered a right-wing vision of UBI. The third version of universal basic income, the left-wing one, proposes to part away livelihood from work: everyone would receive an income adequate for a basic but comfortable life, and then he or she would be free to decide if and how he or she wants to contribute to society.³⁴ This last type of UBI results in the most radical but at the same time the one that can produce the most creative and interesting political and ethical alternatives [48].

This introduction to AI ethics tried to give a general overview on what an ethical approach could be and which are some of the major implications that have to be considered when we design this kind of technological tools.

It is now time to delve deeper into the legislative initiatives in the field of artificial intelligence undertaken by China, Russia, the European Union, and the USA.

³⁴ This would be effective “economic freedom” [48], and AI general transformations of work and society could lead to this new economic scenario. In this regard, as <https://www.forbes.com/sites/bernardmarr/2024/12/12/will-ai-make-universal-basic-income-inevitable/>, Marr, 2024 shows, the debate is still open.

Key Points of Ethics for Artificial Intelligence

- *Ethics and Ontology*: Ethics in AI is deeply connected to ontological considerations.
- We consider two meanings of the word ἦθος (*ēthos*):
 - Habit: Ethics as shaped by repeated actions and societal norms (Aristotle)
 - Dwelling: Ethics as the way humans inhabit and create a relationship of care with their world (Heraclitus, Heidegger)
- *Aristotle's perspective*:
 - Virtue ethics: Ethical behavior is cultivated by “habits.” The laws of the city-state promote habitual good actions leading citizens to a virtuous state.
 - The πόλις (polis) and laws shape the virtues of the citizens.
- *Heraclitus-Heideggerian perspective*:
 - Ethics as continuous care for the human stay in everyday life
 - ἦθος ἀνθρώπων δαίμων (ēthos anthrōpō daimōn) → Ethics as a critical individual awareness of the profound relationship of care humans should create with the place and social environment they live in
- AI and ethics, aware of the important role laws have, must go however beyond regulation:
 - Ethical AI must understand the governmental actions but requires more than just laws and constraints: it is necessary to have a critical approach that allows to comprehend the essential necessities of a re-ontologized reality.
- *Unified Framework of Five Principles for AI* (Floridi, Cows):
 - Beneficence, Non-maleficence, Autonomy, Justice (inherited by AI ethics from bioethical reflection)
 - Added principle for AI ethics: Explicability—transparency and accountability in AI decision-making
- *Practical Applications*: AI ethics in key domains.
 - Education: AI's role in learning environments
 - AI assists in teaching, administration, and monitoring student well-being.
 - AI can promote personalized learning but also facilitate cheating.
 - Risk of bias in training data can affect students unfairly.

(continued)

- Privacy concerns: AI may collect and exploit student data.
- AI impacts social interaction, potentially reducing meaningful student-teacher engagement.
- Digital divide: Unequal access to AI tools may widen educational disparities.
- Climate change: AI for sustainability and environmental concerns.
 - AI helps predict climate patterns and optimize energy use.
 - AI development and implementation have high energy costs and emissions.
 - Computing power demands can create barriers for low-income countries and threaten their efforts to deal with climate change issues.
 - An important guideline: Floridi's 13 recommendations for ethical AI in climate policy.
- Work: automation, labor ethics, and the transformation of employment.
 - AI transforms job roles and redefines economic structures.
 - Risk of job displacement, especially in advanced economies.
 - AI-led job transformation disproportionately affects women.
 - Income disparities may widen due to unequal adaptation to AI.
 - Need for policies and awareness to ensure fair AI integration in the workforce.
 - Universal Basic Income (UBI): A potential solution with various ideological perspectives.
- Ethical AI requires a critical approach to the law-shaped Aristotelian ethical approach that contemplates an Heraclitus-Heideggerian view able to involve a deep comprehension of the effects actions of AI developers and users have on reality:
 - Developing an ethical approach that accompanies law-driven and human-centered actions
 - Recognizing AI's role in the re-ontologization of reality

10.3 Rules

10.3.1 Introduction

In this third section, the analysis will focus on the introduction of the main approaches four important political powers have developed worldwide to deal with

the major concerns deriving from AI. Their approaches, as the research will try to explain, are strictly related to their economical and political strategies for AI.

The analysis that is going to be introduced to the reader must consider as a fundamental premise what has been shown in the previous observations related to ethics and its meanings. This political approach has a crucial role in the way societies tried to govern AI transformations, but it is not sufficient, and a specific critical self-awareness that pones real care in everyday actions developers decide to pursue must be always considered, regardless of the political direction laws will suggest.

Governmental interest for digital technologies, use of data, and AI model development has tried to move toward an ethical and human-oriented path and has been, especially in the European Union [82, 85], a precious shield against the forms of surveillance capitalistic structure assumed to turn personal collected data into profits [103]. Data were and are still used not just to monitor individual behaviors in order to predict choices but also to persuade, manipulate, and finally generate behaviors. This new extractive paradigm simply turns humans into means of production to obtain raw materials (data) that through algorithmic processes are then turned into profits [103]. This structure produces an enveloped environment where humans can rapidly shift from the condition of aims in the perimeter of a human-oriented world into means to exploit in order to generate profits. This can be considered another form of the digital re-ontologization process that has been underlined by Floridi [42]. A transformation that, of course, alarmed European institutions that already in 2016 intervened to protect its citizens data with the release of the GDPR (General Data Protection Regulation) [82].

However, governments had also an important role in the spread of potential threats coming from AI and digital technologies in general. The use of these tools for social control in China and Russia will be one of the main topics of the following pages [91, 98], but also what happened in the USA and EU must be observed with concerned attention.

In this section, the main elements determining the nature of EU and US approach to AI will be analyzed, and in the conclusive part of the research, there will be an introductory attempt to comprehend how emergency became a political and ideological instrument to impose aggressive private data extraction and to develop related algorithmic infrastructures able to produce surveillance methods that undermine human freedom and rights [103].

Again, a critical ethical approach that respects but knows to look beyond the simple perimeter suggested by governmental directions must be considered a good proposal to develop further human-oriented AI solutions.

The following sections will analyze governmental approaches to AI and data regulation, focusing on the ethical challenges posed by digital technologies in the EU, the USA, China, and Russia. Special attention will be given to surveillance, social control, legislative frameworks like the GDPR and AIA, and the geopolitical competition in AI development.

10.3.2 Chinese Approach to AI Development and Regulation: AIDP and the Mission of Reaching AI Global Leadership

China is today a world leader in AI technologies development and has set a specific goal to achieve a complete supremacy in this field around the year 2030 [87]. The main political and institutional tool organized by China's Communist Party leaders to accomplish this aim is the so-called AIDP or NGAIDP, the New Generation Artificial Intelligence Development Plan. It is a complex document directed to the definition of an articulated strategy exploring different layers of economics and society: from Chinese possibilities in the development of national AI technologies able to compete in the international scenario with other powers' AI resources to the chance to make China an important hub that will attract researchers³⁵ from all over the world [87].

The AIDP was the result of a long debate, and attention inside Chinese governmental institutions started already at the beginning of the last decade [87] and is now living a period where the focus is moving to a possible more ethical use of these technologies [91]. The AIDP is led by the AI Strategy Advisory Committee and the Ministry of Science and technology, both institutions related to the central government, but the implementation of the numerous initiatives promoted by the AIDP is a task left to local administrations and the private sector. In the 3-year plan linked to the development plan here discussed, this sort of dialogue between micro- and macro-governing scale is openly declared.³⁶ However, the central government chose AI national champions—among them Baidu and Alibaba—giving these companies specific tasks in the wide field of AI development, trying simultaneously to promote the birth and growth of new Chinese AI start-up companies, whose success³⁷ is not less relevant for the realization of the plan [87].

China puts AI technologies at the center of its political, military, and economic policies, considering these new tools as fundamental to obtain results in international competition, economic development, and social governance. AI is looked at as perfectly coherent to the Chinese attempt to construe a national apparatus defined by parallel skills in the international competition with the USA. According to China's government, it is not possible, nor convenient, to engage a direct race against the USA in fields where the latter has a clearly superior power; so, the aim is to start to overperform in other areas of development, and the new AI tools and

³⁵ In 2023, China was “the largest producer of AI research in the world” [91].

³⁶ To focus on this specific point, important indications are given by Khanal [59].

³⁷ “Technology start-ups within China often receive government support and subsidies for developing AI technologies. As an example, Zhongguancun Innovation Town is a purpose-built, government subsidised, incubator workspace that provides a suite of services to help Chinese technology start-ups succeed, often in the sectors where national champions have been selected” [87].

infrastructure are an unexplored path that could represent a good opportunity³⁸ for China to compete [87].

China has set also important goals for its economy, targeting AI as an important instrument to continue boosting its growth with innovation after previous decades of incredible economic development. On one hand, AI promises to create an incredible expected growth in the country by 2030,³⁹ resulting in an economic development of the 26% [81]), and the investments in this field has already had a rush in 2018, with a stunning growth rate of 48% [64]. On the other hand, although Chinese industry reached an incredible rate of automation in factories and services, the managing of the progressive dismantling of workers opportunities and the spreading of wealth inequalities could be important menaces to the stability of Chinese political and social apparatus [87].

CCP sees in AI also a great potentiality to obtain a more effective social control over the population in the country. Main issues from this point of view are the possible use in social scoring and uncontrolled collection of personal data. Social governance, for example, is implemented by a social credit system based on algorithms and whose complete acceptance by the Chinese people is debated.⁴⁰ These mathematical structures that are at the base of every digital software have also been under continuous attention by the Chinese government that looks at them as potential threats for the absolute control of information that CCP wants to detain.⁴¹ This is a real concern widespread among the leaders of the party, who decided to answer with a regulation that points at algorithms as the main digital element toward which one important attention must be led to obtain a real control of digital tools. Algorithms in China are part of an AI supply chain that was elected as the point of entry for regulation, with a registry. In this registry are included also the training

³⁸ For example, a “leapfrog” in military development: something that can make China a leader of international competition for the arrangement of new digital-equipped armies, especially in cyber warfare [87].

³⁹ The deadline set in the AIDP to reach AI supremacy according to the Chinese government [87].

⁴⁰ “The disruption of the Maoist period followed by an ‘opening up’ has created a moral vacuum within China. These concerns are echoed by the Chinese public, with Ipsos Mori finding that concerns over ‘moral decline’ in China were twice as high as the global. This is something that has been recognised by the Chinese government, with high-level officials, including President Xi, forwarding the idea of a ‘minimum moral standard’ within society [...] In view of the government, AI can be used to this end [...] Articles in the Western media often emphasise the problematic nature of ‘the’ Social Credit System, due to the authoritarian undertones of this pervasive control. Examples of public dissatisfaction with specific features of locally run social credit systems appear to support this viewpoint [102]. In some cases, there have even been cases of public backlash leading to revisions in the rating criteria for local social credit systems. In contrast, some commentators have emphasized that, domestically, a national social credit system may be positively received as a response to the perception of moral decline in China” [87].

⁴¹ “Tracing the origin of the term ‘algorithmic recommendation’ backward in Chinese state media shows that it first emerged during a 2017 CCP backlash against ByteDance’s news and media apps, in which user feeds were dictated by algorithms. The party viewed this as threatening its ability to set the agenda of public discourse and began looking for ways to rein in algorithms used for information dissemination” [91].

data used for the development of the model: they must be diverse and objective [91]. Algorithms and data are also subjected—specially deep synthesis outputs—to a vague form of censorship that includes the necessity to adhere to specific social and moral values, to avoid misleading for the public, and to disturb the economic and social sphere [91]. The creation of this first tool gave bureaucrats the chance to get in touch with this new technology, building a first know-how that has been useful in the advancement of further legislative documents. Then, China's approach became vertical—dealing with a single manifestation of a technology—and iterative so when a regulation gets obsolete, it is just updated with a new release that solves the problems and obsolescence of the previous one [91].

