RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

Departement du Genie Industriel

End-of- Study Project Dissertation for Obtaining State Engineer's Degree in Industrial engineering
Option : Data Science and Artificial Intelligence

Creation of a chatbot assistant for improving Client service

Chebouti Boutheina

jury Composition:

President:	Mr Hakim Fourar Laidi	ENP MCA
Examiner:	Mr Zouaghi Iskandar	ENP MCA

Supervisors :

ENP supervisor:	Mr Arki Oussama	ENP MCA
Company supervsisor:	Mr Karel Bourgois	Voxist

ENP 2024

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

Departement du Genie Industriel

End-of- Study Project Dissertation for Obtaining State Engineer's Degree in Industrial engineering
Option : Data Science and Artificial Intelligence

Creation of a chatbot assistant for improving Client service

Chebouti Boutheina

jury Composition:

President:	Mr Hakim Fourar Laidi	ENP MCA
Examiner:	Mr Zouaghi Iskandar	ENP MCA

Supervisors :

ENP supervisor:	Mr Arki Oussama	ENP MCA
Company supervsisor:	Mr Karel Bourgois	Voxist

ENP 2024

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

Departement du Genie Industriel

Mémoire de Projet de Fin d'Études En vue de l'obtention du diplôme d'Ingénieur d'État en Génie Industriel Option : Data Science et Intelligence Artificielle

Creation d'un Chatbot assistant pour l amelioration du service client

Chebouti Boutheina

Composition du jury:

President:	Mr Hakim Fourar Laidi	ENP MCA
Examinateur:	Mr Zouaghi Iskandar	ENP MCA

Encadrant :

Encadrant de l'ENP:	Mr Arki Oussama	ENP MCA
Encadrant de l'enreprise:	Mr Karel Bourgois	Voxist

ENP 2024

ملخص

التطور السريع في تقنيات خدمة العملاء يخلق تحديات وفرصًا، لا سيما في استخدام الذكاء الاصطناعي لتحسين جودة وكفاءة التفاعل. تستعرض هذه الأطروحة تطوير وتنفيذ نظام دردشة متطور يعمل تعدد الوكلاء في الشركة تحت عنوان "بناء مساعد لتعزيز خدمات العملاء". يهدف هذا العمل إلى التغلب على القيود التي تواجهها النماذج الكبيرة للغة مثل محدودية الوصول إلى مصادر البيانات الخاصة ونقص التحديثات الفورية والقدرات التحليلية المحدودة.

يعتمد الحل المقترح على نظام منظم يقوده وكيل يدير التفاعل بين الوكلاء المتخصصين المتعددين، كل منهم مصمم لأداء وظائف محددة بالاعتماد على الجمع بين التوليد واسترجاع المعلومات في ال (RAG) هذا يسمح بإدارة ديناميكية للبيانات وتنفيذ معقد للوظائف، مع الاستفادة من تقنية الاسترجاع المعزز . تحسين استجابات الدردشة والوصول المحسّن إلى المعلومات، مما يعزز قدرة ال على التعامل مع البيانات الحساسة وتنفيذ الأوامر بفعالية.

النتائج الرئيسية تُظهر أن النظام متعدد الوكلاء يتعامل بكفاءة مع القيود المفروضة على التقليدية، يحسّن الكفاءة التشغيلية، ويضمن تفاعلات أكثر تخصيصًا ووعيًا بالسياق، وهو أمر حيوي للحفاظ على ثقة ورضا العملاء. في الختام، يمثل هذا النظام في نقلة نوعية في تعزيز خدمات العملاء من خلال دمج تقنيات متقدمة وقدرات تحليلية محسنة، ويقدم حلولاً قابلة للتطوير تتلاءم مع التطور المستمر لاحتياجات العمل وتوقعات العملاء
الكلمات المفتاحية : نظام دردشة ,النماذج الكبيرة للغة, وكيل, الذكاء الاصطناعي,الجمع بين التوليد واسترجاع المعلومات.

Resumé

L'évolution rapide des technologies de service client présente à la fois des défis et des opportunités, en particulier pour exploiter l'intelligence artificielle afin d'améliorer la qualité et l'efficacité des interactions. Cette thèse présente le développement et la mise en œuvre d'un système de chatbot multi-agent sophistiqué conçu pour améliorer les services clients chez Voxist. Intitulé "Construction d'un assistant chatbot pour améliorer les services clients", ce travail se concentre sur le dépassement des limites inhérentes aux modèles de langage de grande taille (LLM), telles que l'incapacité à accéder à des sources de données privées, le manque de mises à jour en temps réel et les capacités de raisonnement limitées.

Le cœur de la solution proposée repose sur un système multi-agent structuré, centré autour d'un Agent Meta qui orchestre les interactions entre divers sous-agents spécialisés. Chaque sous-agent est conçu pour des rôles spécifiques, permettant des interactions dynamiques, une gestion des données en temps réel et l'exécution de fonctions complexes. Le système utilise la génération augmentée par récupération (RAG) pour améliorer la réactivité du chatbot et son accès à des informations mises à jour, améliorant considérablement la capacité du chatbot à gérer les données sensibles, exécuter des commandes et effectuer des tâches opérationnelles de manière efficace. De plus, en sollicitant efficacement le LLM, le système améliore ses capacités de raisonnement, permettant des réponses plus précises et contextualisées.

Les résultats clés de cette mise en œuvre indiquent que le système multi-agent traite efficacement les limites des LLM traditionnels en facilitant l'accès sécurisé aux bases de données privées, en permettant des mises à jour en temps réel et en améliorant les capacités de raisonnement. Le système améliore non seulement l'efficacité opérationnelle, mais garantit également que les interactions sont personnalisées et contextualisées, ce qui est crucial pour maintenir la confiance et la satisfaction des clients.

En conclusion, la mise en œuvre de ce système multi-agent chez Voxist représente une étape transformative dans l'amélioration des services clients. En intégrant des technologies avancées, des rôles d'agents stratégiques et des capacités de raisonnement améliorées, le système offre des solutions évolutives qui s'adaptent aux besoins en constante évolution des affaires et aux attentes des clients, mettant en valeur les capacités robustes des systèmes de chatbots pilotés par l'IA.
Mots Clés: LLMS,Chatbot,RAG,Agents,Artificial intelligent,

abstract

The rapid evolution of client service technologies poses both challenges and opportunities, particularly in harnessing artificial intelligence to enhance interaction quality and efficiency. This thesis presents the development and implementation of a sophisticated multi-agent chatbot system designed to improve client services at Voxist. Titled "Building a Chatbot Assistant to Enhance Client Services," this work focuses on overcoming the inherent limitations of large language models (LLMs), such as the inability to access private data sources, lack of real-time updates, and limited reasoning capabilities.

The core of the proposed solution involves a structured multi-agent system centered around a Meta Agent that orchestrates interactions among various specialized sub-agents. Each sub-agent is tailored to specific roles, enabling dynamic interactions, real-time data management, and the execution of complex functions. The system leverages Retrieval-Augmented Generation (RAG) to enhance the chatbot's responsiveness and access to updated information, significantly improving the chatbot's ability to handle sensitive data, execute commands, and perform operational tasks efficiently. Additionally, by effectively prompting the LLM, the system enhances its reasoning capabilities, enabling more accurate and contextually aware responses.

Key outcomes from this implementation indicate that the multi-agent system effectively addresses the limitations of traditional LLMs by facilitating secure access to private databases, enabling real-time updates, and enhancing reasoning abilities. The system not only improves operational efficiency but also ensures that interactions are personalized and contextually aware, which are critical for maintaining client trust and satisfaction.

In conclusion, the implementation of this multi-agent system at Voxist represents a transformative step in enhancing client services. By integrating advanced technologies, strategic agent roles, and enhanced reasoning capabilities, the system offers scalable solutions that adapt to evolving business needs and client expectations, showcasing the robust capabilities of AI-driven chatbot systems.
Keywords: LLMS,Chatbot,RAG,Agents,Artificial intelligent,

Acknowledgements

I would like to express my profound gratitude to my academic advisor and instructor, Mr. Arki Oussama. His consistent support and enlightening teachings have played a crucial role in shaping my academic journey. The knowledge and guidance he provided have been indispensable.

I am immensely thankful to my supervisors at Voxist , Karel Bourgois and Moumene Boumadane, for their expert guidance and steadfast support throughout my thesis work. Their insights and expertise have been crucial in the successful completion of this project.

Special thanks also go to Mr. Abdelwehab Heba, whose pivotal role in securing this internship was instrumental in providing me with an exceptional platform to conduct my research. His efforts have significantly enriched my professional experience.

Additionally, I extend my appreciation to all the professors involved in my academic program. Their dedication to imparting knowledge and their commitment to fostering student success have not only facilitated my academic growth but have also been a source of inspiration throughout my studies.

Moreover, I would like to acknowledge all individuals who directly or indirectly contributed to the completion of this thesis. Your support and encouragement have been invaluable and deeply appreciated.

Thank you all for your belief in me, your encouragement, and your support.

Chebouti Boutheina

Dedication

To my beloved family—my parents, whose endless love and sacrifice have shaped the person I am today; my inspiring little sister, whose dreams propel me forward; and my dear grandmother, whose wisdom and gentle presence have been my sanctuary. You are my heart's foundation and my unwavering light.

I also dedicate this to the soul-deep friendships that have blossomed at Polytech. To my cherished soul sisters—Nesrine, Soundous, and Khawla: in every shared silence and burst of laughter, you have been my confidants and my joy. Our friendship is a beautiful tapestry we continue to weave together, filled with vibrant threads of our memories and dreams.

To my "data girls"—Amira, Samah, Sihem, and Yousra: with every challenge we faced and every triumph we celebrated, you were there. Your support has been a gift beyond measure, and my gratitude is as boundless as our shared aspirations. May our friendship continue to light our paths and warm our hearts.

To my steadfast companions—Abdelhak, Ouanis, Hamza, and Rayane: thank you for standing by my side through thick and thin. Your brotherhood has fortified me, teaching me the true meaning of friendship. Each moment we have shared is a cherished stone in the mosaic of my life.

And to everyone at Polytech: each of you has touched my life, weaving your unique threads into the fabric of my days. You have all painted my world with broader strokes of kindness, laughter, and insight.

Thank you for filling my journey with love, for inspiring me daily, and for believing in me always.

Contents

List of Tables

List of Figures

Acronymes

NLP	Natural Language Processing
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
ITSM	Information Technology Service Management
ML	Machine Learning
DL	Deep Learning
API	Application Programming Interface
BOW	Bag of Words
BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
LLAMA	Large Language Model Meta AI
CPU	Central Processing Unit
GPU	Graphics Processing Unit
IBM	International Business Machines
ASR	Automatic Speech Recognition
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
AGI	Artificial General Intelligenc

General introduction

Context

In the rapidly evolving landscape of digital technologies, businesses are increasingly leveraging artificial intelligence to enhance operational efficiency and customer interactions. Large Language Models (LLMs) have emerged as powerful tools in natural language processing, offering sophisticated solutions for automating and enhancing communication. However, their integration into practical applications presents significant challenges, particularly in dynamic and sensitive environments.

Problem Statement

Given that Voxist's work is fundamentally based on LLMs, it fully acknowledges the limitations these models encounter, such as handling real-time data, managing sensitive information securely, and maintaining context over interactions. These challenges limit the practical utility of LLMs in delivering personalized, accurate, and timely responses. Additionally, integrating LLMs into existing systems presents substantial technical and operational hurdles. Voxist is dedicated to overcoming these challenges by enhancing the technology to securely manage real-time data and maintain context over interactions. We aim to simplify the integration of LLMs into existing systems through the development of robust frameworks, enabling their full potential for more personalized and accurate responses.

Objectives

This thesis aims to develop a multi-agent chatbot system that effectively integrates LLMs to address these inherent limitations at voxist . The proposed system is designed to enhance customer service operations by improving interaction quality, response accuracy, and operational efficiency. The focus is on creating a scalable and adaptable solution that can manage complex user interactions and sensitive data securely.

Outline of the Chapters

The thesis is organized into the following chapters, grouped into two main sections:

State of Play (Chapters 1 and 2)

- **Chapters 1**: This chapter offers a comprehensive presentation of the company, emphasizing its range of products, core missions, and strategic objectives. It provides insights into the company's values, market positioning, and long-term goals, setting the stage for understanding its overall business philosophy.
- **Chapters 2**: Provides an overview of the company's application of LLMs, highlighting the general challenges, the practical limitations in chatbot design, and the strategies to mitigate these issues.

State of the Art (Chapters 3 to 5)

- **Chapter 3** : Explores foundational concepts in machine learning and deep learning, their relationship with NLP, and the development of LLMs.
- **Chapters 4 and 5** : Discusses the advent and evolution of LLMs, the challenges they face it also highlights the use of vectore databases , and the techniques like prompt engineering and RAG that enhance their application in real-world scenarios.

Proposed Solution (Chapter 6)

- **Chapter 6**: Details the proposed multi-agent system's design, describing each component and its role in enhancing client services it also Covers the practical implementation of the solution, including the development of advanced retrieval methods and the integration of user context agents and command execution tools.

This introduction sets the stage for a thorough exploration of both the theoretical background and the practical application of advanced AI technologies to enhance customer service, focusing on the implementation and evaluation to validate the effectiveness of the proposed solutions.

Part I

State Of play

Chapter 1

Company Presentation

1.1 Introduction

Established in 2016 and based in Paris, Voxist has emerged as a leader in the fields of voice recognition and natural language understanding. The company leverages artificial intelligence to transform spoken data into actionable insights, offering services that include call transcription, analysis, and the development of voice assistants tailored to sectors such as hospitality, healthcare, and IT. With strategic partnerships in both France and the United States, Voxist excels in developing advanced ASR (Automatic Speech Recognition) models, employing sophisticated NLP (Natural Language Processing) techniques, and utilizing comprehensive machine learning algorithms. Its commitment to ongoing innovation is evident in its adoption of cutting-edge technologies for enhancing human language comprehension. Voxist's growing team is dedicated to research, development, and marketing, positioning the company as a prominent and influential player in the telecommunications industry.

1.2 Technical Overview

- **Business Name:** Voxist
- **Headquarters:** Paris, Île-de-France
- **Sector:** Specializes in voice recognition and natural language understanding within the telecommunications industry.
- **Website:** http://www.voxist.com

1.3 Terms and Jargon

Term	**Definition**
NLP (Natural Language Processing)	The branch of AI that enables computers to understand and process human languages, bridging the gap between human communication and machine understanding.
Voice Recognition	Technology that allows computers and devices to interpret spoken commands or dictation.
API (Application Programming Interface)	Rules and specifications that allow software programs to communicate and interact with each other.
Endpoint	A specific URL within a web API where resources or services are accessed, such as functions for parsing or sentiment analysis in NLP.
Deployment	The process of implementing a software solution in a production environment to make it available to users or other systems.
Framework	Software foundation used to develop programs; important in NLP for providing libraries and tools for model building and training.
Azure OpenAI	Microsoft's cloud service offering access to OpenAI's advanced models, facilitating the integration of AI capabilities into business applications.
Retrieval-Augmented Generation (RAG)	Combines retrieval of data with a generative model to enhance response generation or task performance in NLP.
Information Extraction	Automatically pulling structured information like entities and relationships from unstructured text.
Large Language Models (LLMs)	AI models trained on extensive text data, capable of performing a broad range of language tasks.

Table 1.1: Key Terms in NLP and Voice Recognition Technologies

1.4 Mission and Objectives

Voxist is dedicated to transforming voice data into valuable information through cutting-edge voice AI technologies, with services including voice transcription and call analysis. The company aims to improve how businesses engage with voice data.

1. **Product Innovation:** Innovate and develop new solutions like intelligent voicemail and voice recognition APIs.
2. **Market Expansion:** Broaden market presence in France and the USA and form strategic partnerships to enhance distribution.

3. **Competitive Edge:** Strengthen market position by offering competitive solutions with low error rates and fast response times.
4. **Team Expansion:** Grow the workforce, particularly in R&D and marketing, to support ongoing innovation and business needs.
5. **Technological Advancement:** Advance in technologies such as real-time ASR-LLM coupling and knowledge management to maintain industry leadership.

1.5 Innovative Products and Partnerships

Voxist is known for its groundbreaking products and ongoing development efforts, including:

1. **Intelligent Visual Voicemail:** Increasingly popular with 30,000 monthly users and expanding through partnerships like that with Cisco.
2. **Voice Recognition API:** Developed in collaboration with OVH, this API is set to revolutionize voice interaction capabilities in the marketplace.
3. **Medical Report Automation:** A service that converts voice recordings into structured medical reports, aiming for a production of 100,000 reports monthly.
4. **Customized Voice Assistants:** Developing sector-specific assistants for enhanced client service in areas like hospitality and tech support.

1.6 Core Values

1. **Innovation:** A relentless pursuit of next-generation technologies in voice recognition and language processing.
2. **Quality:** Commitment to high standards through strategic partnerships and specialized product development.
3. **Adaptability:** Flexibility in providing sector-specific solutions to meet diverse client needs.
4. **Collaboration:** Valuing partnerships with industry leaders to enhance technological and market reach.
5. **Technological Leadership:** A focus on advancing in the realm of AI, ASR, and LLM technologies.

1.7 Partners and Clients

Key collaborators include , The Voice Lab, LIA, Huma-Num, Hub France AI.

Figure 1.1: The Voice Lab

Figure 1.2: Humanum

Figure 1.3: Hub France AI

Figure 1.4: Laboratoire INformatique d'Avignon

1.8 Conclusion

Voxist stands out in the telecommunications sector through its innovative integration of ASR and NLP technologies, enhancing voice data usability across multiple industries. The company's strategic expansions and partnerships, particularly in the US and France, highlight its commitment to innovation and market leadership. Through continuous technological advancements and a focus on sector-specific solutions, Voxist is poised to maintain its influential status in the evolving landscape of voice AI.

Chapter 2

Problem description

2.1 Voxist innovative missions

Voxist has adeptly integrated Large Language Models (LLMs) into its suite of services, leveraging the cutting-edge capabilities of AI to transform spoken data into actionable insights with precision and efficiency.

2.1.1 Information Extraction

At the core of Voxist's service offerings is the use of LLMs for information extraction. This process involves the automatic identification and categorization of key information from spoken interactions. Important data such as names, dates, locations, customer sentiments, and intents are extracted from unstructured voice data, making it structured and more analyzable. This capability is essential for creating meaningful summaries and actionable insights from customer interactions across various industries.

2.1.2 Model Fine-Tuning

To ensure that the LLMs are optimally aligned with specific client needs, Voxist engages in fine-tuning these models on sector-specific data. For instance, LLMs are trained on specialized datasets like medical transcripts for healthcare clients or customer service logs in the hospitality sector. Fine-tuning enhances the models' understanding of industry-specific terminologies and nuances, significantly improving their practical efficacy in targeted applications.

2.1.3 Custom Voice Assistant Development

Voxist also uses LLMs to develop customized voice assistants tailored to different industrial needs. These voice assistants are designed to understand complex queries, perform tasks, and engage in natural dialogues. Thanks to the learning capabilities of LLMs, these assistants continually adapt and evolve based on user interactions, making them more intuitive and user-friendly over time.

2.1.4 Sentiment Analysis

LLMs at Voxist are instrumental in performing sentiment analysis on transcribed calls. This allows the company to detect and analyze the emotional undertones of voice interactions, providing clients with deep insights into customer satisfaction and feedback. This analysis helps businesses understand their customers better and refine their services accordingly.