10.3.3 Russian Approach to AI Development and Regulation: Strategies to Fill Lags, Face Ethical Issues, and Compete in the International Scenario

In Russia, current AI strategies and opportunities are affected by the recent events following the invasion of Ukraine, with a progressive mutation of Russian economy, policies, and academic research in this technological field [13]. However, to understand better the actual situation in the country, as already done before for China, we make a little step back to the last decade.

Russia has tried to set a specific strategy to compete in the worldwide scenario since 2018,⁴² when the government of the Federation released the first of four main policies and programs [93], the *Digital Economy National Program*. It was arranged to pursue three main objectives:

1. To expand domestic investments and development on digital economy
2. To create a stable and secure information and telecommunications infrastructure for high-speed transmission, processing, and storage of large amounts of data
3. To use Russian software in local and national government bodies and organization⁴³

This first program was then followed by the *AI Roadmap*, promoted by the government and led by the private Russian company Sber, which had the duty to detail different AI subtechnologies—from natural language processing to neural interfaces, neurostimulation, and neurosensing—to fund and develop [93].

⁴² We must consider that Russia has what we can call an ethics-based approach to the regulation of AI. Governmental directions on this topic have been tailored by private companies to build an environment where real legal restrictions are more or less absent. So, ethical concerns are often just an excuse for ethics washing and not effective laws to look at when developing new products [75].

⁴³ <https://oecd.ai/en/dashboards/policy-initiatives/http:%2F%2Faiipo.oecd.org%2F2021-data-policyInitiatives-24621>, OECD 2023.

Russia then established the *National Strategy on AI Development*, appointing in 2019 a specific commission with the aim to supervise and update the strategy itself. Russia wanted and wants to use this tool to fill the gap with other more advanced nations, setting 2030 as the deadline to obtain positive responses for its efforts. The strategy looks at the quality of the data employed in the development of new national AI technologies, but at the same time great relevance is given also to the nurture of new talents in the AI research field and to the attempt of creating the best environment for the progressive spreading of AI culture and education in Russian society. The *National Strategy on AI Development* was then emended in February 2024, and great importance was given again to research and development, with the precise strategic will of reducing Russian dependence from other international competitors in AI software and hardware creation [93].

Also relevant is one of the four main Russian policies on AI, the 2020 *AI Federal Project*, whose attempt is to underline how the AI initiatives will be funded, which are the metrics and the schedule for development projects. The project also shows the main areas of interest, including agriculture, healthcare, energy, and transportation [35, 79].

Just with this first look at Russian legislation on AI, we have seen how it has evolved and which are the main points it is composed of. We also got in touch with the recombination and rethinking of necessities linked to the transformation of Russian legislative approach to AI after the beginning of the war in Ukraine.

It is essential to take a deeper look into the composition of Russian *National Strategy on AI development*, where we can recognize six blocks related to specific fields of interest for the use of artificial intelligence.⁴⁴ So, the six blocks are (1) legal and ethical framework; (2) scientific and academic community; (3) data regulation; (4) hardware and software development; (5) education; and (6) healthcare.

For the current analysis, a better understanding of the first part of the block, the one regarding ethics, is necessary. We can underline how the document shows a focus and concern on the creation of the legal and ethical base for the development of research center, linked with industries, clarifying responsibilities on circulation and results of AI data and elaborating national standards for certifying, testing, and confirming compliance of AI items and technologies [89]. This dimension is profoundly interlinked with the third point, the one regarding data regulation: in fact, the Russian government has tried, deploying this strategy, building a huge database to collect information, the fundamental resource to train artificial intelligence systems, but at the same time, the government has tried to provide

⁴⁴ These different areas composing Russian AI strategy are considered essential by governmental organizations to reach a good degree of competitive skills and take part to the international AI scenario, turning Russia into a world leader in the development of AI technologies: “The strategy stresses the strategic importance of AI as a prerequisite for Russia’s entry into the group of economic world leaders as well as the country’s technological independence and competitiveness. Even though Russia is not currently considered a leader in the realm of AI, the document states that Russia has the potential to ‘[become] an international leader in the development and use of artificial intelligence technologies’ [89].

Russian citizens with specific tools to take care of and anonymize their personal data published online [89]. About data collection, it's interesting to look at the Russian Law No.123-FZ and Russian Law No. 258-FZ approved, respectively, in 2020 and in 2021. These two laws represent an attempt to create a sort of data extraction center in the heart of everyday life: the first one works just for a specific area in Moscow and allows the possibility of using anonymized personal health data to contribute to the development of AI technologies for this research field; the second extends the creation of this kind of special data extraction area to the rest of the country—but with a previous deal between industries and the Ministry of Economic Development.⁴⁵

These are the so-called ELRs,⁴⁶ Experimental Legal Regimes, whose creation can be seen also as a good strategic solution to enhance a qualitative development of artificial intelligence systems.⁴⁷ However, if we try to conduct a deeper analysis, using as guide and filter the Unified Framework developed by Floridi,⁴⁸ we'll find out that Russian *National Strategy on AI development* has numerous and diverse problems, lacking of a real ethical accuracy. In principles such as beneficence or non-maleficence, Russia seems to enter a virtuous path and establish a good approach, but the lack of explicability of the main legal or ethical tools to use in order to face the moral and juridical issues emerging from AI development represents a real obstacle to enhance the credibility of the strategy and to obtain the optimistic objectives—competitiveness and leadership of Russia in the global AI race—the government is looking at [89].

⁴⁵ Russian Law No. 123-FZ established an experimental legal regime (commonly referred to as a regulatory sandbox) in Moscow. Coming into effect in July 2020, its aim was to promote innovation over the course of five years. Importantly, the law contains an amendment to the law “On Personal Data,” passed in 2006, that allows for the processing of anonymized personal health data of Russian citizens. The authors of the bill claim that the amendment is needed to gather the large amounts of information needed for the development and testing of AI technologies. In July 2021, Russian Law No. 258-FZ established a similar regulatory sandbox for all of Russia. It aimed to promote development in AI, quantum technology, big data, and robotics. To participate, entrepreneurs and organizations submit an application to the Ministry of Economic Development for approval https://geohistory.today/artificial-intelligence-in-russia/#National_AI_Initiatives_in_Russia, (Sullivan 2022).

⁴⁶ ELRs proposal initiatives are currently 180 and focused on different activities, from healthcare to unmanned transportation and access for AI developers to governmental datasets. The most important Russian tech and financial industries, including Sber, Yandex, and Kamaz, are involved in the use of these legal regimes and are the main partners of the government for this project. More data and insights can be found in <https://ai.gov.ru/en/ai/regulatory/>, Artificial Intelligence of the Russian Federation Web site.

⁴⁷ Floridi suggests the European Union should give a chance to this model of development: “Incentivare finanziariamente lo sviluppo e l’uso di zone speciali legalmente deregolate all’interno della UE per la verifica empirica e lo sviluppo dei sistemi di IA. Queste zone possono assumere la forma di un “laboratorio vivente” *eng.* “To financially incentivize the development and use of specially designated, legally deregulated zones within the EU for the empirical verification and development of AI systems. These zones could take the form of a ‘living laboratory” [42].

⁴⁸ We spoke about the Framework before in this chapter.

As underlined before, there is an important connection between Russian AI development and the events related to the war in Ukraine. One of the first things that became suddenly evident at the beginning of the war was the delay of Russian AI technologies infrastructure and how the sanctions imposed by Western countries obliged tech industries in Russia to depend more and more on state fundings [13]. This, of course, contributed to a general centralization of AI development in the country,⁴⁹ and with the war being a priority for Russia, also research on artificial intelligence was used to extract major data from the battlefield and, as MOD [*Ministry of Defense*] AI department leader Vasily Yelistratov announced, to develop tools able to guarantee supremacy at war [13].

It is important to enlighten the ethical concern raised by the use of AI in the field of warfare and how the present international scenario of tension—the war in Ukraine is deeply interlinked with artificial intelligence use in Russia, but it's just an example in this period of political and economic concerns⁵⁰—is not helping in creating the best environment for a good and qualitative research of ethical AI. We cannot trust in a future machine-led warfare, where human directions and implications will be deleted; instead, the psychological dilemma caused by the use of this kind of technology on the battlefield is even more evident, a use that is inevitably related to the osmotic relationship human beings develop with the tools they use [58].

For Russia, the situation becomes every day more complex because of the deteriorating political environment of the country, with a spread repression against a however still-existing opposition to the government [45], a hostile political atmosphere that, previously with the post-Soviet crisis and then with the recent sanctions adopted by Western countries against the Federation, led to a loss of talented researchers, evident also in the tech and AI field [13]. A lot of Russians, employed as tech developers or entrepreneurs in artificial intelligence industry, decided to leave the country⁵¹ and search for other occupations abroad. Meanwhile, Russia has been

⁴⁹ “With the launch of sanctions against Russia in March 2022, Russian society looked to the state for long-term solutions and paths forward for the domestic high-tech ecosystem, and for funding and technology development in general. In practice, this meant adopting many centralized development pathways with the government at the epicenter of all relevant activity. While the Russian state was already the major entity in national AI and high-tech development prior to the Ukraine invasion, the government, and the MOD [*Ministry of Defense*], in particular, moved to create centralized structures for AI research, development, and use to streamline approaches, avoid duplication of efforts, and have better control over multiple efforts across the country” [13].

⁵⁰ The widespread tensions are expressed also with the economic threat of tariffs for other countries’ imported goods—as shown by <https://www.reuters.com/world/us/trump-promises-25-tariff-products-mexico-canada-2024-11-25/>, Pitas, 2024—and by the continuous political crisis overthrowing Europe (<https://www.telegraph.co.uk/world-news/2024/12/03/crisis-french-german-governments-terrible-moment-eu/>, Crisp, 2024) and important democracies in Asia (<https://www.bbc.com/news/articles/c0kvjx474g1o>, Walker, Lee, 2024).

⁵¹ “Many talented high-tech workers and entrepreneurs in the broader information and communication technologies (ICT) ecosystem that includes AI emigrated abroad at the start of Moscow’s invasion of Ukraine. At least 70,000 and up to 100,000 tech professionals have left the country—bound for Europe, the United States, the Middle East, and several former Soviet republics. While some continued to work for Russian high-tech companies, many were employed by international

isolated from the international innovation ecosystem—e.g., Russian developers had various issues using GitHub—and Russian technological development depends always more on cooperation with its partners, especially China [13].

10.3.4 EU Approach to AI Development and Regulation: The Anthropocentric Way to AI

The European Union legislation started to regulate digital environments and online identities and actions to preserve from potential dangers the lives of European citizens—the famous Regulation (EU) 2016/679, also known as GDPR, is a good example of this attempt[82]⁵²—but also to boost the European economy through the maximization of the opportunities digital instruments can provide to the economic growth of the continent. That’s why in 2015, during Juncker mandate, as president of the European Commission, the Digital Single Market strategy has been established, reaching in 2016 and 2017 important goals regarding data protection and cross-border portability.⁵³

It was in 2016 when, as said before, the Regulation (EU) 2016/679, the GDPR (General Data Protection Regulation) was released. The EU, with this legislative act, succeeded in governing potential breaches in privacy and possible European citizens and industries data leaks [82].

The GDPR act recognized the importance reached by data in economic and technological developments, considering parallel risks this new condition could create for private citizens. At the same time, one of the declared goals the European Commission looked at was the definition of common laws that could help regulate

companies, while others chose to seek entirely new ICT employment. Many remain abroad, instead others have returned due to difficulties settling in new countries. This emigration will remain a long-term issue for the Russian high-tech industry in general” [13].

⁵² This regulative attempt led also to the creation of a strict approach against any possible abuse of data. Italian governmental data-protection authority, Garante per la Privacy, for example, has been the first to stop two online-accessible LLMs, OpenAI (in 2023) and DeepSeek (2025) services, after some issues emerged exactly in the management of users’ data. The decision was not political and related to the Italian Garante fast reaction in observing European privacy laws. In fact, according to the Garante, OpenAI didn’t accomplish the necessary informative duties that could guarantee Italian citizens awareness on the treatment of their data. However, Italy has just been the first and fastest European country to act, but also Spanish and French authorities started the procedural passages necessary to stop OpenAI in 2023, before OpenAI answered positively to the Italian and European requests. In 2025 also, the mobile app of DeepSeek, a new Chinese LLM, has been stopped, and data-protection authorities in Italy and Ireland asked the Chinese developer to answer to the same doubts on data that emerged before with ChatGPT (<https://legrandcontinent.eu/it/2023/04/21/perche-litalia-ha-bloccato-chatgpt/>, Azzolini, 2023; <https://www.bbc.com/news/technology-65139406>, Shiona MvCullum, 2023; <https://www.theguardian.com/technology/2025/jan/29/deepseek-blocked-some-app-stores-italy-questions-data-use>, Booth, Krupa, Giuffrida, 2025).

⁵³ <https://www.consilium.europa.eu/en/policies/digital-single-market/>, European Council, 2020.

the circulation and use of data inside the European Union [82]. Restrictions imposed by the EU changed profoundly the structures and modalities developers had to attain to get an answer for the new challenges the regulation added. Of course, one crucial point has been compliance, which then regarded all the progressive updates the entire world of digital and informational industries went through—from Web navigation to blockchains and Internet of Things (IoT) [36].

The European Union started to develop a regulative practice that intended preservation online the human rights granted in natural everyday life by the fundamental laws of the Union [82]. Meanwhile, digital assets and infrastructure are seen as crucial resources for the future of the European Union.⁵⁴ The economical frame of digital technologies, including AI, had and still has strong influence and profound consequences on ethical and political decisions, thus drawing particular attention from European policymakers.

This is shown by the intense legislative work that characterized EU law-making in the field of digital market. In 2020, the European Union proposed the Digital Market Act (DMA), the Digital Service Act (DSA), and the Data Governance Act (DGA) [80].

The DMA represents an attempt, directed by the Union, to unify the continental previous fragmented digital market frame and to establish clear borders for the actions of so-called digital market “Gatekeepers” (e.g., Amazon or Google). Such borders are linked to specific prohibitions including data combination across platforms and self-preferencing [83]. The DSA instead has a specific focus on e-commerce, with peculiar attention for potential misuse of information by societies. An interesting aspect of this law, actually contributing to ethical and political debates, deals with digital platforms and the hosting of illegal contents. The Data Service Act does not declare the hosting providers (i.e., Meta, X, etc.) responsible for such contents but creates a simultaneous expansion of procedural obligations to obtain the removal of these illegal materials [97]. Finally, the DGA has the purpose of regulating the reuse of publicly shared and private protected data—with the latter ones being shielded also by the application of the GDPR, shaping a European informational environment where precious datasets are easily accessible. This Act tries to reach its aim appointing novel data management intermediaries and encouraging the sharing of data for altruistic purposes. Another important boost is tried to be gained through the implementation of initiatives to establish a favorable cultural atmosphere to entrust data sharing and reuse, in order to provide more data for the further development of new technologies [84].

Already at the publication of Digital Market Act proposals, not few experts underlined different critical points and potential threats for the European digital market itself [80]. Important lacks emerged in the specificity and flexibility of the act, with real risks for the applicability of the law and for possible future transfor-

⁵⁴ We have already seen before what this means when we referred to the EU strategy developed since 2015.

mations and adjustments related to market development and sectorial necessities⁵⁵ [16, 17].

These Acts and the GDPR are examples of European awareness about the necessity of creating a law-making approach that can regulate, managing data circulation and use, the digital life of EU citizens, giving them the opportunity to access the necessary legal instruments to protect themselves in the complex structure of modern *infosphere*.

In order to engage the global political and economic competition for AI supremacy, the European Union since 2018—one year after Chinese AIDP—understood the importance of establishing a continental strategy to face the new challenges emerging from the diffusion and use of artificial intelligence systems [24]. The EU started to look at AI as a technological revolution that would have completely transformed the entire industrial asset of the continent and the world, reconfiguring societies and ways of life.⁵⁶ Europe was aware that members had to coordinate and harmonize their AI policies to reach the desired goal of creating a strong and valuable development oriented to establish good and for-all AI. So, the European Commission found three strategical points of capitalization:

1. A world-leading environment of researchers, labs, and start-ups, integrated in advanced industries in the fields of healthcare, transports, and manufacture and linked to a peculiar strength in robotics
2. The presence of a common Digital Single Market
3. A wealth of industrial, research, and public sector data that are the core of AI development, raw materials without which these technologies could neither exist [24].

The European strategy wants to set the EU as one of the international leaders in AI technological developments⁵⁷ but with particular regard to an AI-led digital transformation that leaves none behind and is able instead to produce more inclusion and to guarantee a sustainable future for the continent. According to the European Commission, these goals, to create an AI that benefits people and society as a whole, can be reached only by an approach to artificial intelligence based on values [24, 41].

⁵⁵ The DMA is the European law that is creating different problems for the introduction in the continent of new AI foreign products and tools, among them the well-known delay, framed by debates and controversies, for the arrival in Europe of Apple Intelligence AI service (<https://www.theguardian.com/technology/article/2024/jun/21/apple-ai-europe-regulation>, Guardian, 2024).

⁵⁶ “Like the steam engine or electricity in the past, AI is transforming our world, our society and our industry. Growth in computing power, availability of data and progress in algorithms have turned AI into one of the most strategic technologies of the twenty-first century. The stakes could not be higher. The way we approach AI will define the world we live in. Amid fierce global competition, a solid European framework is needed” [24].

⁵⁷ This aim, to obtain AI leadership in a global competitive scenario, was already set in 2017 [23], aligning, from a temporal point of view, the starting moment of the race for EU and China. However, it must be recognized that Europe started to promote research and development of AI technologies since 2004, focusing on the specific sector of robotics [24].

The European strategy considered fundamental also the necessity to develop an AI that could represent a real opportunity for everyone. This means that the socioeconomic changes led by AI had and still have to be governed through specific political actions directed to the enhancement of people's digital abilities. The European AI is accompanied by plans for the reskilling of older citizens—still working or not—for the nurturing of young talents, and of diversity and interdisciplinarity [24]. Of course, in order to obtain these goals the adoption of the new artificial intelligence innovations in Europe needed from the very beginning ethical guidelines. This necessity led to the idea of defining a collective work that included an open exchange of views with the main AI stakeholders [24].

The EU strategy for artificial intelligence, with its human-oriented approach based on values, was released in 2018. Following debates and proposals, including the ones on ethical frameworks and the principal issues emerging when observing AI use in everyday life, resulted in the latest great European regulative initiative: the AI Act. The Artificial Intelligence Act (AIA), published on June 13, 2024, has a clear ethical nature and aims to establish in the EU a human-centered approach that guarantees the development of trustworthy and safe AI models and systems [85]. The Act provides also the legislative base to promote and ensure the free movement of products and services linked to artificial intelligence, avoiding a possible fragmentation of the European AI development and market.⁵⁸

The AI act has, of course, practical consequences on the way providers of AI services must develop and deploy their products; in fact, under the new law, AI systems are classified according to their risks. Unacceptable risks (e.g., social scoring systems and manipulative AI) are absolutely forbidden, while high-risk AI systems are accepted but strongly regulated. A smaller part of the Act is focused instead on limited-risk AI systems (chatbots and deepfakes) that have transparencies obligations, while minimal-risk AI systems are unregulated (e.g., AI-enabled video games and spam filters), but things are changing with generative AI [85]. Developers of AI systems, single individuals or entities that want to enter the European market or put into service in Europe high-risk systems,⁵⁹ have to deal with the majority of obligations. Nonetheless, some legal duties fall also on deployers, natural or legal

⁵⁸ “The purpose of this Regulation is to improve the functioning of the internal market by laying down a uniform legal framework in particular for the development, the placing on the market, the putting into service and the use of artificial intelligence systems (AI systems) in the Union, in accordance with Union values, to promote the uptake of human centric and trustworthy artificial intelligence (AI) while ensuring a high level of protection of health, safety, fundamental rights as enshrined in the Charter of Fundamental Rights of the European Union (the ‘Charter’), including democracy, the rule of law and environmental protection, to protect against the harmful effects of AI systems in the Union, and to support innovation. This Regulation ensures the free movement, cross-border, of AI-based goods and services, thus preventing Member States from imposing restrictions on the development, marketing and use of AI systems, unless explicitly authorised by this Regulation” [85].

⁵⁹ The law applies with no differences also if they are not based inside European borders or if these entities are third-country providers [85].

persons that deploy an AI system—whose output will be experienced in the EU—in a professional capacity [85].

Another important passage of the AIA regards General Purpose Artificial Intelligence (GPAI). The locution “GPAI” was already defined during the legislative debates that occurred in the European Commission before the publication of the AI Act, and it led to a comprehensive idea of GPAI as a system able to pursue generally applicable functions, from image recognition to text translation; it is a model that can be used in different contexts and potentially integrated in other AI systems.⁶⁰ According to the AI Act, GPAI systems providers are subjected to strict control from different points of view: they must provide technical documentation and instructions for use, and then they must comply with the Copyright Directive. Finally, providers are obliged to publish a summary of the contents used for the model training. Unless systemic risks are present, regulations become softer for open and free GPAI developers: they have just the duty to comply with Copyright Directive and to publish a training data summary. Additional legal obligations are instead addressed to potentially risky GPAI providers, including the conduction of model evaluations and adversarial testing; they have also the responsibility to report accidents and ensure cybersecurity protection [85].

We’ve talked before about a risk-based evaluation system that enables the AI Act to regulate artificial intelligence models according to their potential danger for people and society. These are models also considered too risky and whose use, because of this, results forbidden in the EU:

1. Models deploying subliminal, manipulative, or deceptive techniques
2. Models exploiting vulnerabilities related to age, disability, or socioeconomic circumstances to cause significant harm
3. Social scoring models, i.e., evaluating or classifying individuals or groups, causing detrimental or unfavorable treatment of those people
4. Models assessing the risk of an individual committing criminal offenses solely based on profiling or personality traits
5. Models compiling facial recognition databases by untargeted scraping of facial images from the Internet or CCTV footage
6. Models inferring emotions in workplaces or educational institutions

⁶⁰ “General-purpose AI system” is an AI system that—irrespective of how it is placed on the market or put into service, including as open source software—performs generally applicable functions such as image and speech recognition, audio and video generation, pattern detection, question answering, translation, and others; a general-purpose AI system may be used in a plurality of contexts and be integrated in a plurality of other AI systems <https://data.consilium.europa.eu/doc/document/ST-14954-2022-INIT/en/pdf>, (Council of the European Union, 2022). This definition has been widely criticized because it is considered too inclusive and unable to capture the real essence of GPAI [96]. Another definitional proposal came from the Institute for the Future of Life that underlined instead the always-present multitasking performance of these kinds of systems and their ability to accomplish tasks they were not directly trained for <https://artificialintelligenceact.eu/wp-content/uploads/2022/05/General-Purpose-AI-and-the-AI-Act.pdf>, (Future of Life Institute, 2022).