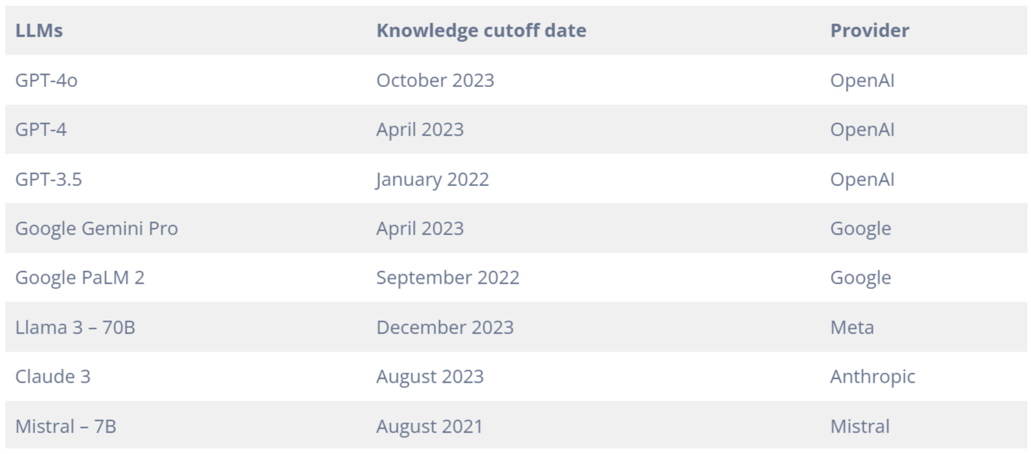
By integrating these advanced AI technologies, Voxist not only enhances its existing capabilities but also sets the stage for continued innovation in the telecommunications industry, thereby solidifying its position as a market leader.

2.2 Challenges AND Limitations Associated with Large Language Models at Voxist

Knowledge Cutoff in Large Language Models

we can delve deeper into the implications of the knowledge cutoff in large language models (LLMs). This knowledge cutoff point is a pivotal concept that significantly influences the performance and reliability of LLMs like ChatGPT, Google Bard, and Microsoft Bing.

Definition and Significance The knowledge cutoff point in LLMs is analogous to the publication date of a textbook. It represents the most recent point in time up to which the information used to train an LLM is current. Beyond this date, the LLM lacks awareness of new developments, leading to potential gaps in knowledge that can affect the accuracy and relevance of its responses.



LLMs	Knowledge cutoff date	Provider
GPT-4o	October 2023	OpenAI
GPT-4	April 2023	OpenAI
GPT-3.5	January 2022	OpenAI
Google Gemini Pro	April 2023	Google
Google PaLM 2	September 2022	Google
Llama 3 – 70B	December 2023	Meta
Claude 3	August 2023	Anthropic
Mistral – 7B	August 2021	Mistral

Figure 2.1: cut off dates for some LLMs [1]

- **Transparency and Reliability:** Transparency about the knowledge cutoff date is crucial for users to assess the reliability of the AI-generated content. Without clear communication of this cutoff, users may inadvertently rely on outdated or inaccurate information.

Challenges Posed by Ambiguous Knowledge Cutoff Dates The lack of clarity about the knowledge cutoff dates in various LLMs, as explored in the episode, can lead to inconsistencies and frustration for users:

- **Inconsistencies in Responses:** Different AI models like ChatGPT, Bing, and Google Bard may have varying cutoff dates, which can lead to discrepancies in the information they provide. This inconsistency can be confusing and diminish the user's trust in the model's utility.
- **Frustration from Lack of Transparency:** As highlighted during the tests conducted by Jordan, ambiguous or undisclosed knowledge cutoff dates complicate the user's ability to gauge the freshness and relevance of the information provided by the LLM. This lack of transparency can hinder effective decision-making based on AI consultations.

Impact on User Experience and Decision Making Understanding the knowledge cutoff is essential for users, especially business owners and decision-makers who rely on LLMs for timely and accurate information:

- **Business Decisions:** For business applications, where decisions often rely on the latest information, knowing the cutoff date helps in determining the utility of the LLM. If the model's training data is not up-to-date, it may not provide the best advice or insights, particularly in fast-evolving fields.
- **Ethical Implications:** Relying on AI for information without awareness of its knowledge limitations carries ethical implications, particularly if decisions based on this information have real-world consequences.

The discussion underscores the importance of transparency about knowledge cutoffs in LLMs. Users need to be aware of the limitations imposed by this cutoff to effectively utilize AI technologies without overestimating their capabilities. As LLMs continue to evolve, ongoing communication from developers about these cutoff dates will be critical in maintaining trust and ensuring that these powerful tools are used responsibly and effectively.

This enhanced focus on the knowledge cutoff point illuminates how critical it is for users to understand and consider this aspect when engaging with LLMs, ensuring that they maintain realistic expectations about the information provided by these models.

Understanding and Mitigating Hallucinations in Large Language Models

Large Language Models (LLMs) like ChatGPT and Bing Chat are known for their ability to generate fluent and coherent text across a variety of topics. However, these models are also prone to producing hallucinations—outputs that deviate from facts or logical consistency. This section explores the nature of these hallucinations, their causes, and strategies to minimize their occurrence.

What Are Hallucinations? Hallucinations in LLMs can range from minor inconsistencies to outright fabricated statements. They are categorized based on their granularity from sentence contradictions, where the model generates contradicting statements, to factual errors, where the output contains incorrect information.

Types of Hallucinations

- **Sentence Contradiction:** The model generates sentences that contradict each other within the same context.
- **Prompt Contradiction:** Outputs contradict the initial prompt provided to the model.
- **Factual Errors:** The model asserts incorrect facts as truths.
- **Nonsensical Information:** Irrelevant or out-of-place information is included in the model's response.

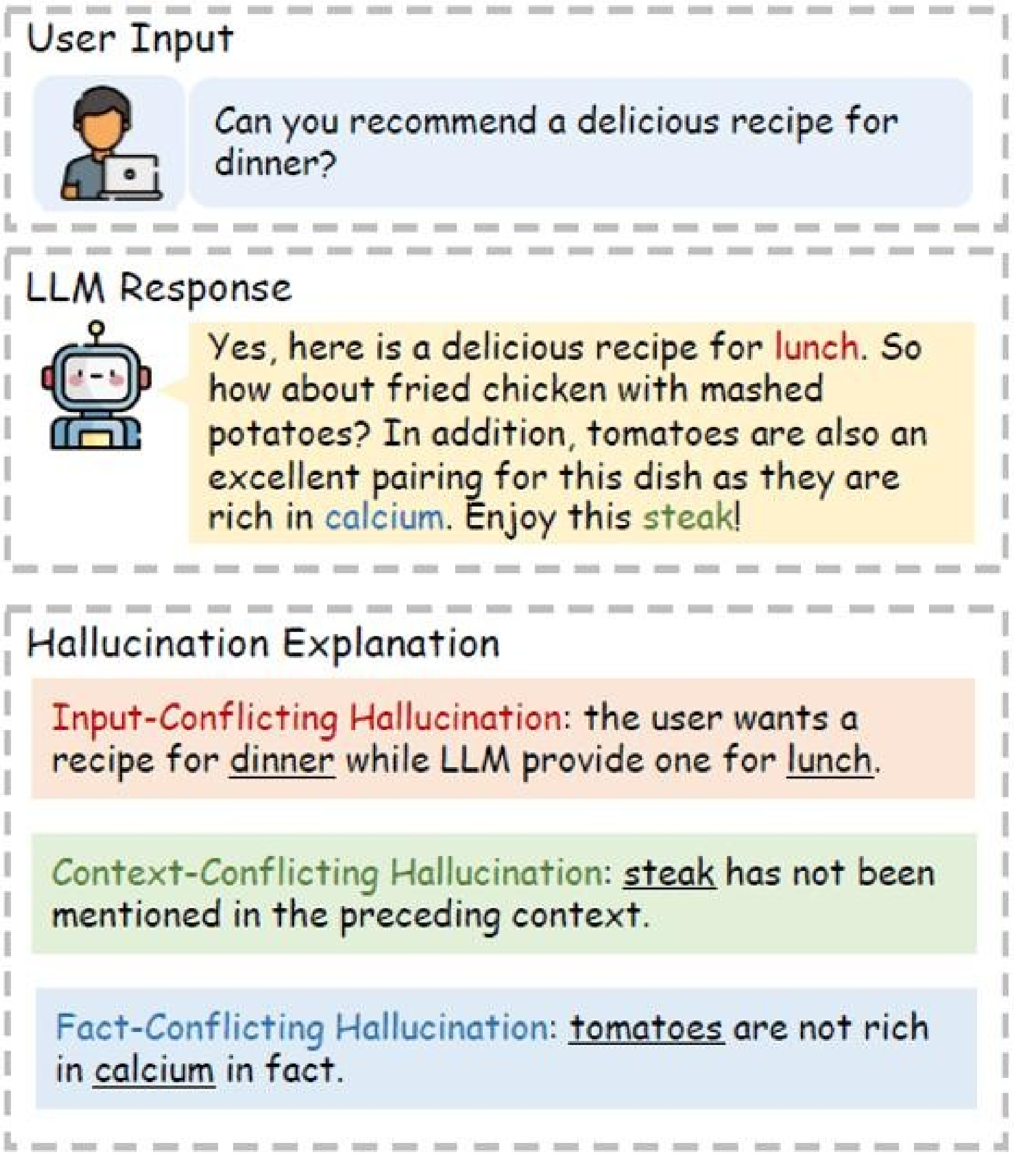

User Input
Can you recommend a delicious recipe for dinner?
LLM Response
Yes, here is a delicious recipe for lunch. So how about fried chicken with mashed potatoes? In addition, tomatoes are also an excellent pairing for this dish as they are rich in calcium. Enjoy this steak!
Hallucination Explanation
Input-Conflicting Hallucination: the user wants a recipe for dinner while LLM provide one for lunch.
Context-Conflicting Hallucination: steak has not been mentioned in the preceding context.
Fact-Conflicting Hallucination: tomatoes are not rich in calcium in fact.

Figure 2.2: Hallucination illustration [2]

Why Do Hallucinations Occur? The generation of hallucinations by LLMs is influenced by several factors including the quality of training data and the input context provided to the model.

Common Causes of Hallucinations

- **Data Quality:** LLMs trained on data with errors, biases, or inconsistencies are more likely to hallucinate.

- **Input Context:** Poorly defined prompts or contradictory information can mislead the model, resulting in inaccurate outputs.

While LLMs offer significant potential, their tendency to produce hallucinations can be a drawback. By understanding the underlying causes and employing effective mitigation strategies, users can enhance the reliability of these models. Through such measures, we harness the full capabilities of LLMs while minimizing the risks associated with their hallucinatory outputs.

Limitations in Reasoning Abilities of Large Language Models

Large Language Models, while impressive in their linguistic capabilities, often exhibit significant limitations when faced with tasks requiring advanced reasoning. This section explores these limitations, the reasons behind them, and the impact they have on the usability and reliability of LLM outputs.

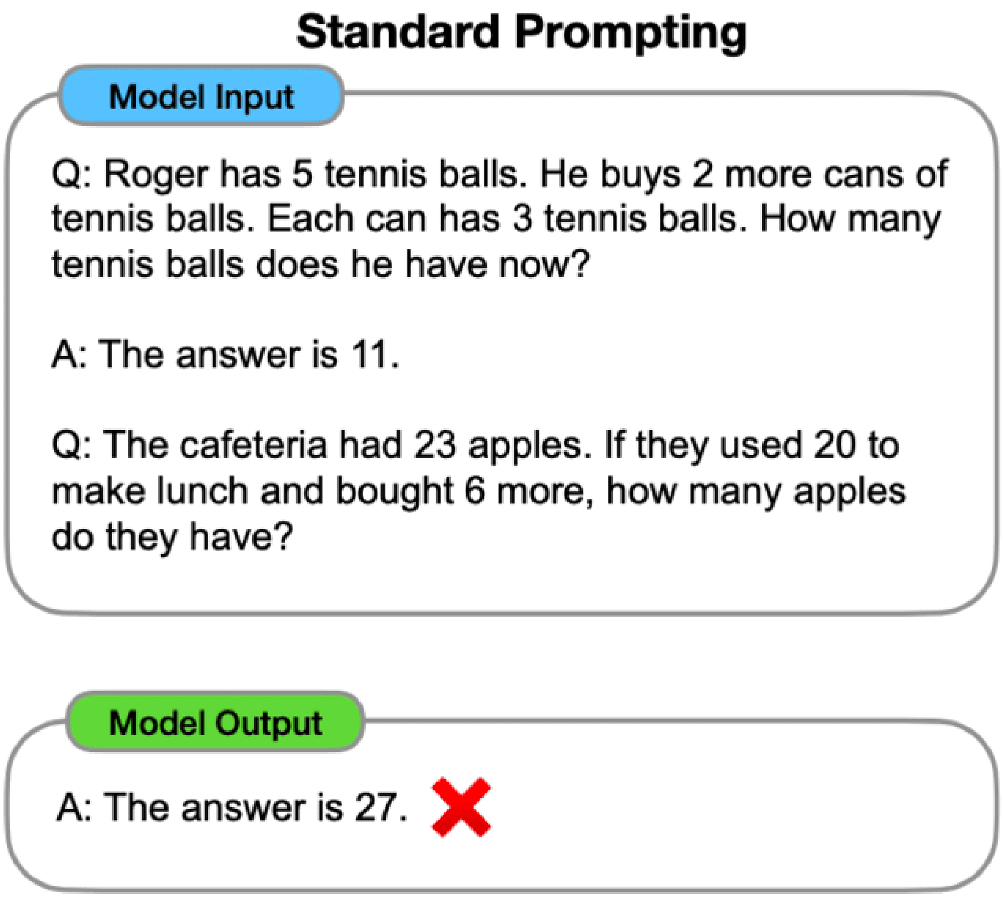

Standard Prompting
Model Input
Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?
Model Output
A: The answer is 27.

Figure 2.3: An illustration of reasoning limitations in math problems [3]

Nature of Reasoning Limitations

LLMs are fundamentally statistical models that generate text based on patterns learned from vast datasets. However, their ability to "reason" in the human sense is inherently limited.

Examples of Reasoning Failures

- **Complex Problem Solving:** LLMs struggle with problems that require logical deductions or strategic thinking, often failing to maintain a coherent strategy throughout an extended line of reasoning.
- **Multi-Step Reasoning:** Tasks that require considering multiple steps or layers of logic can confuse LLMs, leading to contradictory or incomplete conclusions.
- **Deep Contextual Understanding:** LLMs may miss nuances that require an understanding of deeper, implicit meanings, such as sarcasm, metaphorical language, or cultural context.

Causes of Reasoning Limitations The underlying architecture and training processes of LLMs contribute significantly to their reasoning limitations.

Key Factors Contributing to Limitations

- **Training Data Surface-Level Learning:** LLMs primarily learn from the surface structure of data without truly understanding underlying concepts or relationships. They replicate patterns rather than engage in deductive thinking.
- **Lack of World Knowledge:** Unlike humans, LLMs do not possess real-world understanding or the ability to draw on personal empirical experiences, which are often crucial for sophisticated reasoning.
- **Model Architecture Constraints:** The architectures of current LLMs, while effective for pattern recognition and text generation, do not replicate the neural processes associated with human reasoning and problem-solving.

Impact on Practical Applications The reasoning limitations of LLMs restrict their effectiveness in scenarios that require precision and reliability, impacting their deployment in critical areas.

Consequences for User Reliance

- **Misinformation and Errors:** Inaccurate or illogical outputs can lead to misinformation if not carefully reviewed.
- **Reliability in Critical Applications:** For applications like legal advice, medical diagnosis, or financial planning, where reasoning accuracy is paramount, the current generation of LLMs may fall short.
- **User Trust:** Frequent errors or overt limitations in reasoning can erode trust in the capabilities of AI systems among users.

Understanding the intrinsic limitations in the reasoning capabilities of LLMs is crucial for developers, users, and regulators. By acknowledging these limitations, stakeholders can better manage expectations, implement appropriate safeguards, and focus on continuous improvement in AI technologies.

Inability to Access or Process Private Data in Large Language Models

Nature of the Limitation Large Language Models operate on fixed datasets comprised of publicly available information and lack the capability to integrate or retrieve private, personalized, or real-time data during their operations. This inherent limitation significantly restricts their functionality in environments where access to confidential or personalized information is critical.

Examples of Practical Implications

- **Customer Service:** Without access to individual customer histories or specific account details, LLMs are unable to provide effective personalized support, often resulting in generic responses that may not address specific user issues.
- **Healthcare Advice:** LLMs' inability to consult personal medical records or patient histories prevents them from offering accurate medical recommendations, making them unreliable for personal health inquiries or medical advice.
- **Personalized Learning:** The effectiveness of LLMs in educational settings is hampered as they cannot tailor learning experiences based on individual student profiles, progress reports, or learning preferences.

Causes of the Limitation The design and operational framework of LLMs prioritize user privacy and data security, intentionally restricting their access to private databases to prevent misuse and ensure compliance with stringent data protection regulations.

Key Factors Contributing to the Limitation

- **Privacy and Security Protocols:** Stringent data privacy laws such as the GDPR necessitate these restrictions, as non-compliance can lead to severe penalties. Thus, LLMs are deliberately designed not to access or store private information.
- **Training Dataset Constraints:** The general and anonymized nature of the data used to train LLMs means they are not equipped to handle or interpret sensitive or personalized data, which limits their applicability in scenarios requiring customized responses.

Impact on Usability and Trust The limitations in handling private data directly influence the trust users place in LLMs and their applicability across various industries, particularly those requiring a high degree of personalization and confidentiality.

Consequences for Deployment

- **Limited Customization:** LLMs provide outputs that are broad and non-specific, which may not meet the needs of users seeking tailored information or actions based on their personal data.
- **Ethical Concerns:** There is an increased risk of generating inaccurate or irrelevant outputs, which can mislead users, potentially resulting in harm or mistrust.

- **Legal and Compliance Risks:** Deploying LLMs in areas that require personalized data handling without adequate safeguards could lead to violations of privacy laws and regulatory standards.

The restriction on accessing private data underscores a significant challenge in deploying LLMs across sensitive and personalized domains. Recognizing and understanding these limitations is crucial for users and developers to navigate potential risks effectively and leverage LLM capabilities responsibly.

2.3 Strategies to Overcome LLM Limitations when creating Chatbots

2.3.1 Prompt Engineering

Prompt engineering is the strategic crafting of input prompts to guide the LLM towards generating the most relevant and accurate responses. This technique involves framing queries clearly and contextually to optimize the model's output.

Advantages	**Pain Points**
Enhanced Accuracy: Improves the relevance and precision of the model's responses.	Reliance on Skill: Requires skillful crafting of prompts to avoid misguiding the model.
Contextual Relevance: Ensures that outputs are contextually aligned with the intended query.	Limited by Model's Knowledge: Effectiveness is bounded by the underlying knowledge and capabilities of the model.
Quick Implementation: Can be quickly implemented without altering the model's architecture.	May Require Iterative Tuning: Often necessitates multiple iterations to refine prompts for optimal results.

Table 2.1: Prompt Engineering: Advantages and Pain Points

2.3.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation enhances LLM outputs by integrating a retrieval component that fetches relevant external data before generating responses. This approach combines the generative capabilities of LLMs with real-time data retrieval.