7. Biometric categorization systems
8. “Real-time” remote biometric identification (RBI) in publicly accessible spaces for law enforcement (with some exception, e.g., in terrorism, research of missing people, etc.) [85]

These AI systems cannot be deployed and used in the European Union; they are strictly prohibited.

However, there are a number of models that are considered high risk but can still operate in the Union. Among them, there are AI systems exploited for the management of crucial everyday activities, including critical infrastructure (e.g., road traffic, water or electric supply, etc.), education and vocational training, employment, workers’ management and self-employment, the access to and enjoyment of essential private services and essential public services and benefits (e.g., healthcare), and law enforcement (for the occurring of special-risk conditions) [85]. There are also high-risk systems considered not prohibited dealing with biometric data collection and use. Furthermore, at the center of general ethical concerns and considered high risk by the AI Act are also AI algorithms linked to migration, asylum, and border control management [15].

The providers of high-risk artificial intelligence models have the compliance and transparency duties we have already seen, with the additional responsibility to test their models in order to prevent accidents and answer to possible data vulnerability. They must design their high-risk AI system to achieve appropriate levels of accuracy, robustness, and cybersecurity, allowing at the same time deployers to implement human oversight [85].

The AI Act is the first full attempt to regulate the use of artificial intelligence in a vast jurisdictional area [90], and of course, the regulation has to face different challenges, e.g., choosing the right path to obtain a good harmonized standard system to evaluate AI models [63] and find the best answers for them.

Among these complex and challenging concerns, relevant is the one related to the process of inserting the AI Act inside the multilayered and intricate legal sphere composing the European AI regulative scenario. The AIA will necessarily coordinate with another important legislative tool, the GDPR, the 2016 European initiative that led to a first regulative paradigm for the collection of personal data⁶¹ [38]. We know that data are the fuel of artificial intelligence models [42]; however, the AI Act does not really provide a new regulation for data collection, which remains consequently managed by the previous existing legislative action and thus involving directly the GDPR. It is now fundamental to avoid potential conflicts and contradictions between the two acts and try to understand how, after their collection, these data will be used in AI training and which kind of outputs will emerge. It is clear that the AI Act and the GDPR have a lot of common points [38]. They are both regulations, not directives, and both adopt the same risk-based approach to evaluate possible danger emerging from technologies they regulate. Furthermore,

⁶¹ We have discussed before about the GDPR and its consequences on digital data.

they share also some principles and methods that the AI Act now updates and exploits to manage the training of AI models. Firstly is the *extraterritorial action*, which obligates whoever exports inside the EU borders their artificial intelligence models to respect the rules imposed by the Union. Then, there is also a deep care for the *quality and accuracy of data*, linked to a spread approach attempting to achieve a *privacy by design and by default* AI models production. Other important principles are also *transparency*, the possibility for users and experts to check how data are used and for which purpose, and the *automation of decision-making processes*—the most strict point between the AI Act and the GDPR, it deals with the possibility for data to be processed autonomously by machines that provide consequent outputs and options then used to decide about concerning problems⁶² [38].

However, this similar background cannot give the needed answer for a still-absent procedural joint between the two acts. It is possible to identify three principle critical points:

1. The risk to overload the private sector with legal duties, obtaining a negative effect for technological development.
2. Without general guidelines, companies operating in the AI market will be obliged to self-regulate, and this condition could lead to arbitrary and discretionary evaluations.
3. These uncertainties could generate the paradox of a lack of protection for individual data and privacy [38].

These are the themes policymakers and researchers will encounter in the future discussion on the legislative regulation of artificial intelligence, themes that are still evolving and changing, progressively fueling the debate on ethical and legislative tools capable to develop the European anthropocentric approach to AI.

10.3.5 US Approach to AI Development and Regulation: Keeping Global Supremacy Between Ethical Issues and Future Challenges

The USA is nowadays the leading country in artificial intelligence development and research. It has been calculated that 73% of the production of large language models worldwide is based in the USA, with only a 15% total Chinese contribution to LLM design and development [3]. A substantial gap between the two superpowers can be observed also in the field of research during the first 20 years of the twenty-first century, when the USA, although there has been one year of Chinese supremacy

⁶² It is interesting to underline how the GDPR draws limits for this kind of automation, but not directly forbidding this kind of possible data treatments [82]; instead, the AI Act regulates most of these automated decision-making processes, establishing a specific prohibition for some of them (e.g., in some processes involving biometric data) [38, 82].

in 2010, has always prevailed over its most direct competitor for the number of published AI papers.⁶³

Furthermore, the AI supremacy of the USA is also a matter of investments. In 2020, 82% of global investments in the sector of artificial intelligence start-ups has been parted between China and the USA, with the latter spending a total amount of \$27.6 billion. In 2019, the USA spent \$656 billion in artificial intelligence R&D, attesting itself as the world leader for the AI sector [2].

The incredible achievements of American AI are even more impressive compared to the ones obtained by the European Union. Again investments can give us a clear view of the general landscape, underlining differences and gaps. Private investments for AI development in the USA reached 62.5 billion, while Europe had to sum up its continental investments with those of the UK to obtain a total 9 billion.⁶⁴ It has been observed that between 2018 and 2023, the total investments in EU artificial intelligence companies reached 32.5 billion, while in the USA, tech industries involved in AI development attested at 120 billion investments. Generative AI, and overall OpenAI's ChatGPT model breakthrough,⁶⁵ is of course fueling American competitive power and widening the gaps between the USA and Europe [65]. At the same time, these conditions are increasing the attraction of young European talents for the US market—offering more positions in the AI sector compared to EU and UK—for the stimulating work opportunities and salaries [9].

There is also another point that opens a wide range of debates and discussions: the costs of compliance that European regulative processes require to release in the market new AI models. It has been observed how the GDPR bureaucratic duties contributed to a 2.1% reduction of companies profits, with principle consequences fallen on smaller tech developers [46]. Then, first previsions stated that the AI Act is likely to produce additional compliance costs,⁶⁶ causing a possible 20% decrease of AI investments in the European continent [70].

These complex regulative issues, of course, do not resolve in a sort of pure dichotomy where the European Union appears as a simple overregulating entity trying to choke its own AI development and economy, whereas the USA seems an under- or not-regulated space where tech companies can do whatever they think is necessary for the growth of their profits, avoiding any ethical or political concerns

⁶³ “The U.S. has produced a total of 1,365,452 AI papers (25.23% of global output) while China has produced a total of 957,840 papers (17.70% of global output)” [2].

⁶⁴ However, it is interesting to observe how there are more private investments in Europe—including the UK—than China, whose total funding from private entities stops at 7.3 billion [65].

⁶⁵ ChatGPT has been a game-changer in the AI race between China and the USA. Its online upload at the beginning of the current decade surprised Chinese AI industry, showing for the first time a real delay of the country in AI development and the absence of an already-suited tool to answer American innovation <https://tinyurl.com/theguardianAI-supremacy>, (Koetse, 2024).

⁶⁶ “The AIA will cost the European economy 31 billion over the next five years and reduce AI investments by almost 20%. A European SME [small and medium-sized enterprise] that deploys a high-risk AI system will incur compliance costs of up to 400,000 which would cause profits to decline by 40%” [70].

linked to the diffusion of their products. In fact, we have already seen how the AI Act, the GDPR, and other European actions for the continental digital market pursue instead a data regulation impact able to unify the European markets and to boost the future of AI; and, on the other side, we are going to underline that the USA has already released some regulative directives and how American policymakers gave the appropriate attention for AI-related ethical and political issues.

The USA has released in 2020 the National Artificial Intelligence Initiative Act establishing an American AI initiative to guide research, development, and evaluation for federal science agencies [29]. Other acts, instead, were directed to impose to some federal agencies the development of specific AI programs and policies [28, 30]. Ethical concerns have been suited more directly in the 2023 Blueprint for an AI Bill of Rights, where problems linked to possible biases and prejudices inherited by the AI during the training and the possible misuse of AI systems for social scoring or other risky and threatening activities are put into a defined framework.⁶⁷ This first initiative has been followed then by different other attempts to suggest a path to start creating a more defined legal tool meant to support an ethical and safe development of AI technologies. Among these attempts are included the joint efforts by four federal agencies to avoid technological advancements that could break the law, the release of the AI Risk Management Framework by the National Institute of Standards and Technology, the SAFE Innovation Framework for AI Policy, and Blumenthal and Hawley Comprehensive AI Framework. This is what occurred at federal level; instead, from a local point of view, 14 member states of the federation between 2016 and 2022 have already passed AI-related legislation: Maryland is the leader of this regulative wave, followed by California, Massachusetts, and Washington [76].

On October 30, 2023, former US president Joe Biden issued an Executive Order (EO), yet withdrawn by Trump's new administration [27], on "the Safe, Secure, and Trustworthy Development and Use of AI." A new plan for American AI regulation has not yet been proposed, and Biden's action can be considered today the only known approach to establish an ethical path for American AI. It was built starting from a previous work against discrimination inside federal agencies and from a cooperation with self-committed US companies active in the development of artificial intelligence systems (e.g., Amazon, Google, Meta, OpenAI) [26]. In the EO, there are eight main policy fields:

1. *The release of new standards for AI safety and security* field, where there will be direct contribution of private companies that will share their own data with governmental agencies involved in the elaboration of standardized tools for the future development of safe and trustworthy AI
2. *The protection of citizens' privacy from AI-related risks*, with the direct federal support to the development and research of tools able to measure the impact of AI systems on citizens' private data

⁶⁷ <https://www.whitehouse.gov/ostp/ai-bill-of-rights/#applying>, Blueprint for an AI Bill of Rights, 2023.

3. *The advancement of equity and civil rights* that links the Executive Order to the definition of clear guidelines for landlords and to the establishment of benefits programs, federal contractors, and measures to fight algorithmic discrimination and pursue fairness in justice procedures when AI technologies are used⁶⁸
4. *The protection of consumers, patients, and students*, pursued with measures for the responsible use of AI systems in healthcare and defining how these technologies will be exploited to lead the future of education
5. *Support for workers* who will need a governmental action to limit potential harm coming from AI and a maximization of all the benefits these technologies will bring for their jobs, including plans to mitigate labor-market impacts
6. *The promotion of innovation and competition*, a field in which the EO tries to create the perfect environment to boost research and to guarantee the birth and growth of an open and fair AI ecosystem across the country, able to attract also high-skilled immigrants who will contribute to improve the American artificial intelligence sector⁶⁹
7. *The advancement of US leadership abroad*, requiring the expansion of AI engagements in order to promote the design process of global vital standards for AI⁷⁰ but also making possible a safe and responsible development and deployment of artificial intelligence systems abroad to face ethical and political challenges on a global level
8. *Responsible and effective government use of AI*, linked to the executive order need to seek new AI-related guidelines for federal agencies that use these technologies to enhance the quality of the process used to acquire AI products or services and to hire the sector most qualified experts [26]

President Biden's executive order shared and reaffirmed the values and strategic wills already expressed in the National Artificial Intelligence Research and Development Strategic Plan. In this document, the USA offered a national view on artificial intelligence that looks for a human-AI collaboration, focusing on designing user-oriented technologies that privilege user experience and the protection of human agents in the interaction with the machine [71]. The Plan recognized the necessity to develop AI technologies able to advance core values through sociotechnical system design and understanding the ethical, legal, and societal implications of AI systems [71].

⁶⁸ This point of the EO tries to avoid potential injustices emerging from the use of models trained through biased collections of data, an issue that is also linked to the wider problem of apparently always fair algorithmic decision-making [86].

⁶⁹ And it has already been noted how this is effectively happening for European talents, progressively moving to the USA for better and more adequate job opportunities [9].