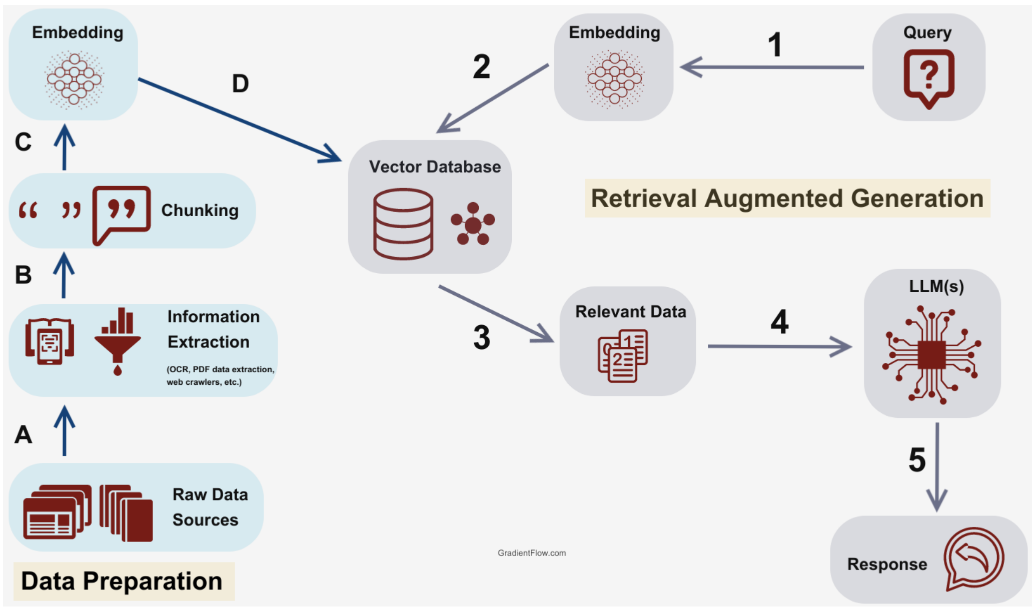


Figure 2.4: Rag process [4]

Advantages	**Pain Points**
Rich Contextual Responses: Provides more informed and context-rich outputs by leveraging external data.	Higher Latency: The retrieval process can introduce delays, affecting real-time response capabilities.
Enhanced Accuracy: Improves factual correctness by incorporating the latest data.	Scalability Issues: The retrieval component may struggle with scaling, especially with large data sets.
Adaptability to Updates: Remains relevant by accessing the most current information.	Computational Intensity: Requires more computational power, potentially increasing operational costs.

Table 2.2: Retrieval-Augmented Generation: Advantages and Pain Points

2.3.3 Fine-Tuning

Fine-tuning adjusts a pre-trained model to specialize in a specific domain or task by continuing the training on a targeted, smaller dataset. This method refines the model's parameters to better align with specific operational needs.

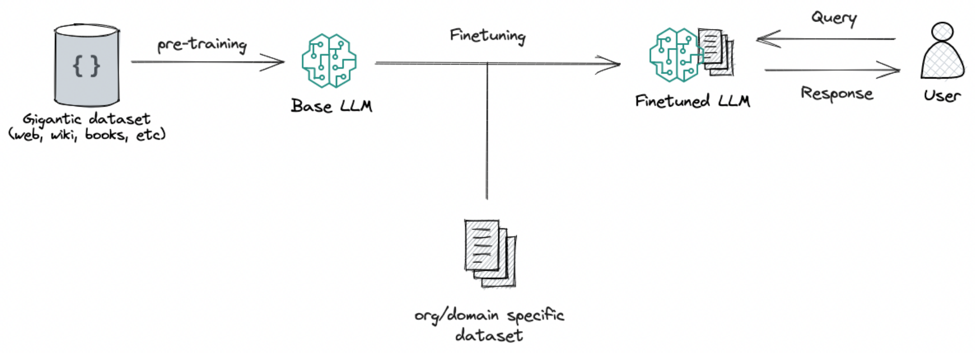


Figure 2.5: Fine tuning process [5]

Advantages	**Pain Points**
Quick Adaptation: Quickly adapts to new domains or specific tasks without extensive retraining.	Overfitting Risk: There's a risk of the model overfitting to the fine-tuning dataset, reducing its generalizability.
Improved Performance: Typically enhances performance on specialized tasks.	Dependency on Quality Data: Success heavily relies on the quality and relevance of the fine-tuning data.
Lower Computational Cost: Less resource-intensive compared to training from scratch.	Limited Scope: Improvements are confined to tasks similar to those in the fine-tuning dataset.

Table 2.3: Fine-Tuning: Advantages and Pain Points

2.4 Improuvement points - What to Choose: RAG with Prompt Engineering vs. Fine-Tuning

When deciding between Retrieval-Augmented Generation (RAG) with prompt engineering and fine-tuning, several factors need to be considered, including the specific requirements of the application, available resources, and desired outcomes. Both approaches have their distinct advantages and limitations, making them suitable for different scenarios.

2.5 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a technique that enhances the capabilities of pre-trained language models by integrating real-time data retrieval. This method pulls relevant information from external databases to enrich the model's responses, making them more accurate and contextually appropriate.

RAG leverages a hybrid approach combining traditional language model generation with dynamic information retrieval. This integration allows the model to access a vast array of up-to-date data, significantly enhancing the breadth and depth of its output.

Key Features

- **Dynamic Data Integration:** By fetching external data as needed, RAG ensures that the generated content is both current and highly relevant.
- **Enhanced Contextual Relevance:** The technique uses additional information to provide responses that are contextually enriched, which is especially beneficial in fields such as news analysis, financial forecasting, and medical diagnostics.
- **Adaptability:** RAG adapts to new queries by accessing the most relevant and recent data, offering flexibility across various applications.

Advantages

- **Up-to-date Information:** Essential for applications where current information is critical, such as in market analysis or emergency response systems.
- **Improved Accuracy:** By incorporating external data, RAG reduces reliance on potentially outdated training datasets, leading to more accurate and factual outputs.
- **Versatility:** Suitable for a wide range of applications, from customer support systems to research tools, enhancing performance by utilizing the latest available data.

2.6 Prompt Engineering

Prompt Engineering involves crafting specific inputs (prompts) to guide the behavior of a language model towards producing desired outputs. This technique is crucial for maximizing the effectiveness of pre-trained language models by optimizing how queries are structured.

Prompt engineering is the art of designing prompts that effectively leverage the innate capabilities of large language models. By fine-tuning the input, this method ensures that the output closely aligns with user expectations, reducing the need for extensive computational resources.

Key Features

- **Strategic Query Framing:** Carefully crafted prompts direct the model's response, enabling more accurate and relevant outputs.
- **Resource Efficiency:** Utilizes the existing capabilities of models without the need for additional data retrieval or retraining, making it efficient in terms of computational usage.
- **Quick Adaptability:** Allows for rapid adjustments in the model's outputs to meet new requirements or to adapt to different contexts without modifying the underlying model architecture.

Advantages

- **Cost-Effectiveness:** Reduces the need for further model training or data retrieval, lowering the overall operational costs.
- **Flexibility:** High adaptability across various domains and tasks, making it ideal for applications that require quick deployment and versatility.
- **Ease of Implementation:** Simple changes in prompt design can lead to significant improvements in model performance, facilitating ease of use and experimentation.

2.7 Retrieval-Augmented Generation (RAG) with Prompt Engineering

RAG combined with prompt engineering leverages external data sources to enhance the outputs of large language models. Prompt engineering strategically frames queries to guide the model, while RAG ensures that the responses are enriched with up-to-date and contextually relevant information from external databases.

When to Choose RAG with Prompt Engineering:

- **Need for Current Information:** If your application requires the latest data or frequently updated information, RAG is beneficial as it pulls in the most recent information.
- **Complex Queries:** For applications involving complex queries that need contextual understanding from external data, RAG excels.
- **Dynamic Domains:** In environments where information rapidly changes, such as news, finance, or research, RAG provides the flexibility to access and incorporate new data.

2.8 Fine-Tuning

Overview: Fine-tuning involves continuing the training of a pre-trained model on a smaller, domain-specific dataset. This process adjusts the model's parameters to better perform specific tasks, enhancing its ability to handle particular contexts and requirements.

When to Choose Fine-Tuning:

- **Specialized Tasks:** If your application requires high performance in a specific domain, fine-tuning is ideal.
- **Resource Constraints:** For applications where computational resources are limited, fine-tuning is less intensive compared to RAG.
- **Consistency:** When consistent performance on specific tasks is crucial, fine-tuning provides tailored improvements.

2.9 Comparison Matrix

Choosing between RAG with prompt engineering and fine-tuning depends on the specific needs of your application. RAG with prompt engineering is ideal for scenarios requiring dynamic and up-to-date information, while fine-tuning excels in specialized tasks needing consistent performance. By understanding the strengths and limitations of each approach, you can make an informed decision to optimize your large language model's performance.

Factor	**RAG with Prompt Engineering**	**Fine-Tuning**
Use Case	Dynamic, frequently updated information	Specialized, consistent tasks
Latency	Higher due to retrieval process	Lower, suitable for real-time use
Computational Resources	High, especially for large databases	Moderate, mostly during training phase
Scalability	Complex, can be resource-intensive	Highly scalable across various platforms
Performance	Superior for context-rich tasks	Superior for domain-specific tasks
Flexibility	High, adaptable to multiple domains	Focused, limited to specific domains

Table 2.4: Comparison of RAG with Prompt Engineering vs. Fine-Tuning

2.10 Conclusion

In conclusion, addressing the limitations of Large Language Models (LLMs) through Retrieval-Augmented Generation (RAG) and prompt engineering proves to be a highly effective strategy. While RAG enhances accuracy by providing access to a broad and up-to-date knowledge base, prompt engineering fine-tunes the model's response capabilities. This combination not only mitigates issues such as hallucinations and contextual errors but also offers a scalable and adaptable framework for various applications, ultimately maximizing the potential of LLMs.

Part II

State of the Art

Chapter 3

Generalities on Artificial intelligence

3.1 Machine learning

3.1.1 Definition

Machine Learning (ML), a subset of artificial intelligenc e (AI), focuses on developing algorithms that enable computers to learn from data and improve automatically through experience. Unlike traditional programming, where a computer follows predefined instructions, ML algorithms learn patterns from data to make decisions or predictions without explicit programming. For instance, instead of defining specific features of a cat, an ML algorithm is trained on thousands of images of cats to recognize patterns and features that define a cat, improving its accuracy over time.

3.1.2 Types of Machine Learning

Machine Learning can be categorized based on the learning approach, availability of labeled data, and feedback mechanism:

Supervised Learning Supervised Learning uses labeled data to learn patterns and relationships between inputs and outputs. This process, known as training or fitting, involves algorithms that learn the relationship between features and target variables.

Unsupervised Learning Unsupervised Learning analyzes and clusters unlabeled data to discover hidden patterns or groupings without human intervention. Main tasks include clustering, association, and dimensionality reduction.

Semi-Supervised Learning Semi-Supervised Learning combines a small amount of labeled data with a large amount of unlabeled data during training. This approach is useful in fields where labeled data is scarce and expensive to obtain.

Reinforcement Learning Reinforcement Learning trains software to make decisions by mimicking the trial-and-error learning process, aiming to achieve the most optimal results.

3.1.3 Limitations of Machine Learning: Challenges and Considerations

While machine learning is powerful, it faces several limitations:

- **Data Dependency**: The quality and quantity of training data are crucial. Poorly labeled or unrepresentative data can lead to inaccurate predictions.
- **Overfitting**: Complex models may overfit training data and fail to generalize to new data, resulting in poor performance on unseen data.
- **Explainability**: ML models can be complex and hard to interpret, making it challenging to explain their decisions.
- **Biased Data**: Training data biases, due to selection, human biases, or measurement errors, can lead to biased predictions.
- **Lack of Diversity**: ML models may struggle with unstructured data like images, sounds, and texts.
- **Computational Cost**: High computational power and storage resources are often required, making ML costly and time-consuming to train and implement.

3.1.4 Conclusion

In conclusion, while Machine Learning (ML) offers transformative capabilities by learning from data and making predictions without explicit programming, it faces significant challenges such as data dependency, overfitting, explainability issues, bias, and high computational costs. Addressing these limitations is crucial for the effective and ethical application of ML technologies across various domains.

3.2 Deep Learning

3.2.1 Definition of Deep Learning

Deep learning is a specialized subset of machine learning and a branch of artificial intelligence that employs multi-layered neural networks, often referred to as deep neural networks. Inspired by the biological neural networks of the human brain, deep learning models simulate complex decision-making processes. These networks are capable of learning from vast amounts of data, largely without direct human intervention, enabling computers to form hierarchical representations for recognizing patterns and making decisions.

Deep learning has become fundamental in driving the artificial intelligence applications that permeate our daily lives, from language processing and computer vision to robotics and automation. This technique excels in high-dimensional function estimation, allowing for sophisticated predictive capabilities across various fields. Its ability to automatically learn and improve from experience underpins most of the advanced AI systems in operation today, providing a powerful tool for tackling complex problems across multiple domains [16, 17, 18, 19, 20].

3.2.2 Detailed Core Concepts of Deep Learning

3.2.3 Neurons and Layers

- **Artificial Neurons:**
  - An artificial neuron is a computational unit that receives inputs (features of data), processes them through a weighted sum, adds a bias, and then applies an activation function, as explained by Kılıçarslan et al. [21].
  - The general operation can be expressed mathematically as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

  - Here, x_i represents the input features, w_i are the weights assigned to each feature, b is the bias, and f is the non-linear activation function.
- **Layer Types:**
  - **Input Layer:** Comprises neurons that directly take the input data and typically perform normalization or scaling. The structural and functional significance of the input layer is detailed in Dubey et al. [22].
  - **Hidden Layers:** These layers consist of numerous neurons that process inputs from previous layers. Each neuron in a hidden layer transforms the inputs into forms that are usable by subsequent layers, extensively discussed by Kılıçarslan et al. [21].
  - **Output Layer:** Outputs the final predictions of the neural network. The design of the output layer is typically tailored to specific tasks, such as using softmax activation for classification, explained in Dubey et al. [22].

3.2.4 Activation Functions

- **Why Non-linear?**
  - Non-linear activation functions allow neural networks to solve complex problems that are not linearly separable, essential for modeling intricate patterns in high-dimensional data, as discussed by Kılıçarslan et al. [21].
- **Key Activation Functions:**
  - **ReLU (Rectified Linear Unit):** Preferred in many neural network architectures due to its simplicity and efficiency in forward and backward propagation, highlighted by Kılıçarslan et al. [21].

$$f(x) = \max(0, x)$$

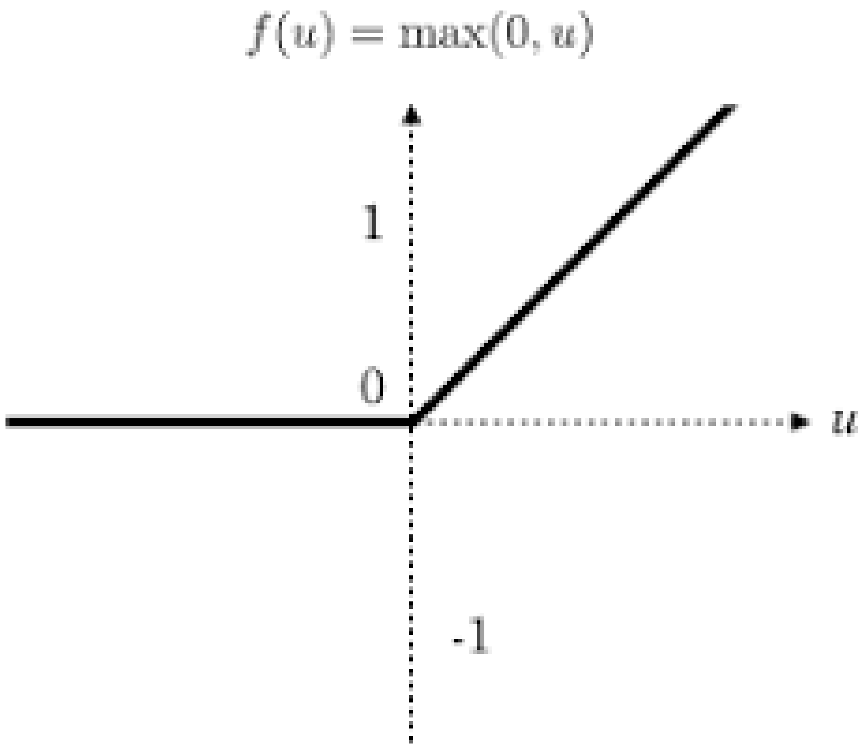


Figure 3.1: Relu acctivation function [6]

- **Sigmoid:** Best suited for output layers in binary classification, transforming inputs into probabilities between 0 and 1, as noted by Kılıçarslan et al. [21].

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

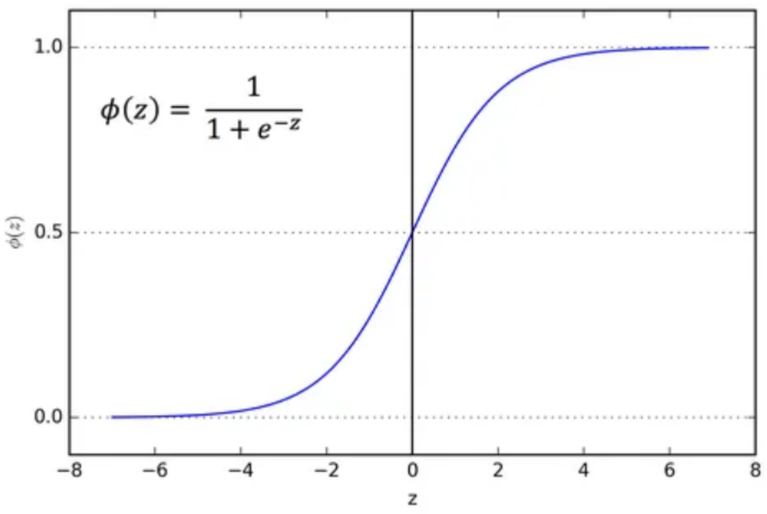


Figure 3.2: Sigmoid acctivation function [7]

- **Tanh (Hyperbolic Tangent):** Provides outputs between -1 and 1, making it useful for data that needs to be centered around zero, discussed by Kılıçarslan et al. [21].