⁷⁰ An already-existing example of these efforts could be the EU-US Research Alliance in AI for the Public Good, a first attempt to create a sort of transatlantic cooperation, involving the USA and the European Union, to develop artificial intelligence systems that aim to face important challenges such as extreme weather and climate events, optimization of energy, and responses to emergencies [95].

These efforts involved different federal agencies and institutions. The Department of Defense (DOD) has released the Responsible AI toolkit used to check if AI projects align to DOD AI ethical principles, while the Department of Education is defining the needed policies and guidance for safe and accessible uses of AI in schools and educational institutions. This work will engage teachers with the creation of an adequate toolkit for the implementation of past recommendations and also developers who will provide resources to generate trust for the AI marketplace and essential questions to guide future AI responsible development. At the same time, the Education Department is collaborating with the Department of Justice to address discrimination and all the other possible harms AI can bring to the preservation of civil rights in the USA [8]. Inside the US Department of Health and Human Services, the National Institutes of Health (NIH) is among the main actors of American AI R&D sector, producing big efforts to protect the privacy and well-being of the patients involved in researches. To reach this data-preservation goal, they have produced policies such as the Common Rule and the NIH Certificates of Confidentiality. Furthermore, the National Institute of Justice (NIJ) established the Predictive Forensic Deoxyribonucleic Acid (DNA) and Predictive Policing efforts that fund workshops on probabilistic genotyping, forensic DNA phenotyping, and forensic investigative genetic genealogy technologies. The same institution worked also on the Strengthening Data-Driven Pretrial Releases in New Jersey effort, an interesting action conducted to revise the Public Safety Assessment (PSA) tool, an AI-based juridical technology that had the aim to give objective pretrial release recommendations, resulting often in bias. The NIJ, exploiting ML and data analysis instruments, is trying now to face the inner discrimination problems the PSA revealed in the past, trying to guarantee high-level ethical standards [8].

Despite the absence of a structured answer for artificial intelligence ethical and political problems—as the European AI Act instead is—it has just been observed how the USA is demonstrating a real concern for the possible harms emerging from the use of these new technologies. This does not mean, of course, that the debate can be considered now unnecessary. In fact, Republican Senator Ted Cruz added to the Future of Artificial Intelligence Innovation Act of 2024 [31] an amendment that directly prohibits the development of AI technologies designed with processes that ensure inclusivity and equity.⁷¹ He claimed that this action will preserve American artificial intelligence progress, avoiding possible future deceleration and the potential overcoming from Chinese AI industry.⁷²

This example makes clear how multilayered and complex the debate on the best way to build a proper ethical regulative path to ensure the future of AI-human relationship can still be.

A last focus can be directed to the issues related to generative AI and copyright. As we have seen, data are at the core of artificial intelligence models development, the real fuel that allows their training and the production of brilliant output.

⁷¹ <https://tinyurl.com/commerce-gov-senate>, Senator Cruz, 2024.

⁷² <https://tinyurl.com/govtech-policy>, Scott J. Mulligan, 2024.

ChatGPT, a generative pre-trained transformer model, since its release, generated hype, spreading at the same time tension, fear, and also fascination and interest for the world of artificial intelligence, large language models, and automated generation [21]. It was also a turning point and a game-changing technology that helped establish a renovated sense of American supremacy in the field of AI.⁷³ However, these generative models were trained through data obtained from the Web and then copied and analyzed by machines.⁷⁴ All these operations require author consent, because they can violate reproduction rights and must consider the right to control derivative works, since the training is based on the data scraped online [73].

In the USA, this situation turned into a wide movement of protests that then led to a class action from the Authors Guild [73] and the publication of an online petition that wants to forbid the use of these contents for AI training and that defines generative AI use of authors' works as "a major, unjust threat to the livelihoods of the people behind those works," a petition that was signed by mostly forty thousand people, among them also important artists as Thom Yorke.⁷⁵

Federal Trade Commission has already opened an inquiry on OpenAI, trying to understand the sources of the training data exploited for the development of ChatGPT, and also the European Union is trying to develop new rules to manage AI-related copyright issues [73]. The debate on copyright is intriguing and still in progress. Some researchers and experts look at machine learning as a humanlike form of education, saying that none would ask for their royalties because their works activated a learning process.⁷⁶ Others instead see in the reform of licensing a possible future for the creation of a data market that will protect authors and that will let the AI industry flourish too [73].

What has just been introduced shows how, in the field of legislative and ethical approach to AI, the USA is today a laboratory,⁷⁷ a not-yet-defined regulative area where contradictions and experiments show us the potentiality of new AI

⁷³ A supremacy that translated also in the growth of private investments [65].

⁷⁴ Processes, and following issues, have already changed today: some models are thought to be trained starting from contents and data produced by other competitors of generative AIs (<https://techcrunch.com/2024/12/27/why-deepseek-new-ai-model-thinks-its-chatgpt/>, Wiggers, 2024). It happens also because a growing part of Web contents, almost the 90% in 2026 according to previsions, will be composed of AI-generated materials [62].

⁷⁵ <https://www.aitrainingstatement.org/>, Statement on AI training, 2024 (last seen on December 29, 2024).

⁷⁶ The idea of a cooperative use of online resources for learning and collective growth are at the base of the Internet media revolution[57]. It is quite complex nowadays to detach that kind of human-oriented approach from the contemporary one where machine learning does not mean the same thing as what education means for human growth and life enhancement [73].

⁷⁷ In the copyright field related to AI, a good example of US constant attempts to innovate and create solutions can be found in the recent publication of a new report by the US Copyright Office. The report affirms that a content generated by LLMs and then edited by humans—so, the simple prompt is not enough—can be considered a new human-generated work, and consequently it can be protected by the laws of copyright [78].

technologies, a potential strength whose outputs must be deeply understood and governed in order to let human beings construe new and better worlds.

Key Points of Rules for Artificial Intelligence

- *The Role of Ethics and Political Governance in AI*
 - Ethical concerns are shaping AI regulations, especially in the EU's human-oriented approach to artificial intelligence.
 - The concept of Emergency and the birth of Surveillance capitalism: Data extraction, behavior prediction, and manipulation.
 - AI-driven re-ontologization and transformation of humans from aims, beneficiaries of digital technologies, into means of production.
 - Governmental role in AI-driven threats and social control.
 - Case studies:
 - China and Russia: AI as a tool for surveillance and governance.
 - USA and EU: Use of emergency conditions to boost data extraction and AI technologies able to limit Freedom and Rights.
- *China's AI Strategy*
 - New Generation AI Development Plan (2030 Goal: Global Supremacy on AI)
 - AI as a pathway to global technological leadership.
 - AI's impact on labor and workforce concerns.
 - CCP's social control through AI: Algorithmic governance, censorship, and adaptation strategies.
- *Russia's AI Approach*
 - Instability affecting AI development.
 - AI ethics and data collection policies:
 - Special legal zones for data extraction.
 - AI in military applications and warfare.
- *The European Union's AI Strategy*
 - Creation of a single and unified European area to collect, manage, and fair use data for economic development (DMA,DGA,DSA): a regulative approach that contributed to enhance EU digital market, creating at the same time some delays in innovation.
 - Human-centric AI regulation:

(continued)

- GDPR (General Data Protection Regulation): governing potential breaches in privacy and possible European citizens and industries data leaks
- AI Act (AIA): Risk-based classification approach.
- Ensures trustworthy and safe AI models and systems.
- Promotes free movement in the Union of AI-related products and services.
- Prevents fragmentation in the European AI market.
- Risk-based classification of AI systems.
 - Unacceptable risk: Strictly prohibited (e.g., social scoring, manipulative AI).
 - High-risk AI: Allowed under strong regulations.
 - Limited-risk AI: Transparency obligations (e.g., chatbots, deepfakes).
 - Minimal-risk AI: Unregulated (e.g., AI in video games, spam filters).
- The AI Act and GDPR: legal harmonization challenges.
 - GDPR remains the main regulation for data collection.
 - The AI Act focuses on how AI systems use data.
 - Key shared principles:
 - Extraterritorial scope: Compliance required for all AI entering the EU market.
 - Data quality and accuracy: Ensuring high-standard AI training.
 - Privacy by design and default: AI systems must respect privacy norms.
 - Transparency: Clear understanding of AI decision-making processes.
 - Automated decision-making: Regulated under both GDPR and AI Act.
- *US AI governance*
 - US dominance in AI R&D and investments.
 - Federal legislative actions:
 - Biden's Executive Order 14110, already withdrawn by Trump's administration, but yet the only known approach to American AI. Key elements: AI privacy, consumer protection, and technological supremacy.
 - New standards for AI safety and security.
 - Protection of citizens' privacy.
 - Promotion of equity and civil rights.
 - Consumer, patient, and student protections.

(continued)

- Support for workers impacted by AI.
- Encouragement of AI innovation and competition.
- Strengthening US leadership in AI globally.
- Responsible government use of AI.
- AI policies across different institutions (DoD, NIH).
- Ethical and political debates on AI in the USA
 - Senator Ted Cruz’s amendment against inclusivity-driven AI.
 - The future of Artificial Intelligence Innovation Act.
- Generative AI and copyright issues
 - AI models training on copyrighted data.
 - Authors Guild class action and petition against AI use of copyrighted works.
 - FTC inquiry into OpenAI’s data sources.
 - Emerging discussions on AI licensing and data markets.
 - US Copyright Office ruling on AI-assisted human-generated works.

10.4 Conclusions: Censorship, Openness of Data and Models, Freedom, and Rights

10.4.1 *Censorship*

In this first part of the conclusive section, the previous overview on general ethical approach to AI and on the way it has been interpreted in legislative actions around the world will be used to allow the research to outline the way censorship operates in the field of AI.

It has been underlined how the Chinese government has tried, with its direct control on algorithms and data, to operate and orient the outputs obtained with the use of generative LLMs, technological output that has been defined in Chinese AI sector as deep synthesis [91]. The censorship is here directed by the central government, led by the Chinese Communist Party, to avoid possible harms or subversive propaganda able to represent a real threat for the stability of the government itself. In China, this means no algorithmic freedom: the process and data from the very beginning are censored, asking developers to create technologies adherent to specific moral values and not disturbing for the social and economic spheres.

This need for censorship translated also in an advancement in political and legislative actions to regulate these technologies, with new processes optimized to afford changes and mutation on the algorithmic side [91].

Russia, although not directly working on artificial intelligence, has demonstrated great interest for data and the way they are used and diffused. The analysis has shown before how ELRs, special zones for data extraction, have been created in Moscow and all around the country,⁷⁸ but this has more effects on data freedom and personal rights than on data censorship. It is instead the *Yaroyava Law*, a law apparently directed to fight terrorism, that has increased surveillance and censorship on the Internet, setting also governmental requirements for data retention that of course affect directly the development of AI models [98].

Interesting also is the debate on forms of censorship not directly related to social control but to the outputs AI generative models can produce: censoring processes that affect and regard also the debate in Europe and the USA.

The European anthropocentric approach presented before tries to protect users from potential threats deriving from AI technologies [85], and also in the USA, these actions took place to preserve the well-being of citizens [26]. However, also in this areas of the world, AI is used to detect contents considered potentially dangerous. A good example is what happens in online social media platforms, where algorithmic control operates to moderate concerning published contents (pornography, hate speech, etc.) [66]. This situation has spread concerns among many of the possibility that algorithmic detection and content censorship can undermine the necessary freedom of speech that lies behind the complex freedom structure of democracy [55].

Also LLMs and TTI (text-to-image) developers have been accused of operating a selected censorship, producing outputs that limit the use range of generative models to obtain specific outputs. It became a great issue widely discussed in the field, not just of ethics but also of aesthetics and artistic debate. The discussion wanted to analyze and comprehend the nature of models as artistic instruments—not immediately related to diffusion and publication of these contents—trying to underline how a sort of puritan censorship (e.g., the production of nudity) cannot be applied to the production of personal works of art.