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

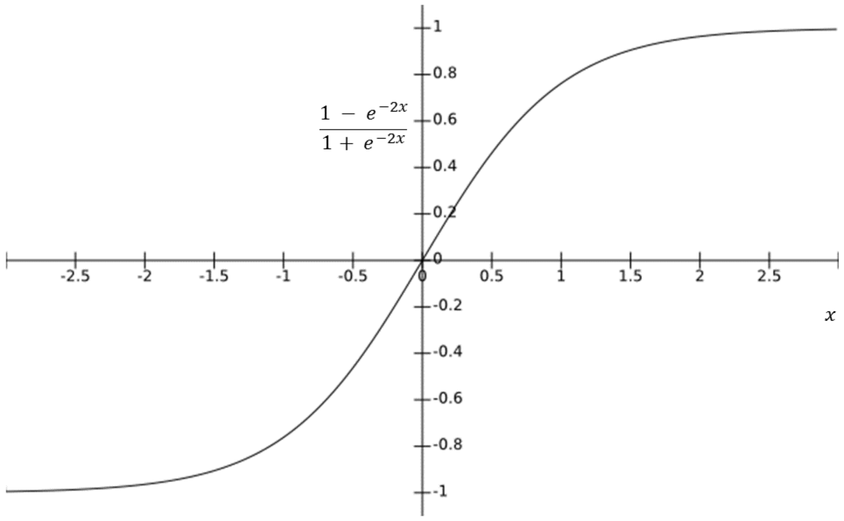


Figure 3.3: Tanh acctivation function [8]

Backpropagation and Learning

- **Mechanics of Backpropagation:**
  - **Error Computation:** The network makes an initial prediction, calculates the error using a loss function (e.g., mean squared error for regression), as discussed by Kılıçarslan et al. [21].
  - **Gradient Calculation:** The gradients of the loss function with respect to each weight are computed using the chain rule of calculus, essential for optimizing neural networks, explained in Kılıçarslan et al. [21].
  - **Weight Update Rule:** Weights are adjusted in the direction that minimizes the loss, guided by the learning rate (η), which determines the step size in the gradient descent, as described in Kılıçarslan et al. [21].

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{\partial E}{\partial w}$$

- **Optimization Algorithms:**
  - Advanced algorithms such as SGD, Adam, and RMSprop improve learning speed and convergence, ideal for training deep neural networks in various settings, explored in Ch et al. [23].

These detailed descriptions offer deeper insight into how deep learning networks function and learn. Understanding these core concepts is crucial for designing and training effective neural network models that can handle complex tasks like image and speech recognition, natural language processing, and more.

3.2.5 Conclusion

In conclusion, deep learning represents a profound shift in artificial intelligence, moving beyond traditional machine learning through its ability to autonomously process and learn from raw data. By employing intricate neural network architectures that capture hierarchical data abstractions, deep learning models excel in complex pattern recognition tasks across a myriad of domains, setting the stage for continued advancements in AI technology.

3.3 NLP and deep learning

Natural Language Processing (NLP) is a dynamic field at the intersection of computer science, artificial intelligence, and linguistics. It focuses on developing algorithms that enable computers to understand, interpret, and generate human language in a meaningful and effective manner. The ultimate goal of NLP is to build systems that can communicate with humans in their natural language, enhancing the accessibility and efficiency of human-computer interaction [24].

3.3.1 Definition of NLP

NLP is a core technology in artificial intelligence that assists machines in processing and "understanding" human language. It bridges linguistic communication between humans and machines, making it crucial for the development of AI applications that require interaction in natural language. NLP heavily leverages deep learning techniques, which utilize advanced neural networks to dramatically improve the ability of machines to translate text, recognize speech, and perform sentiment analysis. This synergy has not only propelled the capabilities of NLP but has also intertwined it deeply with data science, which supports NLP by providing robust statistical methods to analyze and model vast amounts of language data [25].

3.3.2 Evolution of NLP systems

Rule-Based Systems In its early years, NLP relied predominantly on rule-based systems, where linguists and computer scientists manually developed a comprehensive set of linguistic rules for computers to follow. Although effective for structured tasks, these systems lacked the flexibility needed to handle the ambiguity and variability inherent in natural language, limiting their applicability in more dynamic real-world scenarios.

3.3.3 Key Techniques and Methods in NLP

Natural Language Processing (NLP) employs a variety of techniques to decompose and understand human language. These fundamental methods include:

- **Tokenization**: This process divides the text into smaller units, called tokens, which can be words, phrases, or symbols. This step is essential for preparing the data for more complex analyses.
- **Syntactic Analysis**: It involves identifying the grammatical structure of a sentence, using rules to analyze the relationships between words. This enables machines to understand how words in a sentence connect with each other to form meaning.

- **Semantic Analysis**: This method seeks to interpret the meaning of words within their specific context. It goes beyond grammatical structure to understand the nuances of language, including ambiguities and variations in meanings.
- **Word Embeddings**: Embeddings, or lexical embeddings, transform tokens into numerical vectors that capture contextual and semantic aspects of words. This allows models to process text in a dense vector space, where similar words have close numerical representations, thus facilitating tasks such as semantic search and text classification.

In addition to these basic techniques, NLP also relies on advanced models for processing and generating language:

- **Statistical Models**: These models use statistical techniques to understand and predict linguistic structures. For example, they can calculate the probability that one word follows another in a sentence, which is fundamental for tasks like auto-correction and text completion.
- **Deep Learning-Based Models**: With the advent of deep learning, NLP has made significant strides. Deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are used to process long and complex text sequences. More recently, Transformer architectures, which use attention mechanisms to enhance language modeling quality, have revolutionized machine translation, text generation, and natural language understanding.

3.3.4 Deep learning models in NLP

The shift from rigid, rule-based frameworks to adaptive machine learning models marks a significant milestone in NLP. Machine learning offers a more flexible approach, as these models learn to process and interpret language directly from data. This evolution has not only enhanced the efficiency and accuracy of NLP systems but has also allowed them to rapidly adapt to new linguistic contexts and challenges, revolutionizing their scalability and utility across diverse applications.

Building on this foundational progress, the advent of deep learning architectures has further transformed the field of Natural Language Processing, propelling it into a new era of technological advancements. These sophisticated models harness the power of large datasets to automatically learn intricate representations of language, eliminating the dependence on manual feature engineering that was prevalent in earlier methods. By effectively capturing both the contextual and sequential nuances of text, deep learning techniques have significantly improved the performance of NLP systems across a broad spectrum of tasks. From machine translation to sentiment analysis and speech recognition, the capabilities of these models have not only heightened in accuracy and efficiency but have also broadened in scope. This remarkable evolution in NLP demonstrates a shift towards systems that can engage with human language in ways that are profoundly more sophisticated and nuanced, closely emulating human-level understanding and responsiveness.

Recurrent Neural Networks (RNNs) Recurrent Neural Networks (RNNs) are a class of neural networks designed specifically for handling sequential data, such as text or speech. Unlike standard feedforward neural networks, RNNs possess the unique ability to maintain a 'memory' of previous inputs using their internal state, allowing them to exhibit dynamic temporal behavior and process input sequences of any length [25].

The primary architecture of an RNN includes a loop that reuses the same weights at each time step. Mathematically, at each timestep t, the hidden state h_t of the RNN is updated based on the previous hidden state h_{t-1} and the current input x_t. This process is captured by the following equation:

$$h_t = f(W \cdot h_{t-1} + U \cdot x_t + b)$$

where:

- h_t is the new hidden state,
- f is a non-linear activation function such as tanh or ReLU,
- W and U are weight matrices for the hidden state and input, respectively,
- b is a bias term,
- x_t is the input at time t,
- h_{t-1} is the hidden state from the previous timestep.

The output y_t at each timestep is then computed based on the hidden state:

$$y_t = V \cdot h_t + c$$

where V is the weight matrix connecting the hidden state to the output, and c is a bias term for the output.

RNNs are particularly useful for tasks where the sequence of the data is important, as they can consider the entire history of previous inputs when generating the output. This capability makes RNNs ideal for applications like language modeling and machine translation [26].

Despite their potential, RNNs often present challenges in training effectively due to issues like vanishing and exploding gradients. These issues arise during backpropagation through time (BPTT) because gradients propagated over many timesteps can grow exponentially or shrink to zero, making it difficult for RNNs to learn long-term dependencies [26].

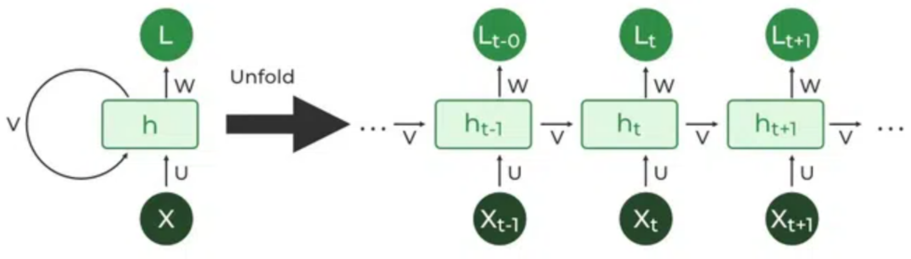


Figure 3.4: Visualisation of RNN [9]

Long Short-Term Memory Networks (LSTMs) Traditional Recurrent Neural Networks (RNNs) struggle with long-term dependencies due to the vanishing gradient problem, where gradients of the loss function can decay exponentially with time during backpropagation. This makes it challenging for RNNs to maintain information in the hidden layers over long sequences. LSTMs, introduced by Hochreiter and Schmidhuber in 1997 [27], address this issue with a more complex internal structure that includes mechanisms called gates. These gates control the flow of information, allowing the network to retain or discard information dynamically.

Architecture and Functionality: LSTM units include three types of gates:

- **Forget Gate:** Decides what information should be discarded from the cell state. This gate looks at the previous hidden state h_{t-1} and the current input x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

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

- **Input Gate:** Decides what new information is going to be stored in the cell state. It involves a sigmoid layer which decides which values to update, and a tanh layer which creates a vector of new candidate values that could be added to the state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- **Output Gate:** Decides what the next hidden state should be. The hidden state contains information about previous inputs. The sigmoid layer decides which parts of the cell state make it to the output, and then the cell state is passed through tanh (to push the values to be between -1 and 1) and multiplied by the output of the sigmoid gate, so that we only output the parts we decided to.

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

These gates allow the LSTM to mitigate the vanishing gradient problem, as they provide a way to allow gradients to flow unchanged. The architecture of LSTMs enables them to learn and remember over long sequences of inputs, making them highly effective for tasks such as time series prediction, sequence generation, and especially complex NLP tasks that require understanding over larger text sequences, such as document summarization and question answering.[27]

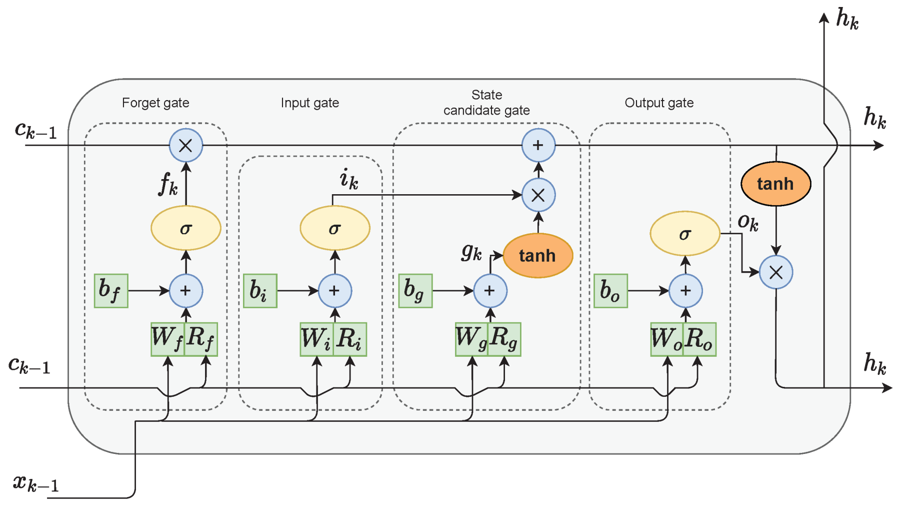


Figure 3.5: Visualisation of LSTM [10]

Transformer Models: Encoder and Decoder Details The transformer architecture is a groundbreaking neural network architecture designed for NLP tasks. It was introduced by Vaswani et al. in the paper "Attention is All You Need" [11]. The architecture relies on the self-attention mechanism to process and generate sequences, making it highly efficient and scalable compared to traditional RNNs and LSTM models.

The problem in RNNs and LSTMs is that their network sequence makes it hard to process long sentences, and the ability to perform tasks in parallel is affected by sequential computation. The transformer approaches these problems by using encoders, decoders, and self-attention.

Encoder: The encoder in the Transformer model consists of a stack of identical layers, each layer comprising two main sub-layers. The first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network. The encoder processes the entire input sequence simultaneously rather than sequentially, facilitated by the self-attention mechanism that allows each position in the encoder to attend to all positions in the previous layer.

- **Self-Attention in Encoder:** This mechanism enables the encoder to consider other words in the input sentence while encoding a specific word, ensuring that the context is comprehensively captured. It is particularly useful for understanding the relevance and relationships between different words in a sentence, irrespective of their positional distance.

Decoder: Similar to the encoder, the decoder is composed of a stack of identical layers. However, it includes a third sub-layer that performs multi-head attention over the encoder's output, in addition to the two sub-layers found in the encoder layers. This configuration enables the decoder to focus on relevant parts of the input sequence:

- **Masked Self-Attention in Decoder:** This attention mechanism is akin to the encoder's self-attention but includes a masking feature to prevent positions from attending to subsequent positions in the sequence. This ensures that predictions for position i are dependent only on known outputs at positions less than i, which is critical for text generation tasks.

- **Encoder-Decoder Attention:** This sub-layer enables the decoder to focus on relevant parts of the input sequence, utilizing the attention queries from the decoder combined with the keys and values from the encoder's output. This mechanism allows each position in the decoder to attend over all positions in the input sequence, effectively blending the features learned by the encoder into the decoder's processing.

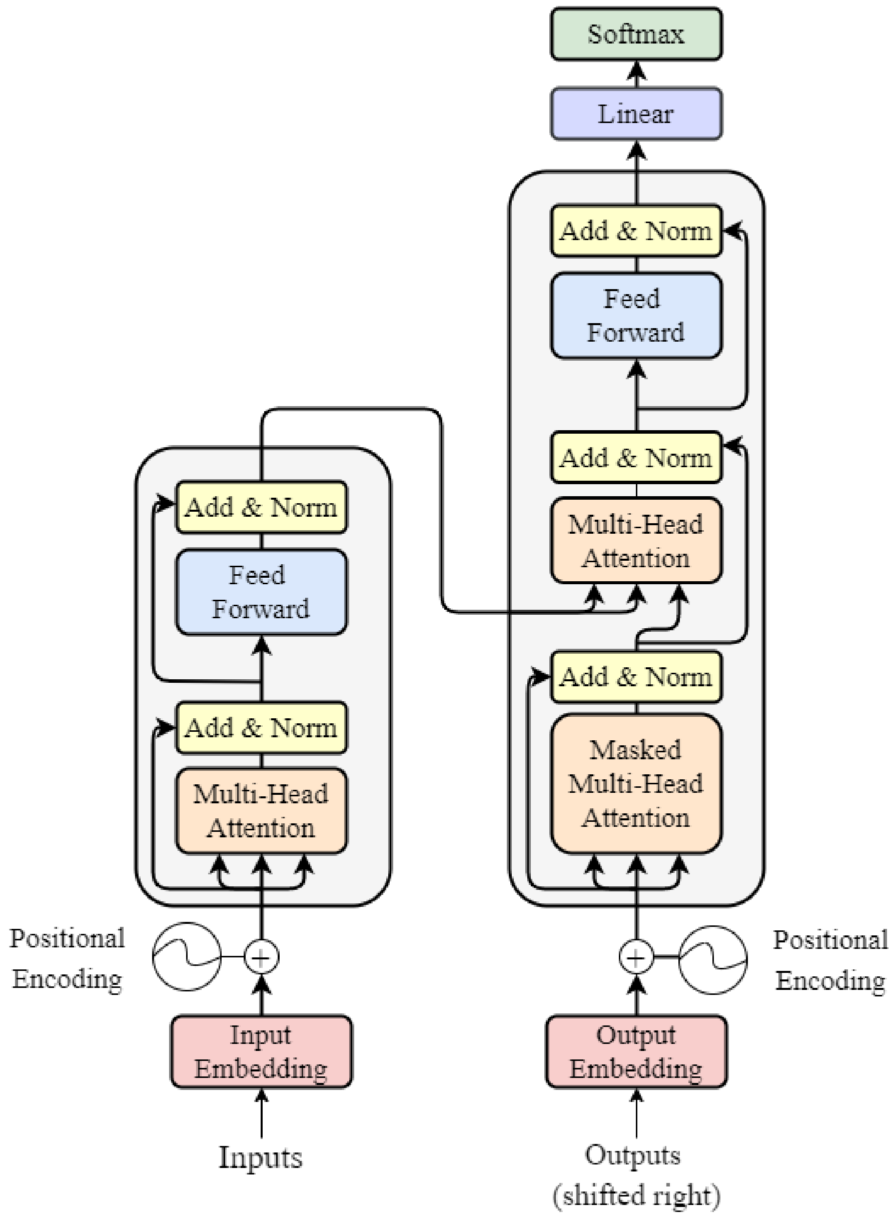


Figure 3.6: Transformers Architecture [11]

Impact and Significance: The integration of encoders and decoders with advanced attention mechanisms allows Transformers to handle dependencies more dynamically compared to models that encode or decode only. This architecture excels in tasks requiring comprehension of the entire input, like translation, where the output sequence needs to be a coherent rephrasing of the input sequence.

The encoder and decoder's advanced functionalities enable Transformers to significantly outperform earlier sequence-to-sequence models on a variety of complex NLP tasks, demonstrating considerable improvements in accuracy and efficiency.

3.3.5 Conclusion

In conclusion, the integration of deep learning within the field of Natural Language Processing (NLP) has fundamentally transformed the capabilities of language technologies. Through the application of sophisticated neural network architectures like CNNs, RNNs, and Transformers, NLP systems can now handle complex language tasks with remarkable efficiency and accuracy. These advancements underscore a significant evolution from rule-based systems to models that understand and generate language through dynamic learning, marking a pivotal shift towards achieving human-like linguistic interaction with machines.

Chapter 4

Large Language Models

The advent of transformer architectures has catalyzed a significant shift in the landscape of natural language processing, setting the foundation for the development of Large Language Models (LLMs). Transformers revolutionized NLP through their unique mechanism of self-attention, which allows models to weigh the importance of different words within a sentence, irrespective of their positional distance from each other. This ability to manage long-range dependencies is crucial for understanding the contextual nuances of language, a critical aspect that was often challenging for prior models like RNNs and LSTMs.

Transformers are inherently designed for parallel processing, unlike their predecessors that processed data sequentially. This architectural advantage means that transformers can be trained more efficiently on larger datasets—a fundamental characteristic that LLMs exploit. By utilizing vast amounts of training data, transformers can generate more accurate and contextually appropriate outputs, which is a significant step towards achieving human-like understanding and generation of language.

The scalable nature of transformer architectures enables the construction of LLMs, which are essentially larger and more powerful versions of transformer-based models. These models are not only trained on extensive text corpora but also fine-tuned through advanced techniques like transfer learning, allowing them to excel across diverse NLP tasks. The success of transformer-based LLMs, such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers), underscores their pivotal role in pushing the boundaries of what machines can understand and achieve with human language.

Thus, the evolution from transformers to LLMs represents a natural progression in the quest for more sophisticated, nuanced, and capable language models in the field of AI. This progression underscores the transformative impact that these architectures have had, paving the way for advancements that are redefining the capabilities of NLP systems.

4.1 Definition of Large Language Models (LLMs)

Large Language Models (LLMs) are advanced artificial intelligence models designed to comprehend, generate, and manipulate human language text on a vast scale. These models are characterized by their ability to achieve general-purpose language understanding and generation through the analysis of extensive datasets using deep learning algorithms [28]. LLMs have significantly evolved the field of natural language processing (NLP) by leveraging transformer-based architectures, which enable them to understand and generate text rapidly and accurately across various domains [28, 29].

4.2 Key Characteristics of Large Language Models

1. **Scale and Complexity**: LLMs utilize massive amounts of data during training, typically with at least one billion parameters, allowing them to capture intricate linguistic patterns and nuances in human language. This scale enhances their performance in understanding and generating text compared to earlier models [28, 30].
2. **Generative AI Capabilities**: LLMs are a form of generative AI, enabling them not only to analyze existing text but also to create original content based on user inputs and queries. This generative aspect sets them apart from traditional language models by facilitating the creation of new text rather than just analyzing existing data [28, 29].
3. **Adaptability and Extensibility**: LLMs serve as a foundation for customized use cases, allowing additional training on top of pre-existing models to tailor them for specific tasks. This adaptability makes LLMs versatile and applicable across a wide range of applications and industries [28, 31].
4. **Transformer Models**: LLMs leverage transformer models, a type of neural network that excels in understanding context, making them proficient in interpreting human language even in ambiguous or novel contexts. This capability to grasp context enhances their language comprehension and generation abilities compared to earlier models [28, 30].
5. **Fine-Tuning and Domain-Specific Models**: LLMs can undergo fine-tuning to adapt to specific tasks, leading to the development of domain-specific models that excel in particular areas such as programming, translation, or content generation. This fine-tuning process enhances the model's performance and applicability in specialized domains [31].

In summary, Large Language Models (LLMs) represent a significant advancement in artificial intelligence, particularly in the realm of natural language processing, due to their scale, generative capabilities, adaptability, utilization of transformer models, and fine-tuning for domain-specific tasks [28, 32, 30, 29, 31].

4.3 Applications of Large Language Models

Large Language Models (LLMs) have spurred transformative changes across multiple industries, underscoring their wide utility and significant impact. Below are some pivotal areas where LLMs are applied:

1. **Natural Language Processing, Chatbots, and Language Translation**: LLMs enhance chatbot capabilities in customer service, providing more natural and engaging user interactions, and are also instrumental in language translation, facilitating accurate and fluent cross-lingual communication [33, 34].
2. **Sentiment Analysis**: These models adeptly determine sentiments and emotions from text, aiding businesses in understanding consumer reactions and gathering actionable insights from social media and customer reviews [34].

3. **Search and Information Retrieval**: Enhancing search algorithms and information retrieval, LLMs improve the accuracy and relevance of search results, enriching user experiences and engagement across digital platforms [33].

4. **Healthcare and Life Sciences**: In healthcare, LLMs interpret complex biological data, which helps accelerate medical research and the development of innovative treatments [35].

5. **Code Generation and Software Development**: Automating coding tasks and aiding in software development, LLMs streamline the creation of software applications and even facilitate robotic training [35].

6. **Financial Services and Anomaly Detection**: In finance, LLMs are pivotal in automating the summarization of financial documents and detecting anomalies in transactions, enhancing efficiency and security in financial operations [35].

These applications demonstrate the broad and transformative impact of LLMs, extending far beyond traditional NLP tasks to areas such as customer interaction, complex data analysis, and sector-specific advancements in healthcare, finance, and more.

4.4 Notable Examples of Large Language Models

Following the introduction of transformer architectures and the definition of Large Language Models (LLMs), various instances of LLMs have been developed, each contributing uniquely to the field of NLP. These models exemplify the practical applications and theoretical advancements possible with modern deep learning technologies.

GPT Series:

- The Generative Pre-trained Transformer (GPT) series, developed by OpenAI, showcases the evolution of LLMs with its successive iterations. Each version aims to refine and enhance the model's language processing capabilities.
- **GPT-3**, introduced in 2020, was a pioneering model capable of producing text that is contextually relevant and stylistically coherent, making it suitable for a range of applications from writing assistance to conversation simulation.
- **GPT-4**, the latest in the series as of 2023, further improves on the creative and analytical abilities of its predecessors, setting new standards for the application of LLMs in even more complex tasks such as summarizing extensive documents, creating content across various formats, and even solving advanced computational problems.

LLaMA:

- The LLaMA (Large Language Model Meta AI) by Meta AI offers a range of models tailored to different scales of use, from smaller models suitable for limited-resource applications to large models designed for cutting-edge research.
- Its ability to be fine-tuned for particular tasks makes it a versatile tool in commercial NLP applications, contributing significantly to advancements in areas such as automated customer service and personalized content recommendations.

PaLM (Pathways Language Model) family:
developed by Google, includes the initial **PaLM model**, announced in April 2022 and made public in March 2023. This transformer-based large language model (LLM) possesses 540 billion parameters and was pre-trained on a corpus containing 780 billion high-quality text tokens. It is designed to handle a wide array of natural language tasks and applications. The model's pre-training involved the use of 6144 TPU v4 chips within the Pathways system, enabling highly efficient training across multiple TPU Pods. The PaLM model demonstrates the ongoing benefits of scaling, achieving state-of-the-art few-shot learning results on numerous language understanding and generation benchmarks. Furthermore, the **PaLM-540B model** not only surpasses state-of-the-art fine-tuned models on a variety of multi-step reasoning tasks but also performs comparably to humans on the newly introduced BIG-bench benchmark.

Additional LLMs:

- The development of LLMs has not been limited to these examples. Other significant models include FALCON, Mistral 7B, BLOOM, LaMDA, MT-NLG, Stanford Alpaca, FLAN UL2, GATO, PaLM, Claude, and ChatGLM.
- Each of these models brings forward unique capabilities, pushing the boundaries of what artificial intelligence can achieve in understanding and generating human-like text.

These examples of LLMs demonstrate the robustness and diversity of the current landscape in NLP technology, highlighting the profound impact of these models on various aspects of digital communication, content creation, and beyond.

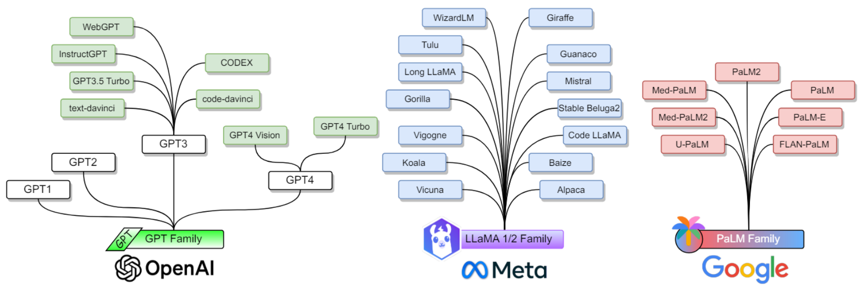


Figure 4.1: LLM Families [12]

4.5 Embeddings

4.5.1 Word Embeddings

Word embeddings are a fundamental concept in natural language processing (NLP) that represent words as numerical vectors in a high-dimensional space. This approach allows machines to understand the relationships between words, capturing their semantic meaning and context. Word embeddings are a crucial component in many NLP applications, including language modeling, text classification, and machine translation.

4.5.2 How Word Embeddings Work

Word embeddings work by assigning each word a unique vector in a high-dimensional space, typically ranging from 100 to 300 dimensions. These vectors are learned through neural networks that analyze the context in which words appear, based on the idea that words found in similar contexts likely share semantic meaning. The training process involves predicting the surrounding words in a sentence, known as the skip-gram model, or predicting the target word given the context words, known as the continuous bag of words model.

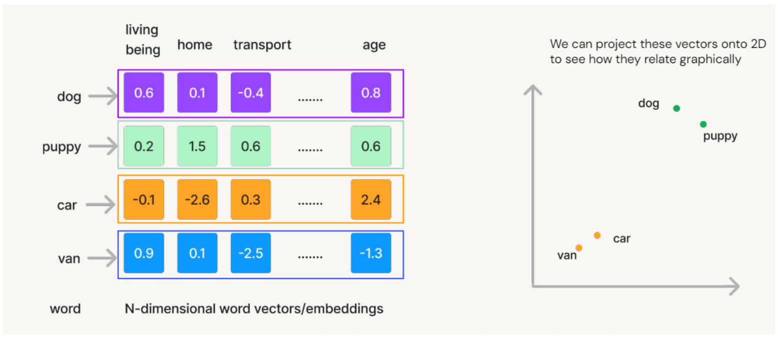


Figure 4.2: Dense Vector representation -Databricks Large Language ModelsApplication through Production

4.5.3 Word Embedding Algorithms

BERT Embeddings

BERT (Bidirectional Encoder Representations from Transformers) is a transformative model in the field of natural language processing (NLP) that generates deep contextualised word embeddings. Developed by researchers at Google, BERT has significantly advanced the state of the art in NLP tasks such as question answering, sentiment analysis, and language inference. Here's a detailed, step-by-step explanation of BERT embeddings, including theoretical aspects:

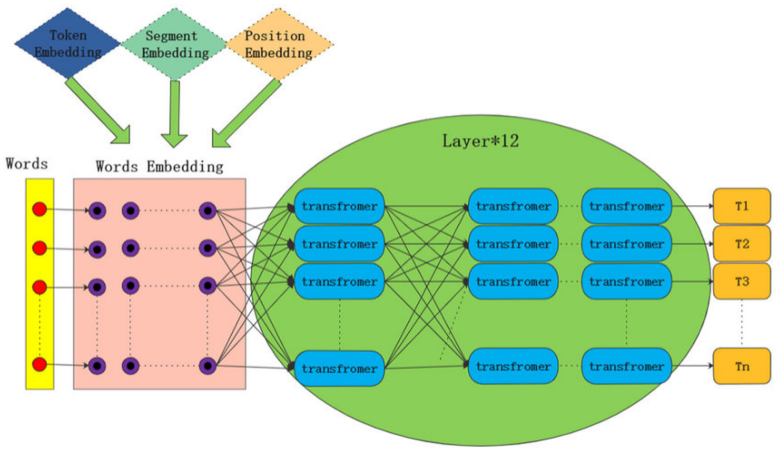


Figure 4.3: Bert Embeddings Architecture

1. **Foundation of BERT: Transformers**
   BERT is based on the Transformer architecture, introduced in the paper "Attention is All You Need" by Vaswani et al. (2017). The Transformer model discards traditional

recurrent layers and relies entirely on an attention mechanism to draw global dependencies between input and output, which allows for more parallelization and reduces training times.

- **Attention Mechanism:** The key innovation of the Transformer is the self-attention mechanism, which computes the relevance of each part of the input data to other parts. This mechanism allows the model to weigh the importance of each word in a sentence, regardless of its position, making it truly bidirectional.

2. **BERT Architecture**
BERT's architecture is a multi-layer bidirectional Transformer encoder. Here are the key components:

- **Input Representation:** BERT transforms input text into fixed-size vectors. Each token is represented by combining its corresponding token, segment, and position embeddings.
- **Token Embeddings:** Represent the tokens generated from subword tokenization (WordPiece).
- **Segment Embeddings:** Differentiate between sentences in tasks that involve multiple inputs (e.g., question answering).
- **Position Embeddings:** Indicate the position of a token within a sentence, which is crucial since the model does not use recurrent layers.
- **Bidirectional Context:** Unlike previous models that read text from left to right or right to left, BERT reads the entire sequence of words at once. This allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

3. **Pre-training BERT**
BERT is pre-trained on two unsupervised tasks:

- **Masked Language Model (MLM):** Random words in each sentence are replaced with a special token "[MASK]", and the model learns to predict the original word based only on its context. Unlike traditional left-to-right language modeling, this allows the representation to fuse the left and the right context, which helps in understanding the meaning of the sentence better.
- **Next Sentence Prediction (NSP):** The model learns to predict if a given sentence logically follows another sentence. This is important for tasks that require understanding the relationship between sentences, such as question answering and natural language inference.

4. **Fine-Tuning BERT**
Once pre-trained, BERT can be fine-tuned with additional output layers for a wide array of specific tasks without substantial modifications to the architecture:

- **Fine-tuning:** This involves training the entire model end-to-end on a smaller dataset specific to a particular task. The fine-tuning adjusts the weights of the pre-trained network to better perform the task, such as sentiment analysis or question answering.

Word2Vec

Word2Vec is a shallow, two-layer neural network model designed to reconstruct the linguistic contexts of words. It takes a large corpus of text as input and produces a high-dimensional vector space, typically consisting of several hundred dimensions. Each unique word in the corpus is assigned a corresponding vector in this space. Words that share common contexts in the corpus are positioned closely to one another, facilitating efficient similarity computations. including theoretical aspects:

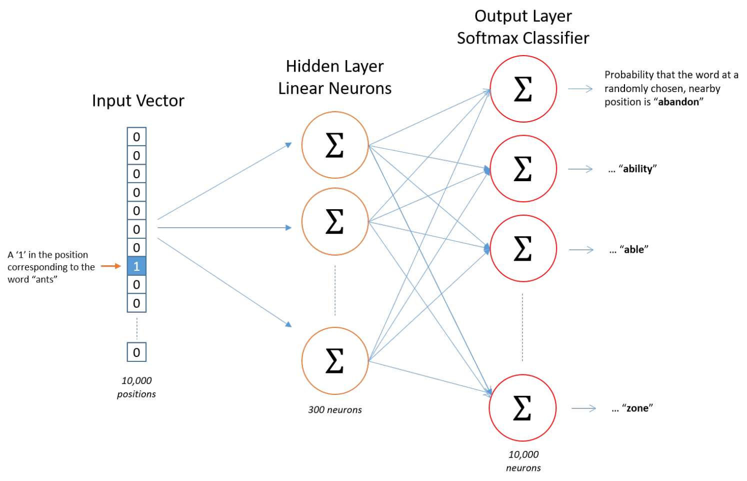


Figure 4.4: Word2Vec Embeddings Architecture

Architecture

Word2Vec can be implemented in two forms: the Continuous Bag of Words (CBOW) model and the Skip-Gram model. Both models are algorithmically similar but differ in the way they process the input and output data.

- **CBOW Model:** This model predicts a target word based on the context words surrounding it. It treats the entire context as a single observation, which tends to smooth over the distributional information, making it particularly useful for smaller datasets.
- **Skip-Gram Model:** In contrast to CBOW, the Skip-Gram model predicts the surrounding context words from a target word. It treats each context-target pair as a separate observation.

Mechanism of Learning Word Embeddings

Word2Vec utilizes a simple feedforward neural network with a single hidden layer. During training, the model adjusts the weights of the network to minimize a loss function, aiming to improve the prediction accuracy of context words given a target word.

- **Input Layer:** The model takes one-hot encoded vectors representing words from the vocabulary.
- **Hidden Layer:** This layer acts as a fully connected layer whose weights form the word embeddings. The output from this layer is essentially the word vector for the input word.

- **Output Layer:** The output layer uses a softmax function to provide a probability distribution over the vocabulary, predicting the likelihood of context words.

4.6 Vector DB

4.6.1 Vector Database Management System (VDBMS)

A Vector Database Management System (VDBMS) is a specialized form of database designed to efficiently handle vector data, which are high-dimensional representations of data objects. Unlike traditional databases that manage scalar values and structured data, VDBMSs are optimized to store, manage, and process vector embeddings that capture the complex relationships and semantic meanings inherent in various forms of unstructured data such as text, images, audio, and video. [36]

4.6.2 Vector DB VS Traditional DB

Aspect	**Traditional DB**	**Vector DB**
Data Type Handling	Primarily designed for structured data with fixed schemas. Inefficient in handling unstructured data like text, images, audio.	Optimized for managing unstructured data by transforming it into vector embeddings that are efficiently stored and retrieved.
Scalability	Struggle with the volume, velocity, and variety of modern data. Performance bottlenecks often require costly hardware upgrades.	Designed to scale effectively with large datasets, supporting high-speed similarity searches and low-latency queries.
Flexibility	Schemas are predefined and inflexible, making it difficult to adapt to changing data structures and requirements.	Schema-less and more adaptable, allowing for seamless integration of diverse data sources and evolving data structures.
Performance	Can face significant performance issues when managing large amounts of unstructured data, leading to increased costs.	More cost-effective scaling and improved performance capabilities make handling large data volumes more feasible.
Data Retrieval	Limited to exact matches and simple queries. Cannot perform complex similarity searches required by modern applications.	Capable of complex similarity searches across high-dimensional data spaces, crucial for AI-driven applications like recommendation systems.
Cost Efficiency	Handling large and complex datasets can be costly due to the need for frequent upgrades and maintenance.	Generally more cost-effective in the long run due to better scalability and the avoidance of frequent hardware upgrades.

4.6.3 Use Cases of Vector Databases

1. **Enhanced Search Capabilities:**
   - Similarity Search: Utilizes vector embeddings to perform deep, content-based searches across various media such as text, images, and audio. This technology allows platforms to offer searches that consider the context and content of the items rather than just metadata.
   - Semantic Match: Moves beyond traditional keyword matching by understanding the semantic meaning of the content, providing more accurate search results that reflect the intent and context of user queries.
   - Product Search Enhancement: Optimizes e-commerce search engines by interpreting product features and user intent at a semantic level, improving the relevancy of search results and thereby enhancing user experience.
2. **Personalization through Recommendation Systems:**
   - General Use: Vector databases power recommendation systems that personalize user experiences by matching user profiles with item embeddings. This results in highly tailored suggestions across various services.

4.6.4 How does Vector DB work

This subsection elucidates the operational mechanisms and workflow of vector databases, specialized systems designed to efficiently manage and process high-dimensional vector embeddings. The comprehensive explanation extends from the initial data input to integration with broader data ecosystems, emphasizing the technological advancements and operational efficiencies.

1. **Data Embedding**
   - Transformation Process: The process commences with the conversion of raw data from varied sources, including text, images, audio, and video, into vector embeddings. These embeddings are numerical representations that encapsulate the semantic meaning and intrinsic features of the data.
   - Technological Tools: Advanced services such as Azure OpenAI Embedding Service or using local embedding models from Hugging Face are deployed to facilitate this transformation, ensuring precise and meaningful vector representation.
2. **Indexing and Storage**
   - Algorithmic Indexing: Following vectorization, the embeddings are indexed using state-of-the-art algorithms like Product Quantization (PQ) or Hierarchical Navigable Small World (HNSW). These algorithms organize the embeddings into data structures optimized for rapid search and efficient retrieval.
   - Structured Storage: The indexed embeddings are systematically stored within the vector database, which enables swift access and robust management, underpinning the performance and responsiveness of the system.
3. **Querying and Retrieval**

- Processing of Queries: Queries are submitted to the database in vector form and processed through a comparison with the indexed vectors, utilizing predefined similarity metrics.
- Efficient Retrieval: The most congruent vectors are retrieved based on these metrics, providing users with results that are not only relevant but also precise.

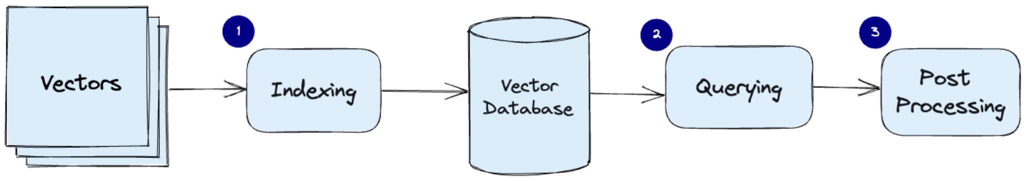


Figure 4.5: Diagram illustrating how vector databases work.[13]

4.6.5 Indexing in Vector Databases

Indexing in vector databases is a data structure technique that organizes high-dimensional vector embeddings to facilitate quick and efficient data retrieval. This process is crucial for enabling rapid similarity searches, which are essential for AI applications like recommendation systems and natural language processing. By structuring data into easily searchable formats, indexing significantly enhances the performance of vector databases, improving query speed and reducing computational load. This optimization allows vector databases to manage large datasets effectively, supporting dynamic AI-driven environments with enhanced scalability and responsiveness.