A solution to avoid censoring restrictions imposed in initial model training is through the use of fine-tuning and RAG techniques. Of course, open-source models result more suit for this purpose [34].

However, nowadays, a process where content moderation results less relevant for developers is starting to affirm, producing new controversies and stimulating further debates.⁷⁹

⁷⁸ These actions of the Russian government have been deeply analyzed in the previous section of the research.

⁷⁹ <https://techcrunch.com/2025/02/16/openai-tries-to-uncensor-chatgpt/>, Zeff, 2025.

10.4.2 *Openness of Data and Models*

Another major topic that occurred to be central in the previous analysis is the one of collection, free movement, and open use of data: meaningful circumstances that are at the core of artificial intelligence and generative artificial intelligence development.⁸⁰

The European Union has tried to create, since the GDPR release in 2016 and with other market regulation actions, a unified and safe network where data can be easily accessible by different actors. The Digital Governance Act (DGA) especially contributed to the definition of this European scenario, regulating public and private data and shielding the latter through the cooperation with the GDPR[84].

This is an institutional solution that aimed to propose a model of development that could lead to the development of technologies based on open data, able to enhance the quality of specific sectors—for example, healthcare—through all Europe [84].

Open data are easily accessible to everyone with minimal or absent economic barriers [77], a horizontal approach that consents an efficient diffusion of information, enabling researchers and organizations to produce innovation in an easier way. This confirms how EU legislation has tried to enhance equal opportunities, offering a real anthropocentric paradigm of digital development.

Openness of data is the first step that has led and will lead to design open artificial intelligence models, including of course generative models. However, these are multilayered and complex technologies that must be deeply analyzed before they can be considered really open. It is not sufficient that their training data are accessible, but it is necessary to consider also other elements:

1. The openness of models' constitutive algorithms
2. The openness of the researches that brought to the realization of the model
3. The possibility to obtain a real open access [100]

Openness is not just an important ethical issue—as it has already been seen in previous passages of this research—because it is a necessary element to achieve qualitative research outputs with less economic efforts [77] and also because it can be used as a manipulating label to promote developers of models and services that call themselves open while respecting real and complete openness just partially [100]. This is the main reason why the term “openwashing” has started to spread in order to define models that attract users with their self-defined openness, without really offering then a real complete open product. This controversy has been partially solved by White et al. with the proposed Model Openness Framework that ranks machine learning models according to five principles: open science, open source, open data, and open access [100].

⁸⁰ It has been already shown before the importance of data for the development of these technologies.

Open-source models, although not being the best options in terms of performance, will guarantee a democratized access and better reproducibility, strengthening at the same time human supervisions⁸¹ [67].

Their use could become also a good resource to overcome possible censoring mechanisms introduced in closed-source models, turning open-source AIs (e.g., open TTI models) into personalized powerful instruments for free artistic production and research [34]. In this specific field, the debate on how these models should be trained has also become interesting. Important discussions involved the copyright issue too. As seen before, artists, especially in the USA, organized to protect their works from being used for models training, but it is not a so-unified front.

The Italian artist and philosopher Francesco D’Isa proposed an open use of artistic works and expressions to train models in order to expand the horizon of machines’ cognitive world.⁸² He claims that this approach can represent a more appropriate path to create instruments with the real ability to help humans realize their artistic aims.

D’Isa thus suggests an entirely open approach where models can be explored and rearranged for the artist’s aims, asking at the same time artists to remove any restriction that does not allow full access for the creation of a wider and more representative training dataset for machines [34].

10.4.3 Freedom and Rights

The previous section centered on AI regulations that helped the reader face the principle ethical issues related to the use of AI and the potential dangers for human rights and freedom that can emerge.

It has been observed how the Chinese Communist Party decided to exploit artificial intelligence power to gain and preserve social control over the Chinese population with the arrangement of a social credit system [91]. AI has been used by the CCP also to obtain biometric data and then to repress the Uighurs Muslim minority’s attempt to affirm their identity, an attempt that helped also the infiltration of terroristic organizations inside the region, producing a strong controlling action by the central government through the use of AI [98].

Another concerning issue involving AI and potential dangers for freedom and rights regards the implementation of Experimental Legal Regimes (ELR) in Russia. It has been shown before how in these special areas the possibility of collecting data became easier, exposing directly the privacy of Russian citizens and enhancing the

⁸¹ An aspect that has been introduced before examining the *Unified Framework of Five Principles for AI in Society*.

⁸² He refers directly to the famous Wittgenstein’s idea of the perimeter of a person’s own language as the perimeter of his or her own world. Machines can be properly used to represent something if their knowledge of the world is expanded and contemplates different points of view and sensibilities [34].

quality of social controlling tools developed by the government.⁸³ These data are not used only to face possible security challenges but also to recognize protesters and silence dissent [98].

Furthermore, Russia is increasing the use of AI technologies in order to turn them into instruments fitted for surveillance and propaganda. Thus algorithmic control is used to limit the free use of platforms and Internet services. For example, the Roskomnadzor (The Federal Service for Supervision of Communications, Information Technology and Mass Media) is a Russian federal office that uses artificial intelligence tools to scan Web publications, find contents potentially threatening for governmental order, and ban them. Other uses of AI to limit the freedom of citizens have increasingly spread after the beginning of the Ukrainian war and have seized the possibility to use, if not authorized, VPN services [98].

These uses of artificial intelligence models for the collection of private data and surveillance have been pointed out by the European Union as major threats to the freedom and rights of European citizens. As seen before, the Artificial Intelligence Act reveals these preoccupations and tries to give a strong answer [85]. The European legislative action forbids artificial intelligence systems that could be used for social scoring, for emotional tracking at work and in educational spaces, and for biometric and real-time biometric categorization [85]. As already noted, the prohibition of this last typology of AI technologies has some exceptions. Biometric data can be used to recognize and act against individuals in moment of emergency, for example, when someone is missing or during a terrorist attack [85].

Emergency has played, as Shoshana Zuboff has already noted [103], an important role in the development of digital surveillance techniques that have undermined and still undermine freedom and personal rights. The dot-com bubble economic emergency produced the necessity for big players in the field of digital technologies (e.g., Google or Meta) to turn private data into profits without users' consent and a condition of social and political emergency, following the 9/11 terrorist attack, and drew institutional interest to the development of artificial intelligence algorithms able to guarantee control and surveillance [103].

So, these exceptions in the European AIA must be regarded with extreme attention and concern, because they can represent a possible breach for future political misuses of AI also in the European Union. Dealing with these possible emerging issues must be considered essential, to consider the critical behavior suggested by the Heraclitus-Heideggerian approach described in the first section of this research, an approach that looks beyond the simple perimeter of the law to embrace a real and deep care of the stay.

A last point that must be underlined in this conclusive passage is strictly related to the amendment, mentioned before, and added by the American Republican Senator Ted Cruz to the Future of Artificial Intelligence Innovation Act of 2024 [31]. It explicitly forbids the development of AI models that ensure inclusivity and equity.

⁸³ https://geohistory.today/artificial-intelligence-in-russia/#National_AI_Initiatives_in_Russia, Sullivan 2022.

Cruz affirms that these two conditions could represent a limitation for the competition of American AI in the global scenario.⁸⁴ However, this vision seems to lack a more articulated comprehension of what inclusivity and equity can guarantee. It has been shown how equity and inclusivity are not limits but the fundamental resources to provide artificial intelligence models with datasets that propose a wider and more complex vision of the real dynamics existing in human societies [12]. Models designed following this inclusive vision, through the application of specific rules, have a wider perspective of what the real world is and can contribute better to include and guarantee rights and freedom of underrepresented social groups [12].

It is a paradigm not so different from the one proposed by the approach of D’Isa to the openness of data [34]. A wider dataset, able to include differences and alternative visions, can comprehend better different necessities of users and can really know and understand multiple sensibilities. A not-inclusive and not-equal model must be simply considered a biased model [12],

a vision that recalls again the condition of *ἦθος* (*ēthos*) as the care for the stay, able to look at the borders offered by the law and to overcome them for a wider and more essential comprehension of reality, a paradigm that can really reactivate critical thinking in the use and development of artificial intelligence.

Key Points of Conclusions

- Censorship in AI emerges as a key issue in ethical and legislative discussions.
- Analysis of censorship mechanisms in different geopolitical contexts.
- *China: State-controlled algorithmic censorship*
 - Chinese government directly controls algorithms and data with a registry.
 - Censorship is implemented at the development stage, ensuring AI aligns with state-approved moral and social values.
 - Legislative advancements optimize AI regulation, adapting to technological mutations.
- *Russia: surveillance and data control*
 - Focus on data collection.
 - ELRs (Extractive Logic Regions) established for data extraction.
 - Yaroyava Law increases state surveillance, requiring extensive data retention, affecting AI development.

(continued)

⁸⁴ <https://tinyurl.com/ovtech-ai-bills>, Scott J. Mulligan, 2024.

- *Forms of censorship in the EU and the USA: AI and content moderation*
 - AI used to detect and remove harmful content (e.g., hate speech, pornography).
 - European and US regulations prioritize user protection.
 - Among many, concerns arise over AI moderation undermining the freedom of speech that lies behind the complex structure of democracy.
- *Generative models and censorship*
 - Accusations of selective censorship in LLMs and TTI models.
 - Debate on AI as an artistic tool: should nudity and controversial topics be restricted?
 - Open-source models and fine-tuning/RAG techniques as solutions to bypass initial censorship.
 - Shift toward reduced content moderation, sparking new ethical debates.
- Openness of data and models as a central theme in AI development.
- Focus on the free movement, collection, and open use of data.
- *Advantages of open data*
 - Minimizes economic barriers, ensuring broad accessibility.
 - Facilitates research, innovation, and fair digital development.
 - Supports the EU's anthropocentric paradigm in digital governance.
- *Openness in AI models: a multilayered concept*
 - Openness is not only about training data but also:
 - Transparency in model algorithms.
 - Open access to research behind the models.
 - Real accessibility and the removal of any form of restriction.
- *The issue of “openwashing”*
 - Some AI developers claim openness without fully implementing it.
 - Model Openness Framework (White et al.) ranks AI models based on:
 - Open science
 - Open source
 - Open data
 - Open access
- *Open-source AI: advantages and controversies*
 - Ensures democratized access, reproducibility, and human supervision.
 - Helps bypass censorship in closed-source models.
 - Facilitates free expression, creation, and research.

(continued)

- *Openness and copyright debates for AI training*
 - Artists (especially in the USA) oppose AI training on their works.
 - Francesco D’Isa’s Perspective: advocates open artistic data for AI training.
 - Suggests models should be freely explored and rearranged for artistic purposes.
- *Freedom and rights*
- Ethical issues and human rights risks emerging from AI applications.
- *AI as a tool for social control:*
 - China: social credit system and minority repression
 - AI used by the Chinese Communist Party (CCP) for mass social control.
 - Biometric data collection and surveillance targeting Uighur Muslims.
 - Justification: emergency conditions that allow counterterrorism efforts, leading to strict government oversight.
 - Russia: Experimental Legal Regimes and AI surveillance
 - Experimental Legal Regimes (ELR): Special zones easing data collection.
 - AI-enhanced surveillance identifies protesters and suppresses dissent.
 - Roskomnadzor: AI-powered censorship and control of online content.
 - Increased restrictions post-Ukraine war, including bans on unauthorized VPNs.
- *European Union’s legislative response*
 - Artificial Intelligence Act (AIA): addresses AI-related threats to freedom and privacy.
 - Concerns over exceptions: emergency-based AI use may create loopholes for misuse.
- *Surveillance capitalism and AI’s role in political power*
 - Shoshana Zuboff’s analysis:
 - AI surveillance developed in emergency conditions due to economic crises and security fears.
 - Dot-com bubble → monetization of user data (Google, Meta, etc.).
 - Post-9/11 → government interest in AI surveillance.
 - AIA’s role: needs scrutiny to prevent future political misuse in the EU.

(continued)

- *US policy and AI ethics debate*
 - Ted Cruz’s amendment (Future of AI Innovation Act, 2024):
 - Prohibits AI models prioritizing inclusivity and equity to protect US AI competitiveness.
 - Critique: inclusivity as an AI strength
 - Equity and inclusivity improve AI dataset diversity and representation of reality complex structure [12].
 - Ensures better representation of underrepresented groups.
 - Prevents biased AI outcomes.
- *Inclusivity, data openness, and ethical AI development*
 - D’Isa’s approach: open datasets enhance AI’s knowledge—this guarantee a better impact for AI in artistic and cultural fields.
 - Wider datasets improve AI’s understanding of human sensibilities.

References

1. Akgun, S., Greenhow, C.: Artificial intelligence in education: Addressing ethical challenges in k-12 settings. *AI Ethics* **2**(3), 431–440 (2022)
2. AlShebli, B., Memon, S.A., Evans, J.A., Rahwan, T.: China and the us produce more impactful ai research when collaborating together. *Sci. Rep.* **14**(1), 28576 (2024)
3. Analysis and Research Team.: Chatgpt in the public sector – overhyped or overlooked? Council of the European Union, pp. 1–23 (2023)
4. Arendt, H.: *Vita Activa*. Piper, Girvan (1967)
5. Aristotle: *De Partibus Animalium*. Harvard University Press, Cambridge (1911)
6. Aristotle: *The Nicomachean Ethics*. Harvard University Press, Cambridge (1934)
7. Arruda, E.P., Arruda, D.P.: Artificial intelligence for sdg 4 of the 2030 agenda: Transforming education to achieve quality, equality, and inclusion. *Sustain. Econ.* **2**(2), 34–34 (2024)
8. Artificial Intelligence Research and Development Interagency Working Group: 2020–2024 progress report: Advancing trustworthy artificial intelligence research and development. Technical Report, Executive Office of the President of the United States (2024)
9. Atomico: *State of European Tech 2024* (2024)
10. Ayeni, O.O., Al Hamad, N.M., Chisom, O.N., Osawaru, B., Adewusi, O.E.: Ai in education: a review of personalized learning and educational technology. *GSC Adv. Res. Rev.* **18**(2), 261–271 (2024)
11. Bailey, J.: Ai in education. *Edu. Next* **23**(4), 29–36 (2023)
12. Bartoletti, I., Xenidis, R.: Study on the impact of artificial intelligence systems, their potential for promoting equality, including gender equality, and the risks they may cause in relation to non-discrimination. Council of Europe Publishing, Strasbourg (2023)
13. Bendett, S.: *The Role of AI in Russia’s Confrontation with the West*. CNAS (2024)
14. Berardi, F.: *Contro il lavoro*. Edizioni della Libreria (1970)
15. du Boucher, G.: Les algorithmes et l’intelligence artificielle contre les étrangers en europe. *Plein droit* **140**(1), 31–34 (2024)

16. Cabral, L., Haucap, J., Parker, G., Petropoulos, G., Valletti, T.M., Van Alstyne, M.W.: The EU digital markets act: a report from a panel of economic experts. In: Cabral, L., Haucap, J., Parker, G., Petropoulos, G., Valletti, T., Van Alstyne, M. (eds.) *The EU Digital Markets Act*, Publications Office of the European Union, Luxembourg (2021)
17. Caffarra, C., Morton, F.S.: *The european commission digital markets act: a translation*. VOXEU (2021)
18. Calabrese, S., Conti, V.: *Che cos' è una fanfiction*. Carocci editore, Rome (2019)
19. Cardona, M.A., Rodríguez, R.J., Ishmael, K., et al.: Artificial intelligence and the future of teaching and learning: Insights and recommendations. In: *Pelita Internasional Conference (PIC)* (2023)
20. Cazzaniga, M., Jaumotte, M.F., Li, L., Melina, M.G., Panton, A.J., Pizzinelli, C., Rockall, E.J., Tavares, M.M.M.: *Gen-AI: Artificial Intelligence and the Future of Work*. International Monetary Fund, Washington (2024)
21. Chan, A.: Gpt-3 and instructgpt: technological dystopianism, utopianism, and “contextual” perspectives in ai ethics and industry. *AI Ethics* **3**(1), 53–64 (2023)
22. Coeckelbergh, M.: Ai for climate: freedom, justice, and other ethical and political challenges. *AI Ethics* **1**(1), 67–72 (2021)
23. European Commission: Communication from the commission to the european parliament, the european council, the council, the european economic and social committee and the committee of the regions, artificial intelligence for europe, SWD(2018) 137 final. Communication (eu), **155** (2017)
24. European Commission: Communication from the commission to the european parliament, the european council, the council, the european economic and social committee and the committee of the regions, artificial intelligence for europe, SWD(2018) 137 final. Communication (eu) **132** (2018)
25. European Commission: *Digital Education Action Plan. 2021–2027*. European Parliament Luxembourg, Belgium (2020)
26. Executive Office of the President: Exec. order 14110: Safe, secure, and trustworthy development and use of artificial intelligence (2023)
27. Executive Office of the President: Exec. order 14179: Removing barriers to american leadership in artificialintelligence (2025)
28. Congress.gov: H.r.2575 - 116th congress (2019–2020): AI in government act of 2020 (2020)
29. Congress.gov: H.r.6216 - 116th congress (2019–2020): National artificial intelligence initiative act of 2020 (2020)
30. Congress.gov: S.1353 - 117th congress (2021–2022): Advancing american AI act. (2022)
31. Congress.gov: S.4178 - 118th congress (2023–2024): Future of artificial intelligence innovation act of 2024 (2024)
32. COWLS, J., Tsamados, A., Taddeo, M., Floridi, L.: The ai gambit: leveraging artificial intelligence to combat climate change—opportunities, challenges, and recommendations. *Ai Soc.* **38**, 1–25 (2023)
33. Di Piazza, A., Di Piazza, M.C., La Tona, G., Luna, M.: An artificial neural network-based forecasting model of energy-related time series for electrical grid management. *Math. Comput. Simul.* **184**, 294–305 (2021)
34. D’Isa, F.: *La rivoluzione algoritmica delle immagini*. Luca Sossella Editore (2024)
35. Edmonds, J., Bendett, S., Fink, A., Chesnut, M., Gorenburg, D., Kofman, M., Stricklin, K., Waller, J.: *Artificial intelligence and autonomy in Russia*. CNA (2021)
36. El-Gazzar, R., Stendal, K.: Examining how GDPR challenges emerging technologies. *J. Inf. Policy* **10**, 237–275 (2020)
37. Ernst, E., Merola, R., Samaan, D.: Economics of artificial intelligence: Implications for the future of work. *IZA J. Labor Policy* **9**(1), 1–35 (2019)

38. Falletta, P., Marsano, A.: Intelligenza artificiale e protezione dei dati personali: il rapporto tra regolamento europeo sull'intelligenza artificiale e GDPR. *Rivista italiana di informatica e diritto* **6**(1), 119–137 (2024). <https://doi.org/10.32091/RIID0155>
39. Floridi, L.: A defence of informational structural realism. *Synthese* **161**(2), 219–253 (2008)
40. Floridi, L.: *La quarta rivoluzione: come l'infosfera sta trasformando il mondo*. Raffaello Cortina Editore (2017)
41. Floridi, L.: The european legislation on ai: A brief analysis of its philosophical approach. *Philos. Technol.* **34**(2), 215–222 (2021)
42. Floridi, L.: *Etica dell'intelligenza artificiale: Sviluppi, opportunità, sfide*. Raffaello Cortina Editore (2022)
43. Floridi, L., Cowls, J.: A unified framework of five principles for ai in society. In: *Machine Learning and the City: Applications in Architecture and Urban Design*, pp. 535–545. Wiley, Hoboken (2019)
44. Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., et al.: Ai4people—an ethical framework for a good ai society: opportunities, risks, principles, and recommendations. *Minds Mach.* **28**, 689–707 (2018)
45. Franceschelli, M.C., Varese, F.: *La Russia che si ribella. repressione e opposizione nel paese di Putin* (2024)
46. Frey, C.B., Presidente, G.: Privacy regulation and firm performance: Estimating the gdpr effect globally. *Econ. Inquiry* **62**, 1074–1089 (2024)
47. Gailhofer, P., Herold, A., Schemmel, J.P., Scherf, C.S., de Stebelski, C.U., Köhler, A.R., Braungardt, S.: *The Role of Artificial Intelligence in the European Green Deal*. European Parliament Luxembourg, Belgium (2021)
48. Graeber, D.: *Policy for the Future of Work*. Springer, Berlin (2020)
49. Guilherme, A.: AI and education: the importance of teacher and student relations. *AI Soc.* **34**, 47–54 (2019)
50. Han, B.C.: *Le non cose: come abbiamo smesso di vivere il reale*. Einaudi (2022)
51. Heidegger, M.: *Über den Humanismus*. Vittorio Klostermann (1949)
52. Heidegger, M.: *Die Technik und die Kehre* (1962). Aufl. Pfullingen (1962)
53. Heidegger, M.: *Sein und Zeit*. M. Niemeyer Tübingen (1977)
54. Henderson, P., Sinha, K., Angelard-Gontier, N., Ke, N.R., Fried, G., Lowe, R., Pineau, J.: Ethical challenges in data-driven dialogue systems. In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 123–129 (2018)
55. Hidy, K.M.: The speech gods: Freedom of speech, censorship, and cancel culture in the age of social media. *Washburn LJ* **61**, 99 (2021)
56. Ise, T., Oba, Y.: Forecasting climatic trends using neural networks: an experimental study using global historical data. *Front. Robot. AI* **6**, 446979 (2019)
57. Jenkins, H.: *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. The MIT Press, Cambridge (2009)
58. Johnson, J.: The AI commander problem: Ethical, political, and psychological dilemmas of human-machine interactions in AI-enabled warfare. *J. Military Ethics* **21**(3–4), 246–271 (2022)
59. Khanal, S., Zhang, H., Taihagh, A.: Development of new generation of artificial intelligence in china: When Beijing's global ambitions meet local realities. *J. Contemp. China* **34**(151), 19–42 (2025)
60. Kim, P.: AI and inequality. In: *Forthcoming in the Cambridge Handbook on Artificial Intelligence & the Law*, Johnson, K., Reyes, C. (eds.) (2022). Washington University in St. Louis Legal Studies Research Paper (21-09), 03 (2021)
61. Kumar, S.: *Breaking privilege paradoxes: Ethical framework for AI advancement in an unequal world*. Engineering Archive. <https://doi.org/10.31224/3207>
62. Lab, E.I.: *Law enforcement and the challenge of deepfakes*. Technical Report, European Union (2022)