4.6.6 Similarity Measures in Vector Databases

Similarity measures are pivotal in various domains like natural language processing and computer vision for assessing the closeness of vector embeddings. These measures play a critical role in applications ranging from semantic search and recommendation systems to anomaly detection. The choice of an appropriate similarity metric is crucial as it directly influences the effectiveness of algorithms used for retrieving the most relevant vectors in response to a query.

.

Euclidean Distance Euclidean distance measures the straight-line distance between two vectors in a multidimensional space and is calculated using the formula:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

where a and b are vectors. It is most suitable for systems where the magnitude of vector components represents meaningful data. However, it is sensitive to the scale of the data, which can be a limitation in environments where different dimensions do not share a common scale.

Cosine Similarity Cosine similarity, on the other hand, evaluates the cosine of the angle between two vectors, providing a measure of orientation and not magnitude:

$$\text{sim}(a,b) = \frac{a \cdot b}{||a|| \, ||b||}$$

This measure is independent of the vector magnitude, making it ideal for comparing the similarity in direction of two vectors, often used in text analysis and other fields where orientation is more significant than magnitude.

Dot Product Similarity Dot product similarity is a straightforward approach that sums the products of corresponding components of two vectors:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

This metric can reflect both magnitude and orientation, making it useful in environments where both properties of vectors are essential for performance.

4.6.7 Vector Database Examples

In the dynamic field of data science, vector databases are crucial for managing high-dimensional data efficiently. These databases are tailored for applications in artificial intelligence (AI), machine learning (ML), and natural language processing (NLP), where traditional databases fall short due to their inability to handle complex data relationships and unstructured formats effectively.

4.6.7.1 ChromaDB

ChromaDB is an open-source vector database that enhances the development of Large Language Model (LLM) applications by allowing easy management of text documents and conversion of text to embeddings for similarity searches.

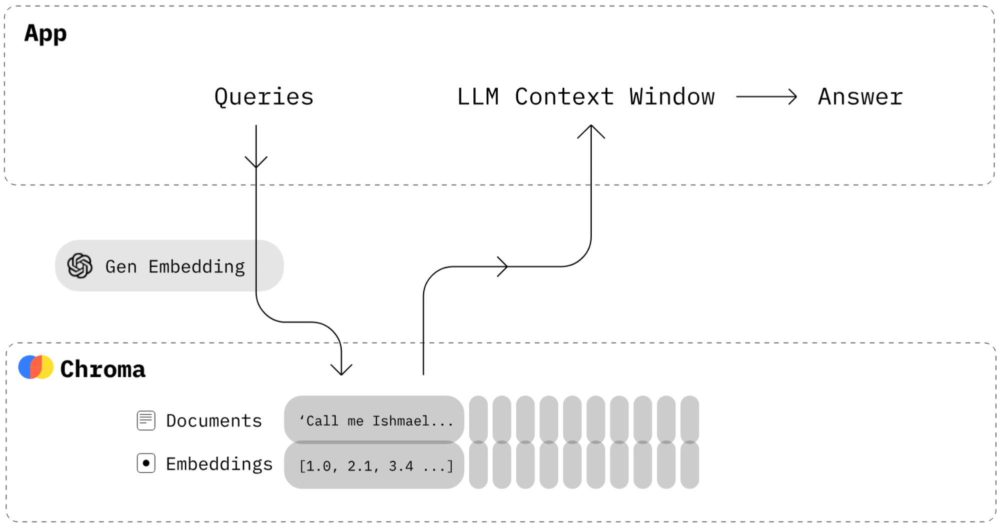


Figure 4.6: Building llms using chromadb

Key Features:

- Open-source, with over 8,000 GitHub stars.
- Supports extensive queries, filtering, and density estimates.
- Scalable from a Python notebook to production clusters seamlessly.

4.6.7.2 Pinecone

Pinecone is a managed vector database designed specifically for handling the challenges of high-dimensional data. It supports scalable, real-time data ingestion and low-latency search, making it ideal for large-scale ML applications.

Key Features:

- Provides a fully managed service.
- Ensures real-time data ingestion and low-latency searches.
- Highly scalable and integrates with LangChain.

4.6.7.3 Qdrant

Qdrant is both a vector database and a powerful tool for vector similarity searches, suitable for developing complex applications like matching systems and recommendation engines.

Key Features:

- Open-source with over 12,500 GitHub stars.
- Features a versatile API and advanced filtering options.
- Designed with a cloud-native architecture for superior scalability and efficiency.

These vector database examples illustrate the variety of tools available for effectively managing high-dimensional vector data, each equipped with unique features to cater to different application needs in AI, ML, and beyond.

4.7 Introduction to Prompt Engineering

Prompt engineering has rapidly evolved into a critical technique within the field of Natural Language Processing (NLP), especially pertinent to the deployment of Large Language Models (LLMs). This method involves the strategic use of task-specific instructions or prompts that guide the behavior of these models, simplifying the application of deep learning models and extending their utility across diverse domains [37].

4.7.1 Defining Prompt Engineering

Prompt engineering is the art and science of crafting textual prompts that effectively guide the outputs of pre-trained models like LLMs. These prompts act as direct instructions, which the models interpret to generate desired responses without any modifications to their underlying architecture, democratizing the use of sophisticated machine learning models [?].

4.7.2 Components of Effective Prompts

Effective prompting of Large Language Models (LLMs) involves a series of key components that structure the interaction and guide the model's responses:

- **Input Data:** This is the primary information that the model encounters and needs to process, influencing the model's output accuracy [38].
- **Exemplars:** Specific examples of correct input-output pairs provided within the prompt, serving as a guide to the desired format and quality of the model's responses [38].
- **Instruction:** A clear and concise textual description of the expected output, directing the model's attention and processing [38].
- **Indicators:** Tags or formatting elements that introduce structure to the prompt, aiding in the effective segmentation of the prompt [38].
- **Context:** Additional information that enhances the model's understanding of the input, improving accuracy and relevance of responses [38].

4.7.3 Types of Prompting

Prompting techniques are categorized based on complexity and the specificity of the task they address:

- **Zero-shot Prompting:** Involves presenting a model with a task it has not explicitly seen during training, testing the model's ability to generalize from its training data [39].
- **Few-shot Prompting:** Provides the model with a few examples of the task, helping it understand the task's requirements more clearly [39].
- **Chain of Thought Prompting:** Encourages the model to simulate a step-by-step reasoning process before arriving at a conclusion, useful for complex problem-solving [39].
- **Instruction-based Prompting:** Direct and task-specific prompts that guide the model to produce outputs that adhere closely to the given instructions [39].
- **Hybrid Prompting:** Combines elements of the above strategies to optimize model performance for more complex or nuanced tasks [39].

Advanced Prompting Techniques: Tree of Thoughts and Graph of Thoughts

Advanced techniques like Tree of Thoughts (ToT) and Graph of Thoughts (GoT) are designed for tackling complex problems that benefit from strategic problem-solving, allowing for dynamic exploration of possible solutions. These approaches use natural language to evaluate progress and integrate search algorithms for enhanced problem-solving capabilities

Each of these prompting techniques demonstrates the flexibility and adaptability of prompt engineering, applying LLMs to a range of NLP tasks from simple classification to complex problem-solving scenarios [39].

4.8 Conclusion:

In conclusion, the progression from traditional NLP methods to the transformative introduction of transformer architectures has set the stage for the development of Large Language Models (LLMs). These models, embodying advanced transformer technology, are crucial in handling the complexities of human language by efficiently managing long-range dependencies and context. The evolution to LLMs illustrates a significant leap forward in AI, providing more nuanced, accurate, and contextually relevant language processing capabilities. This advancement not only enhances the practical applications of AI in various domains but also signifies a pivotal shift towards models that more closely emulate human-like understanding and interaction.

Chapter 5

RAG AND AI AGENTS

5.1 Introduction to RAG

In the field of artificial intelligence, large language models (LLMs) such as GPT-3 and BERT have revolutionized text generation. Despite their remarkable capabilities, they face significant challenges, including reliability and the timeliness of the information provided. Due to their reliance on static training data, a major limitation is their inability to access private organizational data. As a result, LLMs may produce outdated, inaccurate, or even “hallucinated” responses, inventing fictional information.[40]

To address these challenges, fine-tuning LLMs for specific tasks is often employed. This method requires substantial resources in terms of annotated data and processing capacity. Moreover, the increasing complexity of the algorithms used in fine-tuning LLMs leads to a significant rise in time and computational resource costs. Advanced algorithms require intensive computations and robust computing infrastructures to function efficiently, which can considerably burden operational costs. Additionally, continuous updating is necessary to incorporate recent information developments, further increasing the complexity and associated costs.

Retrieval-Augmented Generation (RAG) emerges as an innovative response to these challenges. RAG redirects LLMs towards reliable and up-to-date knowledge sources, enabling better control over the generated content and offering more transparency to users. This approach is particularly relevant for organizations looking to securely integrate their own private data into the text generation process, thus overcoming one of the main limitations of LLMs.

In this chapter, we will explore the impact of RAG on the use of LLMs, focusing on how it addresses issues of reliability, timeliness, and access to private data. We will also examine how RAG, by enhancing the accuracy and relevance of the generated responses, makes LLMs more suited for applications in professional and sensitive environments.

5.2 Definition

Retrieval-Augmented Generation (RAG) is a framework originally developed by researchers at Meta AI, which merges information retrieval and text generation, thereby enriching the responses of LLMs with data from external sources such as databases and websites. As explained by Luis Lastras, an expert in language technologies at IBM Research, the operation of RAG can be likened to that of a model relying on external resources, similar to consulting a book, rather than being confined solely to its internal knowledge. This approach enhances the accuracy and relevance of the responses produced by LLMs, thus providing results that are richer in context and tailored to the specific needs of the user.'[40, 41, 42]

5.3 RAG Pradigms

5.3.1 Naive Rag

In the Naive Retrieval-Augmented Generation (RAG) method, the process involves three main phases: Indexing, Retrieval, and Augmentation & Generation. This basic form of RAG is a straightforward implementation of the concept and follows these steps:

Indexing:

- **Loading Data:** Import of all documents or information to be used, which can be in various formats such as PDF, HTML, Word, Markdown, or from websites or databases.
- **Data Splitting:** Division of large documents into smaller pieces or chunks, typically less than 500 characters each, but adjusted according to the LLM's context limitations.
- **Data Encoding (embeddings):** Conversion of data into a vector form using an encoding model to make them comprehensible for computers.
- **Data Storage:** Saving these vector embeddings in a vector database for easy search and retrieval.

Retrieval:

- When a user query is received, it is transformed into a vector using the same encoding model as in the indexing phase.
- The system calculates similarity scores between the query vector and the vectors of chunks in the corpus, identifying and retrieving the top K chunks that have the highest similarity with the query.

Augmentation & Generation:

- The selected chunks, along with the original query, are compiled into a prompt.
- The LLM then responds to this prompt, drawing on its inherent knowledge or the information contained in the provided documents.
- For multi-turn dialogues, the existing conversational history can be incorporated, enhancing the model's capability in ongoing interactions.

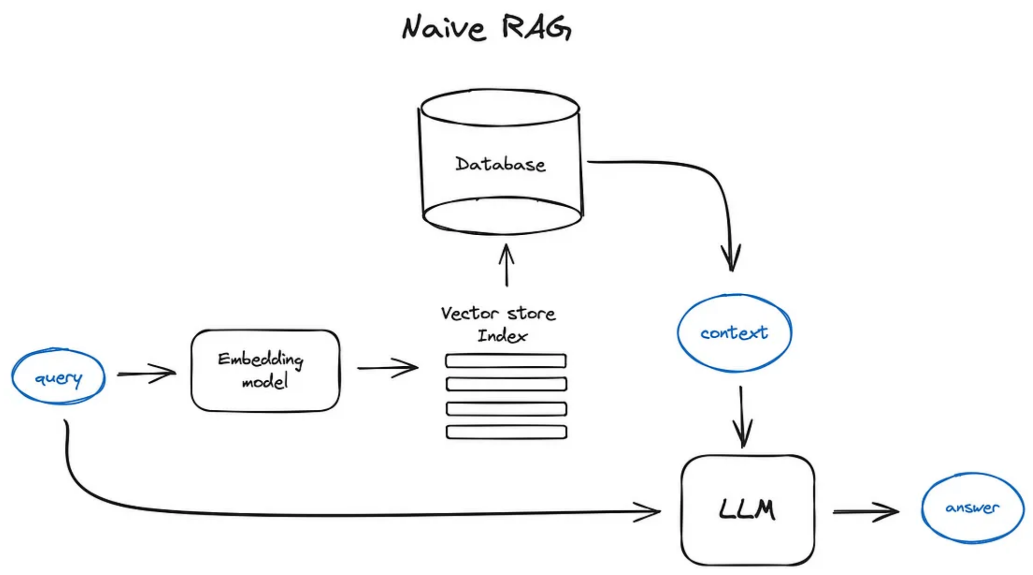


Figure 5.1: Naive rag pipeline [14]

Naive RAG is characterized by its "Retrieve-Read" framework and straightforward approach of combining retrieval and generation. The process involves indexing, retrieval, and generation, allowing the inclusion of external information to enhance the LLM's responses. This makes Naive RAG a fundamental and essential approach in the development and application of Retrieval-Augmented Generation systems.[40, 41, 42]

5.3.2 Advanced RAG

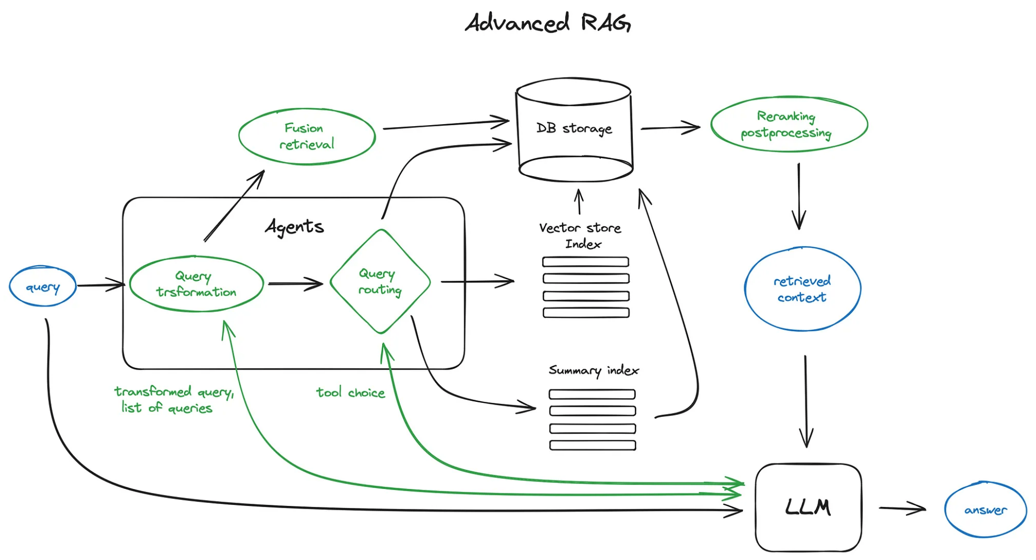


Figure 5.2: advanced rag pipeline [14]

Chunking and Vectorization

Chunking: In the context of Retrieval-Augmented Generation, 'chunking' involves segmenting large texts into manageable pieces. The size of the chunks, adjusted according to the embedding model, aims to balance the necessary context for understanding and the specificity required for effective retrieval. Various chunking strategies are employed to optimize information retrieval. Among these are fixed-size chunking, which ensures uniformity, and recursive chunking, suited for texts with varied structures. These methods aim to balance the amount of context needed for understanding and the precision necessary for effective search within the RAG system.

Vectorization: Following chunking, the next step is to convert these text chunks into vectors using suitable models. Models like bge-large or embeddings from the E5 family are often used for this purpose. The choice of embedding model is crucial as it determines the quality and utility of the vectors in the retrieval process. To evaluate and compare the performance of various embedding models, one can refer to the rankings available on the Hugging Face Model Hub, a recognized leaderboard in the field of artificial intelligence.

Overall, chunking and vectorization are critical for transforming large textual documents into a format that can be utilized by machine learning models, particularly in the context of RAG systems. Proper execution of these steps is essential to ensure that the system can retrieve the most relevant and contextually appropriate information in response to user queries.

Search Index

Vector Stores: In the context of Retrieval-Augmented Generation (RAG) systems, embeddings are numerical representations in a multidimensional space, typically for unstructured data. Given this specificity, traditional Relational Database Management Systems (RDMS) are unsuitable for storing these vector embeddings. In response, vector databases, or vector stores, are developed specifically to store and retrieve embeddings efficiently. These databases are unique in their ability to support various embedding models and to implement advanced search algorithms for identifying similar vectors, making them indispensable in RAG applications.

In RAG systems, two types of vector storage are distinguished:

- **Vector Storage Libraries:** Libraries such as FAISS and ANNOY are designed for scenarios where vector data remains static, suitable for batch data applications that do not require frequent updates. They are limited to storing embeddings and do not support CRUD (Create, Read, Update, Delete) operations, complicating the addition or modification of data in existing indexes.
- **Vector Databases:** On the other hand, vector databases like Pinecone, Weaviate, and Milvus facilitate dynamic management of vector data, including CRUD operations. These systems store both the embeddings and the associated documents, making them suitable for environments requiring frequent updates, flexibility, and scalable data management. These databases are optimized for speed and scalability, meeting the needs of applications requiring fast queries and regular updates.

Hierarchical Index:

In the case of large databases containing numerous documents, efficient search and retrieval of relevant information become crucial. A strategic approach to address this challenge is the implementation of a two-level indexing system. This system comprises two distinct indices: one for summaries and the other for document fragments.

- **Summary Index:** The first index contains summaries of each document. These summaries are comprehensive representations of the main ideas or content of the documents. The purpose of this index is to provide an initial, high-level filtering mechanism. When a query is received, it is first executed against this summary index. This step acts as a preliminary filter, quickly narrowing down the potential pool of documents to those most likely to contain relevant information based on their summaries. This helps to swiftly eliminate irrelevant documents and focus the search on a more manageable subset.
- **Chunk Index:** The second index consists of smaller, more detailed fragments of each document. After identifying relevant documents through their summaries, the query is then processed against this second index. This deeper search is limited to the already filtered set of documents. The fragment-based index allows for more granular and precise information retrieval. It examines the details of each relevant document, retrieving the exact fragments containing information aligned with the query.

Fusion Retrieval / Hybrid Retrieval

Hybrid retrieval, or fusion retrieval, is an advanced method that combines the strengths of keyword-based search and semantic search to improve the accuracy and relevance of search results. This approach compensates for the limitations of each method by using their complementary capabilities, thus offering a more robust solution for complex information retrieval tasks[**?**].