63. Laux, J., Wachter, S., Mittelstadt, B.: Three pathways for standardisation and ethical disclosure by default under the european union artificial intelligence act. *Comput. Law Security Rev.* **53**, 105957 (2024)
64. Lin, S., Wang, M., Jing, C., Zhang, S., Chen, J., Liu, R.: The influence of ai on the economic growth of different regions in china. *Sci. Rep.* **14**(1), 9169 (2024)
65. Madiega, T.A.: AI investment: EU and global indicators (At a Glance) (2024). [https://www.europarl.europa.eu/thinktank/en/document/EPRS_ATA\(2024\)760392](https://www.europarl.europa.eu/thinktank/en/document/EPRS_ATA(2024)760392)
66. Malec, L., Lešetický, J.: Social media content moderation, censorship and ai detection evasion techniques. In: *IDIMT-2024: Changes to ICT, Management, and Business Processes through AI* (2024)
67. Manchanda, J., Boettcher, L., Westphalen, M., Jasser, J.: The open source advantage in large language models (LLMs). *arXiv preprint arXiv:2412.12004* (2024)
68. Marx, K.: *konomisch-philosophische Manuskripte aus dem Jahre 1844*. Zenodot (2011)
69. McCarthy, J., Minsky, M.L., Rochester, N., Shannon, C.E.: A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Mag.* **27**(4), 12–12 (2006)
70. Mueller, B.: How much will the artificial intelligence act cost europe? Technical Report, Information Technology and Innovation Foundation (2021)
71. National Science and Technology Council (US). Select Committee on Artificial Intelligence: The National Artificial Intelligence Research and Development Strategic Plan: 2023 Update. National Science and Technology Council (US), Select Committee on Artificial Intelligence, 1–37 (2023)
72. Organizzazione delle Nazioni Unite.: *Trasformare il nostro mondo: l’agenda 2030 per lo sviluppo sostenibile*. Risoluzione adottata dall’Assemblea Generale il **25** (2015). <https://www.cisl.it/wp-content/uploads/2019/03/assemblea%20generale%20ONU%2025%20settembre%202015.pdf>
73. Opderbeck, D.W.: Copyright in ai training data: a human-centered approach. *Okla. L. Rev.* **76**, 951 (2023)
74. O’Meara, D.J.: Tracking the sources of the fragments of heraclitus in stobaeus’ anthology. *Heraklit im Kontext* **8**, 439 (2017)
75. Papishev, G., Yarime, M.: The limitation of ethics-based approaches to regulating artificial intelligence: regulatory gifting in the context of russia. *AI Soc.* **39**(3), 1381–1396 (2024)
76. Parliament, E.: United states approach to artificial intelligence. Technical Report, European Parliament (2024)
77. Pasquetto, I.V., Sands, A.E., Borgman, C.L.: Exploring openness in data and science: What is “open,” to whom, when, and why? *Proc. Assoc. Inf. Sci. Technol.* **52**(1), 1–2 (2015)
78. Perlmutter, S.: Copyright and artificial intelligence. part 2: Copyrightability. Technical Report, United States Copyright Office (2025)
79. Petrella, S., Miller, C., Cooper, B.: Russia’s artificial intelligence strategy: the role of state-owned firms. *Orbis* **65**(1), 75–100 (2021)
80. Picht, P.G., Richter, H.: Eu digital regulation 2022: data desiderata. *GRUR Int.* **71**(5), 395–402 (2022)
81. PwC: Sizing the prize: What’s the real value of AI for your business and how can you capitalise? PwC (2017)
82. Regulation, P.: Regulation (eu) 2016/679 of the european parliament and of the council. *Regulation (eu)* **679**, 2016 (2016)
83. Regulation, P.: Regulation (eu) 2022/1925 of the european parliament and of the council. *Regulation (eu)* **1925** (2022)
84. Regulation, P.: Regulation (eu) 2022/868 of the european parliament and of the council. *Regulation (eu)* **868** (2022)
85. Regulation, P.: Regulation (eu) 2024/1689 of the european parliament and of the council. *Regulation (eu)* **1689** (2024)
86. Rejmaniak, R.: Bias in artificial intelligence systems. *Białostockie Studia Prawnicze* **3**(26), 25–42 (2021)

87. Roberts, H., Cows, J., Morley, J., Taddeo, M., Wang, V., Floridi, L.: *The Chinese Approach to Artificial Intelligence: An Analysis of Policy, Ethics, and Regulation*. Springer, Berlin (2021)
88. Samuel, A.L.: Some moral and technical consequences of automation—a refutation. *Science* **132**(3429), 741–742 (1960)
89. Saveliev, A., Zhurenkov, D.: Artificial intelligence and social responsibility: the case of the artificial intelligence strategies in the United States, Russia, and China. *Kybernetes* **50**(3), 656–675. (2021)
90. Schuett, J.: Risk management in the artificial intelligence act. *Eur. J. Risk Regul.* **15**(2), 367–385 (2024)
91. Sheehan, M.: China’s ai regulations and how they get made. *Horizons J. Int. Relat. Sustain. Develop.* **24**, 108–125 (2023)
92. Shrestha, M., Manandhar, S., Shrestha, S.: Forecasting water demand under climate change using artificial neural network: a case study of Kathmandu Valley, Nepal. *Water Supply* **20**(5), 1823–1833 (2020)
93. Szakonyi, A., Chellasamy, H., Dawson, M.: Russia’s strategies for leveraging AI policies and investments for global economic competitiveness (2024). <https://aisel.aisnet.org/mwais2024/12>
94. Tocchetti, A., Corti, L., Balayn, A., Yurrita, M., Lippmann, P., Brambilla, M., Yang, J.: AI robustness: a human-centered perspective on technological challenges and opportunities. *ACM Comput. Surveys* **57**(6), 1–38 (2025)
95. Trade, E.U., Council, T.: AI for public good: EU-U.S. research alliance in ai for the public good. Technical Report, European Commission (2024)
96. Triguero, I., Molina, D., Poyatos, J., Del Ser, J., Herrera, F.: General purpose artificial intelligence systems (gpais): Properties, definition, taxonomy, societal implications and responsible governance. *Inf. Fusion* **103**, 102135 (2024)
97. Union, E., Economic, E., Committee, S.: Digital Services Act and Digital Markets Act—Stepping stones to a level playing field in Europe. European Economic and Social Committee (2021). <https://doi.org/doi/10.2864/28842>
98. Ünver, H.A.: Artificial intelligence (AI) and human rights: Using ai as a weapon of repression and its impact on human rights (2024)
99. Volkov, D., Ivanova, N., Morozov, P., Sokolova, O.: Ethical considerations in ai and machine learning: A framework for responsible development. *J. Mach. Learn. Healthcare Decis. Support* **4**(1), 36–59 (2024)
100. White, M., Haddad, I., Osborne, C., Yanglet, X.Y.L., Abdelmonsef, A., Varghese, S.: The model openness framework: Promoting completeness and openness for reproducibility, transparency, and usability in artificial intelligence. *arXiv preprint arXiv:2403.13784* (2024)
101. Wiener, N.: Some moral and technical consequences of automation: As machines learn they may develop unforeseen strategies at rates that baffle their programmers. *Science* **131**(3410), 1355–1358 (1960)
102. Zhang, Y., Han, W.: In depth: China’s burgeoning social credit system stirs controversy. *CAIXIN* (2019)
103. Zuboff, S.: *Il capitalismo della sorveglianza: il futuro dell’umanità nell’era dei nuovi poteri*. Luiss University Press, Roma (2019)

Chapter 11

Epilogue



Massimo Mecella 

For many years, information systems engineering has been developed as discipline of design. Principled approaches are investigated and applied in order to develop artefacts; refinements from abstract (conceptual) till concrete (executable/runnable) artefacts are carried out in order to achieve a system. Different methodologies have been proposed over the years, starting from older *waterfall*, later *iterative*, and nowadays *agile* approaches. But independently on the way of scheduling and sequencing activities and tasks, (practically) the quality of the designer (or of the designer team) heavily impacts the overall quality of the system. Quality of an information system is a well-investigated topic, with many facets (including the quality of the information managed by the system, the quality of the software constituting the system, the quality of the processes carried out through the system, etc.); undoubtedly, it is agreeable that the quality of the persons (designers) modeling and designing and building the system is a primary concern.

Methodologies and practices investigated and proposed over the years support the designer (team), but mostly knowledge, skills, and capabilities of the designers are in their minds. We refer to this approach as *heavily design-based approach*, in order to emphasize that the design activities are crucial.

With generative AI and LLMs, we can conversely attempt a different approach: a designer can be assisted by a *smart twin*,¹ which can iteratively support his/her

¹ The reader should not mislead this concept with the one of *digital twin*, which is a digital model of an intended or actual real-world physical product, system, or process (a physical twin) that serves as a digital counterpart of it for purposes such as simulation, integration, testing, monitoring, and maintenance [1–4].

M. Mecella (✉)
Sapienza Università di Roma, Rome, Italy
e-mail: mecella@diag.uniroma1.it

work. A precise design is no more crucial. The designer can quickly try and iterate the production of the artefact he/she is in charge of, being supported by the assistant.

In analogy with low-code approaches to software development, we define this approach \ll low design \gg . Still far away from there, but initial symptoms are here (see the survey and our research).

This transformation in IS engineering presents both opportunities and challenges:

- Faster. Adopting smart twins makes the design cycle faster.
- More standardized. Smart twins will be trained on the best possible practices, and even if customized (e.g., through RAG techniques) to different needs and development contexts, there will be a positive standardization effect on outcomes and practices.
- Potential risks for innovation and disruptive ideas. The trend toward standardization that potentially can be induced by adopting smart twins is a risk in terms of innovation and opportunities for disruptive ideas. In a design approach in which all designers adopt similar assistants, there is the risk of limiting creativity of individuals. Appropriate countermeasures, yet to be devised, should be planned in order to avoid this.
- Dependency on proprietary AI models (depending on which LLMs are used). This is quite obvious; designer will adopt proprietary models.
- Openness of AI models as a key concern. It is related to the above point; governance of AI/LLMs should push toward open models and transparent datasets (cf. Chap. 10), even if this clearly contrasts with possible business models, based on fees, which can justify investments from vendors and providers.
- Higher costs and environmental impact (cf. the footprint of LLMs and their costs when used via paid APIs). Companies adopting smart twins/assistants should evaluate the business model, for engineering the information system, that can afford, if any, such costs and impact.

Our claim is that LLMs can become, and indeed are already partly becoming, a powerful assistant for designers. However, a human-in-the-loop approach remains essential. Moreover, openness is a crucial factor in ensuring transparency, accessibility, and ethical development.

We would conclude with a few sentences from Bertrand Russell:

\ll The fundamental cause of the trouble is that in the modern world the stupid are cocksure while the intelligent are full of doubt. \gg

to mean that clearly this approach based on smart twins currently raises many doubts related to correctness, effectiveness, and, in general, meaningfulness, but also

\ll One should never be certain of anything, because nothing deserves certainty, and so one should always maintain an element of doubt in one's beliefs, and one should be able to act vigorously in spite of the doubt. \gg

to mean that nevertheless we cannot simply refuse it and not act and that, conversely, information systems engineers and researchers should act vigorously, in order to drive and correct this trend up to the point of making it meaningful, as it will happen independently.

References

1. Moi, T., Cibicik, A., Rølvåg, T.: Digital twin based condition monitoring of a knuckle boom crane: An experimental study. *Eng. Failure Anal.* **112**, 104517 (2020). <http://doi.org/10.1016/j.engfailanal.2020.104517>. hdl:11250/2650461. ISSN 1350-6307
2. Haag, S., Anderl, R.: Digital twin – Proof of concept. *Manuf. Lett. Ind. 4.0 Smart Manuf.* **15**, 64–66 (2018). <http://doi.org/10.1016/j.mfglet.2018.02.006>. ISSN 2213-8463.
3. Boschert, S., Rosen, R.: Digital Twin—The Simulation Aspect. In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*, pp. 59–74. Springer International Publishing (2016). http://doi.org/10.1007/978-3-319-32156-1_5
4. Negri, E.: A review of the roles of Digital Twin in CPS-based production systems. *Procedia Manuf.* **11**, 939–948 (2017). <http://doi.org/10.1016/j.promfg.2017.07.198>