1. **Keyword-Based Search:** Traditionally, keyword-based search is the cornerstone of search engines. It operates on the principle of exact word matching. This method utilizes sparse vector search, where documents are represented as vectors in a multi-dimensional space. Each dimension corresponds to a unique keyword, and the value in each dimension reflects the presence and frequency of the keyword in the document, often enhanced by the TF-IDF (Term Frequency - Inverse Document Frequency) weighting scheme. Despite its effectiveness for retrieving precise terms, this method is limited by its sensitivity to synonyms and variations in wording, which can result in missing contextually relevant documents that do not contain the exact keywords.
2. **Semantic Search:** In contrast, semantic search, also known as dense vector search, uses machine learning algorithms to understand the meanings behind words and phrases, creating richly detailed vector representations of data. These dense representations, with predominantly non-zero values, capture contextual relationships between words, allowing the system to effectively handle synonyms, linguistic nuances, and even multilingual content. The main challenge here is that semantic search may overlook important keywords if these are not well represented in the training data for the vector models.
3. **Combining Both Methods:** Hybrid retrieval integrates these two approaches by independently processing a query through both keyword-based and semantic search methods. Each generates a list of results based on their respective algorithms. The fusion of these results is accomplished through a reranking process, where relevance scores from both methods are combined using a weighted sum formula:

$$\text{hybrid score} = (1 - \alpha) \times \text{keyword score} + \alpha \times \text{semantic score}$$

   where α is a parameter ranging from 0 to 1, allowing the system to balance the influence of keyword accuracy and semantic context according to the specific needs of the user[**?**, 43].

Reranking

1. **Definition:** Reranking, in the context of information retrieval, is a crucial process that involves reorganizing the documents or data segments initially retrieved by a search system. This technique adjusts the ranking of these items to improve the quality of the final results presented to the user. Initially, a basic retrieval step extracts a broad set of potentially relevant results using methods such as sparse vector similarity or dense vector search. However, these initial rankings often lack precision, requiring further refinement to ensure that the most relevant results are prioritized at the top of the list.

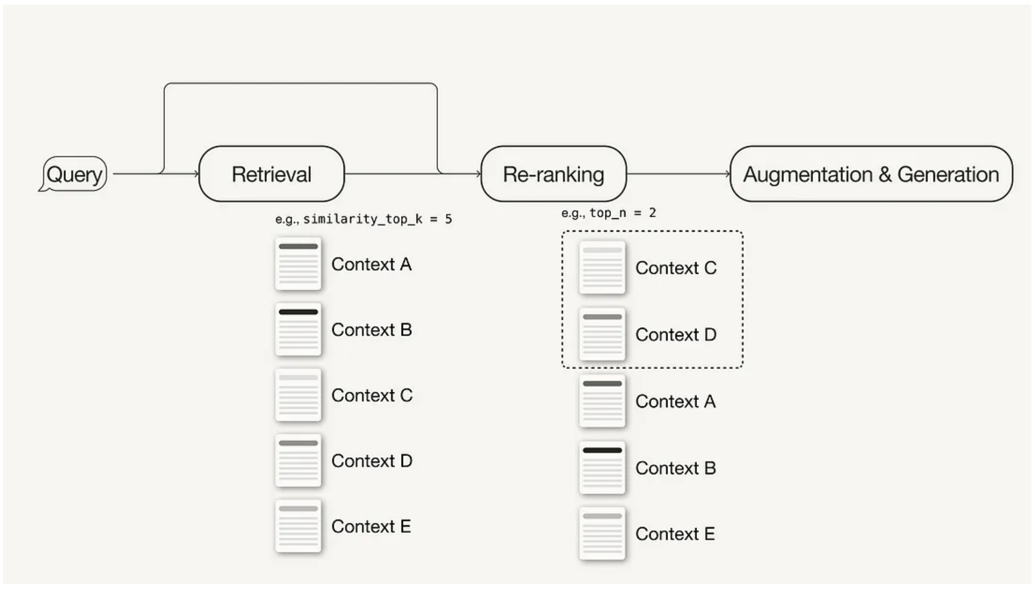


Figure 5.3: Reranking process [15]

2. **Necessity of Reranking:** The main purpose of reranking is to address the deficiencies of the initial retrieval phase, which often produces a list with a suboptimal order. The initial list, though expected, may include relevant documents that are not highly ranked due to the limitations of traditional ranking algorithms that heavily rely on keyword matching or basic vector similarity. Consequently, important contextual cues and deeper content relevance, crucial for high-quality retrieval, might be overlooked.

3. **Implementation of Reranking:** Reranking can be executed by various methods:

   - *Rule-Based Methods:* These rely on predefined metrics such as Diversity, Relevance, and Mean Reciprocal Rank (MRR). They use simple but effective criteria to reorganize results based on specific, often quantifiable, attributes of the search results.
   - *Model-Based Approaches:* These involve more sophisticated techniques using machine learning models, particularly those in the BERT series like Span BERT. These models are trained to better understand the nuances of language and context, providing a more nuanced reranking based on a deeper semantic understanding.
   - *Specialized Reranking Models:* Tools such as Cohere reranker or bge-reranker-large are specifically designed for reranking tasks and are optimized to improve relevance and diversity in reranked results.
   - *Large Language Models:* Models like GPT can also be adapted to perform reranking by leveraging their extensive training on vast datasets to assess relevance and context more effectively than traditional methods.

Query Transformation

1. **Purpose:** Query transformation is an essential process in the retrieval pipeline, aiming to convert user queries into a form that enhances the effectiveness and efficiency of the retrieval system. This transformation adjusts the query to better match the underlying data structure and retrieval algorithms[44].

2. **Techniques:**

   - *Expansion:* This technique involves adding related terms or synonyms to the query to increase the scope of the search and improve the chances of retrieving relevant documents.
   - *Simplification:* Simplifying the query can help focus the search on key terms, reducing the noise and increasing precision.
   - *Normalization:* Converting terms to a standard format, such as lowercasing, stemming, and lemmatization, ensures consistency in how queries and documents are matched.

3. **Benefits:** Query transformation enhances the search process by ensuring that the query aligns more closely with the semantics of the data and the capabilities of the search engine. It helps bridge the gap between the user's natural language expression and the technical requirements of the retrieval system[44].

Chat Engine

One of the essential aspects of building an effective RAG (Retrieval-Augmented Generation) system, capable of processing the same query multiple times, lies in the dialogue logic. This includes accounting for the context of the dialogue, similar to classic chatbots from the pre-LLM era. This feature is crucial for managing follow-up questions, anaphoras, or arbitrary user commands relating to the previous dialogue context.

There are several modes of the chat engine:

- **ContextChatEngine:** A popular and relatively simple approach. This method starts by retrieving the relevant context for the user's request, then sends this context to the LLM, accompanied by the chat history from the memory buffer. This allows the LLM to consider the previous context when generating the next response.
- **CondenseChatEngine:** A simple chat mode based on a query engine that directly queries your database. For each interaction in the chat:
  1. Generate a standalone question: First, generate a standalone question from the context of the conversation and the last message received.
  2. Query the Query Engine: Next, query the query engine with the condensed question to obtain a response.
- **Condense Plus Context Mode:** A more sophisticated method, where during each interaction, the chat history and the last message are condensed into a new query. This query is then sent to the index, and the retrieved context is transmitted to the LLM with the original user message to generate a response.

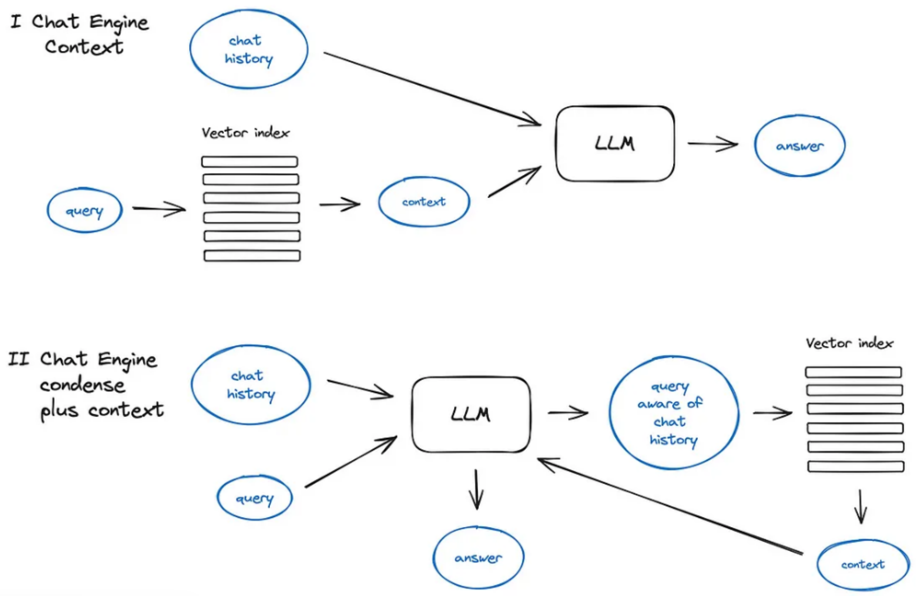


Figure 5.4: Chat Engine types [14]

It is important to note that there is also support for OpenAI agents in the LlamaIndex Chat Engine, offering a more flexible chat mode. Similarly, Langchain supports OpenAI's functional API.

Illustration of different types and principles of Chat Engine

While other types of dialogue engines, such as the ReAct agent, exist, we will directly address the agents themselves later This segmentation of dialogue management techniques showcases the evolution of interactive dialogue systems, leveraging advanced technologies to enhance the accuracy and relevance of responses in continuous conversation contexts.

Example of Application

Illustration with a chatbot programmed to answer questions about programming. Depending on the nature of the question, the chatbot uses routing to consult the relevant index, thereby improving the accuracy and speed of the responses provided.

5.4 Rag Evaluation

Generation metrics used in rag

The primary objective of a Retrieval-Augmented Generation (RAG) system is to produce a meaningful and contextually relevant output. The evaluation of such systems must ensure that the generated content effectively utilizes the retrieved context without merely duplicating it, thereby avoiding redundancy and incomplete responses. It is essential to develop metrics that accurately assess each of these criteria.[45]

RAG systems utilize two key metrics to evaluate the performance of a Large Language Model's (LLM) output: **Faithfulness** and **Answer Relevance**.

5.4.1 Faithfulness

This metric evaluates how factually correct the generated answer is based on the provided context. A response is considered faithful if it can be verified against the context without introducing inaccuracies. However, a high faithfulness score does not necessarily imply relevance to the query posed, as the response could simply reiterate the context without addressing the query's specifics[46].
Faithfulness score = $\frac{\text{Number of claims in the generated answer that can be inferred from given context}}{\text{Total number of claims in the generated answer}}$

5.4.2 Relevancy

This metric determines the pertinence of the response to the posed question. An answer scoring high on this metric directly addresses the query, enhancing user satisfaction. However, it is penalized if it includes irrelevant or repetitive information, or if it fails to provide a comprehensive answer.

To summarize, the evaluation of a RAG system involves a combination of assessing the factual accuracy (Faithfulness), contextual relevance (Answer Relevance), of the responses. These metrics ensure that the generated answers are not only accurate and relevant but also provide meaningful engagement in conversational applications.

5.5 AI Agents

The advent of generative AI, notably through platforms like ChatGPT, initially popularized applications focused primarily on dialogue systems that interact over a fixed corpus using Retrieval-Augmented Generation (RAG). As the field matures, there is a growing pivot towards next-generation AI applications embodying a more sophisticated architectural paradigm—AI agents. These agents are envisioned not merely as tools for dialogue but as robust systems capable of complex reasoning, strategic planning, and dynamic interaction with both data and environments.

Recent strides in foundation models, particularly those like GPT-4, alongside burgeoning open-source initiatives such as AutoGPT and BabyAGI, illustrate a significant shift. These models are no longer confined to the traditional frameworks of response generation but are evolving into autonomous agents that exhibit a higher degree of functionality and intelligence. Unlike the zero-shot prompting approach, where a user inputs a query and receives a static response, these agentic systems integrate planning, iterative loops, reflection, and advanced control structures to manage tasks comprehensively from start to finish. This enhanced capability is further augmented by their proficiency in utilizing external tools, plugins, and function calling, enabling them to perform a wide array of general-purpose tasks.

The research community is actively exploring the potential of these AI agents, particularly debating the merits of single-agent versus multi-agent systems. Single-agent architectures excel in well-defined problem spaces with minimal need for external feedback, whereas multi-agent systems are designed for scenarios requiring collaboration and diverse execution paths. Multi-agent systems leverage the communicative and reasoning abilities of LLMs, allowing multiple specialized agents to interact and solve complex problems collaboratively.

In summary, AI agents represent a significant evolution in artificial intelligence, expanding the capabilities of generative models beyond simple dialogue to encompass comprehensive task management and strategic planning. These advancements promise to unlock new potential across various applications, driving further innovation and development in the field.

5.5.1 Definition :

AI agents are sophisticated entities powered by language models capable of planning and executing actions to fulfill specific goals across multiple iterations. These agents are complex systems comprising potentially single or collaborative multiple agents working together to address various challenges [47]. Each agent, equipped with distinct personas, utilizes a range of tools to perform tasks either autonomously or as a part of a team. A key feature of some AI agents is their ability to retain and retrieve information outside their immediate interactions, significantly enhancing their functionality.

5.5.2 Core Components of AI Agents

The core functionality of an AI agent is encapsulated in the triad of "brain, perception, and action," essential for the agents to comprehend, reason, and interact with their environment [47]. This conceptual framework allows AI agents not only to interact dynamically with external data sources but also to alter these sources through sophisticated toolsets. Thus, AI agents transcend traditional static data interactions, moving towards a more interactive and mutable engagement with data.

- **Decision-making Thought:** AI agents, particularly those based on large language models (LLMs), exhibit a sophisticated decision-making process. Guided by prompts, these agents can decompose complex tasks into manageable subgoals [48], methodically think through each part (sometimes exploring multiple paths) [49], and incorporate lessons from past experiences [50] to enhance decision-making in complex situations. This capability significantly boosts the autonomy and problem-solving effectiveness of a single LLM-based agent.
- **Tool-use:** LLM-based agents possess the capability to employ external tools and resources to execute tasks, thus expanding their operational functionalities [51, 52, 53]. This ability enables agents to operate more effectively within diverse and dynamic environments, handling various interactive and data-manipulation tasks.
- **Memory:** The memory capabilities of LLM-based agents involve both short-term in-context learning [54] and long-term storage using external vector databases [55]. Such memory functionalities allow agents to preserve and retrieve information over extended periods, enhancing contextual coherence and the ability to learn from interactions [56].

5.5.3 Types of AI Agent Architectures

AI agents can be structured in various architectures, each designed to optimize different aspects of task execution and collaboration:

- **Single Agent Architectures:** These systems rely on a single language model to perform all planning, reasoning, and tool execution. In these architectures, the agent operates independently, though there may be mechanisms for human feedback [47].
- **Multi-Agent Architectures:** In contrast, multi-agent systems involve two or more agents that may use the same or different language models. These agents can share or have access to distinct tools, and each agent typically maintains its own persona [47].
- **Vertical vs. Horizontal Architectures:** Multi-agent systems can be organized in vertical architectures where one agent leads and others report to them, or in horizontal architectures where agents operate as equals in a collaborative environment. These setups reflect different strategies for task division and collaboration, suitable for varying operational needs and complexity levels [57].

The exploration and implementation of AI agents in various configurations offer a promising avenue for enhancing the interactivity and capability of language model-based systems. By structuring agents in single or multi-agent systems, and further refining their operational models to vertical or horizontal architectures, we can tailor AI systems to meet specific operational and collaborative needs effectively.

Part III

Proposed Solution

Chapter 6

Proposed Solution

6.1 Design of the solution

In response to the comprehensive study of the technologies currently employed by the company, and after closely examining their pain points, we have decided to adopt a modern approach to develop our chatbot assistant. This initiative is part of our commitment to leveraging cutting-edge technology to enhance our operational efficiency and customer interaction. In the following sections, we will outline the design of this solution, revisiting the initial needs and objectives that we aim to achieve. We will also detail the necessary steps for its implementation, showcasing our concrete vision for its application.

6.1.1 Client Requirements identification

To elaborate on the needs for this project, we must address the specific requirements essential for the effective operation of the customer service chatbot at Voxist. This chatbot, designed to serve as a competent and knowledgeable first point of contact, aims to handle a wide range of queries and tasks, from accessing tickets and private client data to managing complex company documents and executing orders. Here is a structured breakdown of the needs and objectives:

1. **Sensitive Data Access:**
   - **Client and Patient Data:** The chatbot must be able to securely access and interact with private data including detailed tickets where clients describe their issues. This includes fields containing sensitive client information, which are crucial for providing personalized and effective support.
   - **Company Documentation:** The chatbot must have access to internal documents that describe troubleshooting processes and methods. This includes ITSM diagnostics and client-validated documentation, which must be dynamically retrieved and used in accordance with confidentiality standards to ensure data security.
2. **Command Execution:**
   - The chatbot should be capable of carrying out actions directly via commands. This includes routine troubleshooting tasks such as resetting passwords, unlocking accounts, or software installations, which the chatbot must manage either through scripted actions or direct intervention.

3. **Dynamic Interaction and Real-Time Updates:**

   - **Context Awareness:** The system must be adept at understanding and adapting to the ongoing context of a client or patient, such as recognizing an open ITSM ticket or identifying specific needs based on the user's history and current state.
   - **Real-Time Data Management:** Despite the inherent limitations of LLMs in updating real-time, the chatbot must incorporate techniques like Retrieval-Augmented Generation (RAG) to retrieve the most relevant and updated information from a mix of private databases and public data sources.

4. **Handling Complex Cases:**

   - **Simple vs. Complex Requests:** The chatbot must differentiate and handle simple requests requiring quick solutions and more complex scenarios where higher-level intervention is necessary. For complex cases, the chatbot must be able to efficiently escalate the problem to human agents, ensuring smooth transition and communication.

5. **User Interface and Experience:**

   - **Ease of Use:** The interface must be intuitive, allowing users to navigate and interact easily without confusion, which is particularly important in high-tension environments like customer service.

6. **Saving the Conversation**

The design and implementation of this chatbot require a clear understanding of these needs to ensure thatit enhances the efficiency and effectiveness of customer service operations at Voxist. The following sections will delve deeper into the design of this solution, detail the implementation steps thoroughly, and present a vision of its practical application.

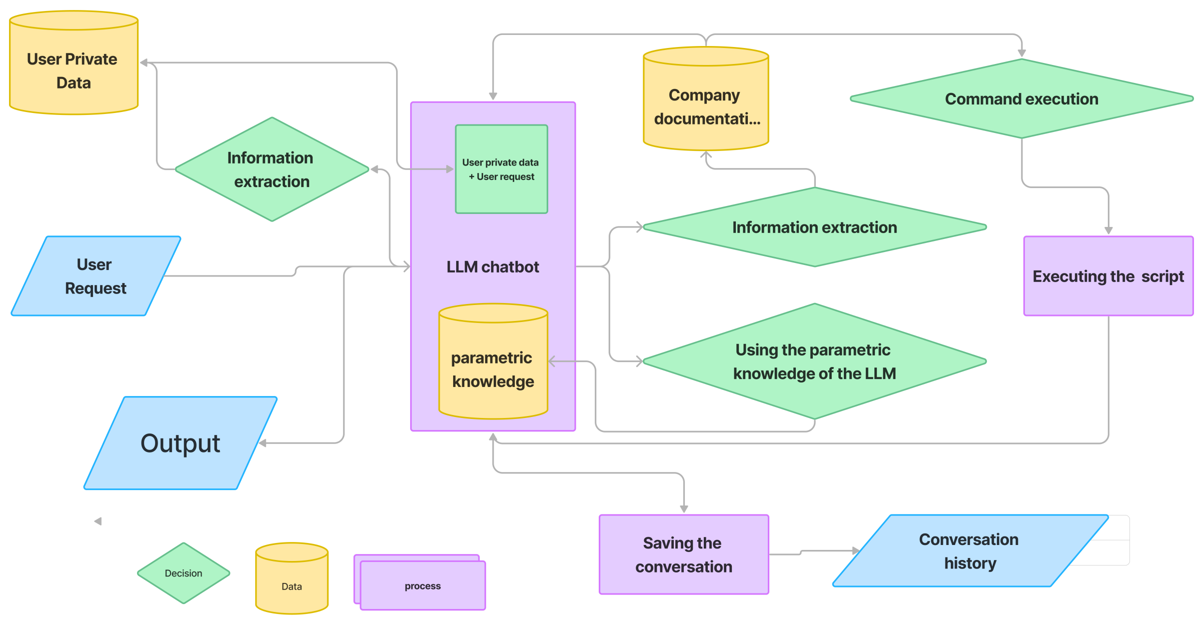


Figure 6.1: Processing scenario

6.1.2 Proposed Solution: Implementing a Structured Multi agent system

Overview

The envisioned solution for enhancing client service within the company involves constructing a sophisticated multi-agent system. This system is centered around a Meta Agent, which functions as the orchestrator, orchestrating interactions among various specialized sub-agents tailored to specific roles. Each sub-agent acts as a tool or extension of the Meta Agent, equipped to handle different aspects of the service process based on user inputs and enriched prompts.

Technologies Used

The development of the multi-agent chatbot system involved the integration of several advanced technologies and platforms. These technologies played crucial roles in ensuring robust performance, scalability, and seamless integration with existing business operations. The key technologies used are as follows:

- **OpenAI APIs**
  The OpenAI APIs, specifically the GPT-4o model, were central to the chatbot's natural language processing capabilities. The APIs facilitated the seamless generation of human-like text, enabling the system to understand and respond to user queries with high accuracy and relevance. The choice of GPT-4o was driven by its balance of cost efficiency and performance, making it suitable for dynamic customer service environments.

- **Llama Index**
  The Llama Index was used as a core framework for managing and retrieving large volumes of text data. It enabled the system to perform advanced Retrieval-Augmented Generation (RAG) tasks, ensuring that responses were not only contextually accurate but also backed by relevant information from the company's document repository. This integration was crucial for maintaining the depth and reliability of interactions.

- **GROQ Cloud**
  GROQ Cloud provided the computational infrastructure necessary to handle the intensive processing demands of the multi-agent system. Its scalable and efficient cloud-based solutions ensured that the system could manage high volumes of concurrent interactions without compromising on performance. This infrastructure was essential for the real-time processing and delivery of responses, supporting the system's operational efficiency.

- **Hugging Face**
  The Hugging Face platform was utilized for accessing a variety of pre-trained models and tools that enhanced the system's natural language understanding capabilities. By leveraging Hugging Face's extensive library of models, the system was able to incorporate state-of-the-art techniques in NLP, improving its ability to comprehend and generate complex responses. This platform also facilitated the integration of additional models, providing flexibility and scalability.

- **Langfuse**
  Langfuse played a critical role in monitoring and optimizing the performance of the LLMs used in the system. It provided tools for tracing, evaluating, and managing prompts,

as well as metrics for cost and latency analysis. Through Langfuse, the system was able to track the cost per million tokens and compare the latencies of different models, ensuring that the chosen model, GPT-4o, offered the best trade-off between cost and performance. This ongoing monitoring and optimization were vital for maintaining the system's efficiency and reliability.

System Components

1. **Meta Agent (Orchestrator):** This is the central decision-making agent that coordinates all other sub-agents. It assesses user inputs, determines the necessary action, and delegates tasks to the appropriate sub-agents. The Meta Agent ensures seamless integration of responses and actions to provide a cohesive user experience. It is enhanced with a comprehensive prompt that contains all operational guidelines, ensuring it makes informed decisions and effectively manages the orchestration of sub-agents.
2. **User Context Agent:** Using a RAG (Retrieval-Augmented Generation) mechanism, this agent manages a database of user profiles stored in JSON format. It retrieves user-specific information relevant to the interaction, ensuring that all communication and solutions are tailored to individual user needs and historical interactions.
3. **Documentation Agent:** Another RAG-based agent, this one facilitates access to the company's internal documentation. It retrieves information and solutions pertinent to user issues, including detailed processes and internal company knowledge that may aid in resolving user queries.
4. **Execution Agent:** Responsible for executing commands as part of the solution process, this agent handles operational tasks that require interaction with the company's systems, such as resetting passwords or updating user settings.
5. **Conversation Saving Agent:** This agent archives all user interactions, ensuring that every conversation is recorded for future reference. This is crucial for maintaining continuity in user service and for leveraging past interactions to train and improve the system.

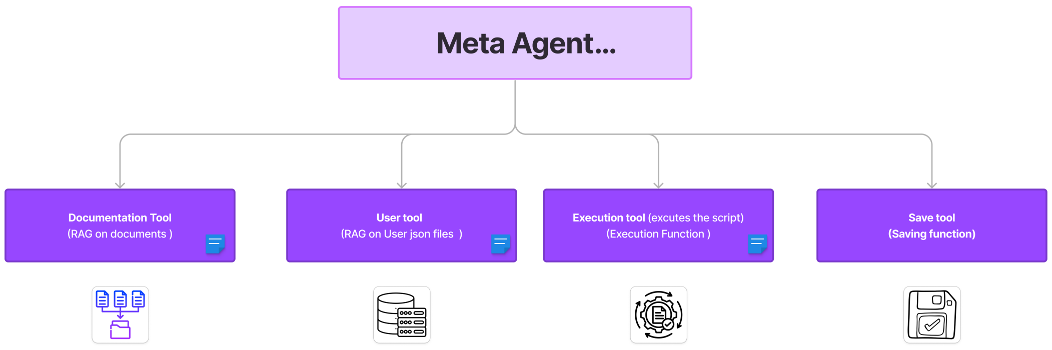


Figure 6.2: Solution Architecture

6.1.3 Operational Workflow

we have developed a streamlined operational workflow that ensures efficient handling of user interactions and optimal performance of our intelligent systems. This workflow includes the following steps:

- **Initial Interaction:** Users initiate contact through a defined interface, presenting their issue or request.
- **Input Assessment:** The Meta Agent analyzes the input and decides which agents need to be engaged based on the nature of the request and the context provided.
- **Agent Coordination:** Depending on the assessment, the Meta Agent may call on:
  - The User Context Agent to fetch personalized user data.
  - The Documentation Agent to access relevant procedural content.
  - The Execution Agent if a direct action needs to be performed.
- **Action and Response Formulation:** The Meta Agent compiles the outputs from the various agents, formulates a coherent response or set of actions, and communicates this back to the user.
- **Conversation Archiving:** The Conversation Saving Agent archives the entire interaction for compliance, training, and enhancement purposes.

6.1.4 Benefits of the Multi-Agent System

- **Personalized Interactions:** By leveraging the User Context Agent, the system ensures that all interactions are deeply personalized, increasing user satisfaction and service efficiency.
- **Resource Accessibility:** The Documentation Agent provides quick access to necessary information, reducing the time needed to resolve queries.
- **Operational Efficiency:** The Execution Agent automates parts of the solution process, streamlining operations and reducing manual intervention.
- **Continuous Improvement:** The Conversation Saving Agent allows for ongoing learning and system refinement, enhancing the system's performance over time based on real interactions.

This multi-agent system architecture not only promises to enhance the efficiency and effectiveness of client service operations but also ensures adaptability and scalability. By integrating advanced technologies such as RAG with strategic agent roles, the system is poised to transform client service management in a dynamic business environment.

6.2 Implementation of the solution

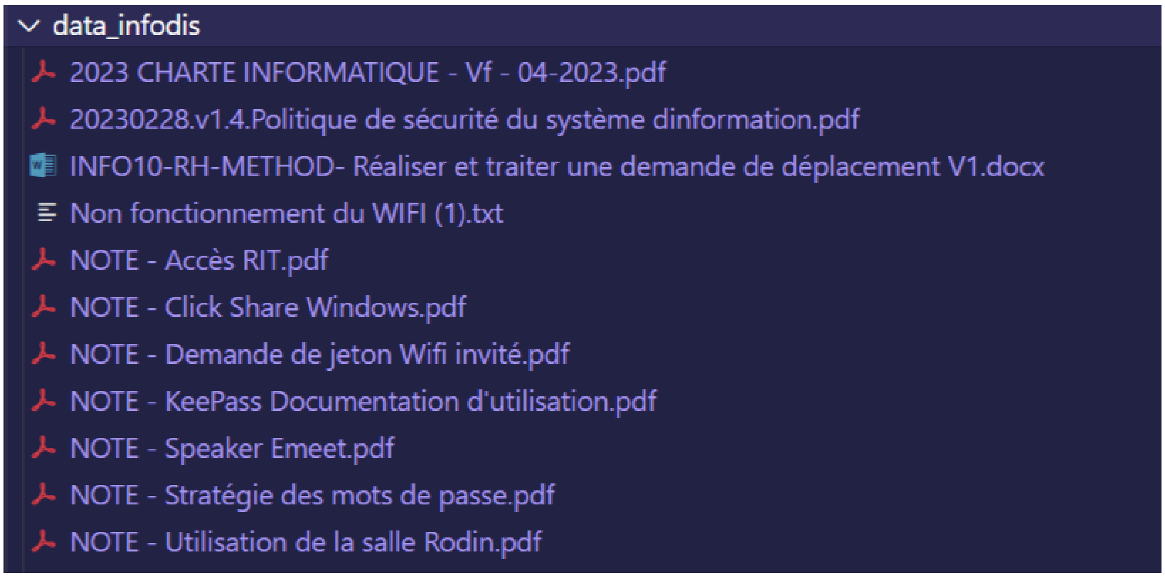
This chapter focuses on the practical implementation of the multi-agent system designed for enhancing client services. It details the critical steps involved, from the development of the dataset to the deployment of system components and the design of the user interface. The following subsections will provide a detailed account of each step, highlighting the methodological choices, technologies employed, and the outcomes achieved during the implementation process.

6.2.1 Development of the documentation tool -Advanced RAG system

This section outlines the development of an advanced Retrieval-Augmented Generation (RAG) method utilizing the Llama Index framework. Each phase of the development process is focused on specific aspects of the RAG tool.

Document Collection

The initial phase of the RAG development process commences with the acquisition of internal company documents. These documents, numbering 25 in total and comprising a mix of PDFs and DOC files, serve as the primary data sources for the project. Each document contains up to 30 pages, providing a substantial volume of content for analysis.



data_infodis
2023 CHARTE INFORMATIQUE - Vf - 04-2023.pdf
20230228.v1.4.Politique de sécurité du système dinformation.pdf
INFO10-RH-METHOD- Réaliser et traiter une demande de déplacement V1.docx
Non fonctionnement du WIFI (1).txt
NOTE - Accès RIT.pdf
NOTE - Click Share Windows.pdf
NOTE - Demande de jeton Wifi invité.pdf
NOTE - KeePass Documentation d'utilisation.pdf
NOTE - Speaker Emeet.pdf
NOTE - Stratégie des mots de passe.pdf
NOTE - Utilisation de la salle Rodin.pdf

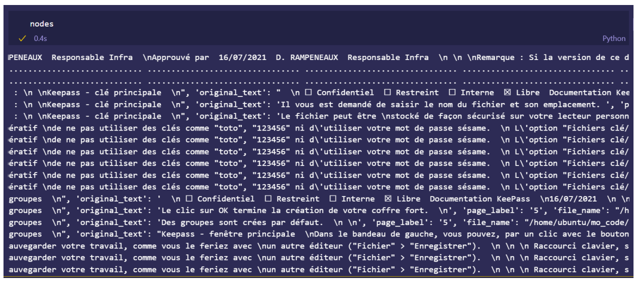
Figure 6.3: A glimpse of the documents used in the RAG

Document Cleaning

The collected documents undergo thorough cleaning to remove any undesired characters and annotations, such as part-of-speech tags. This step ensures the purity and usability of the data for subsequent processes.

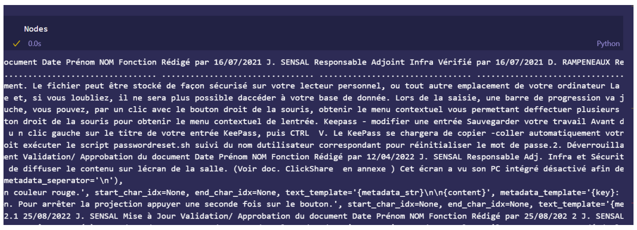
Data Preparation and Vector Database Configuration

The cleaned documents are then prepared for embedding and storage. The Qdrant database, selected for its robust vector data handling capabilities, is configured as follows:



```
nodes
✓ 0.4s                                                                                             Python
IPENEAUX  Responsable Infra  \nApprouvé par  16/07/2021  D. RAMPENEAUX  Responsable Infra  \n \n \nRemarque : Si la version de ce d
.......................... ................................ ................................ ..
.......................... ................................ ................................ ..
 : \n \nKeepass - clé principale  \n", 'original_text': "  \n ☐ Confidentiel  ☐ Restreint  ☐ Interne  ☒ Libre  Documentation Kee
 : \n \nKeepass - clé principale  \n", 'original_text': 'Il vous est demandé de saisir le nom du fichier et son emplacement. ', 'p
 : \n \nKeepass - clé principale  \n", 'original_text': 'Le fichier peut être \nstocké de façon sécurisé sur votre lecteur personn
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
ératif \nde ne pas utiliser des clés comme "toto", "123456" ni d\'utiliser votre mot de passe sésame.  \n L\'option "Fichiers clé/
groupes  \n", 'original_text': '  \n ☐ Confidentiel  ☐ Restreint  ☐ Interne  ☒ Libre  Documentation KeePass  \n16/07/2021  \n \n
groupes  \n", 'original_text': 'Le clic sur OK termine la création de votre coffre fort.  \n', 'page_label': '5', 'file_name': "/h
groupes  \n", 'original_text': 'Des groupes sont crées par défaut.  \n \n', 'page_label': '5', 'file_name': "/home/ubuntu/mo_code/
groupes  \n", 'original_text': "Keepass - fenêtre principale  \nDans le bandeau de gauche, vous pouvez, par un clic avec le bouton
auvegarder votre travail, comme vous le feriez avec \nun autre éditeur ("Fichier" > "Enregistrer").  \n \n \n Raccourci clavier, s
auvegarder votre travail, comme vous le feriez avec \nun autre éditeur ("Fichier" > "Enregistrer").  \n \n \n Raccourci clavier, s
auvegarder votre travail, comme vous le feriez avec \nun autre éditeur ("Fichier" > "Enregistrer").  \n \n \n Raccourci clavier, s
```

(a) a part of the documentation before cleaning

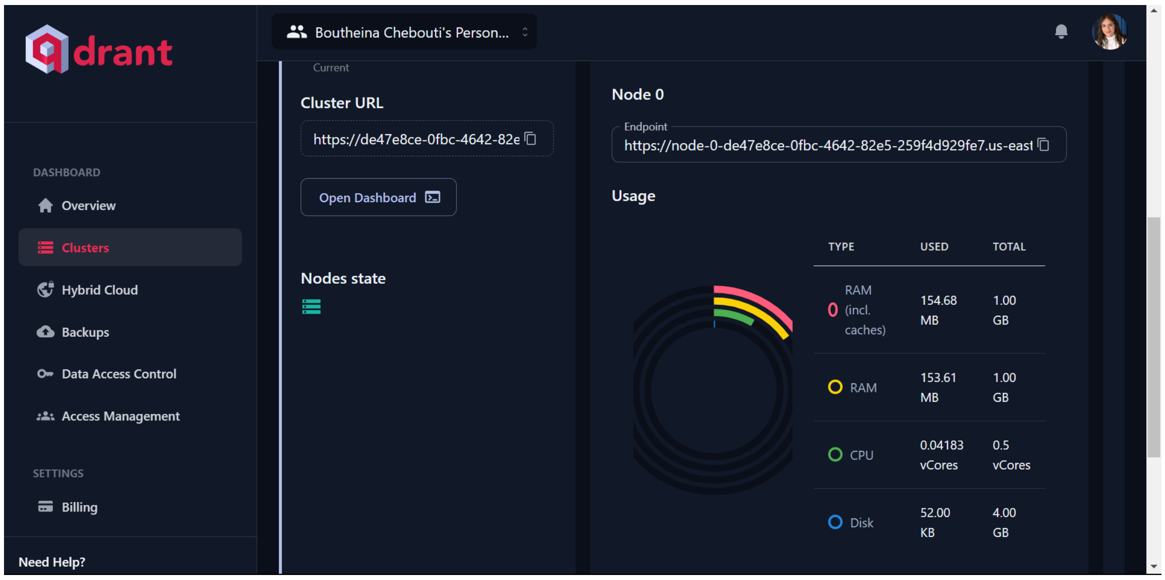


```
Nodes
✓ 0.0s                                                                                             Python
ocument Date Prénom NOM Fonction Rédigé par 16/07/2021 J. SENSAL Responsable Adjoint Infra Vérifié par 16/07/2021 D. RAMPENEAUX Re
................................ ................................ ................................
ment. Le fichier peut être stocké de façon sécurisé sur votre lecteur personnel, ou tout autre emplacement de votre ordinateur La
e et, si vous loubliez, il ne sera plus possible daccéder à votre base de donnée. Lors de la saisie, une barre de progression va j
uche, vous pouvez, par un clic avec le bouton droit de la souris, obtenir le menu contextuel vous permettant deffectuer plusieurs
ton droit de la souris pour obtenir le menu contextuel de lentrée. Keepass - modifier une entrée Sauvegarder votre travail Avant d
 u n clic gauche sur le titre de votre entrée KeePass, puis CTRL  V. Le KeePass se chargera de copier -coller automatiquement votr
oit exécuter le script passwordreset.sh suivi du nom dutilisateur correspondant pour réinitialiser le mot de passe.2. Déverrouilla
ent Validation/ Approbation du document Date Prénom NOM Fonction Rédigé par 12/04/2022 J. SENSAL Responsable Adj. Infra et Sécurit
 de diffuser le contenu sur lécran de la salle. (Voir doc. ClickShare  en annexe ) Cet écran a vu son PC intégré désactivé afin de
metadata_seperator='\n'),
n couleur rouge.', start_char_idx=None, end_char_idx=None, text_template='{metadata_str}\n\n{content}', metadata_template='{key}:
n. Pour arrêter la projection appuyer une seconde fois sur le bouton.', start_char_idx=None, end_char_idx=None, text_template='{me
2.1 25/08/2022 J. SENSAL Mise à Jour Validation/ Approbation du document Date Prénom NOM Fonction Rédigé par 25/08/202 2 J. SENSAL
```

(b) a part of the documentation after

Figure 6.4: Comparision between the documentation before and after cleaning

- Establishment of a collection within the Qdrant vector store.
- Configuration of a free trial setup, including:
  - 4GB Disk Storage
  - 1GB RAM
  - 0.5 vCPU



Cluster URL
https://de47e8ce-0fbc-4642-82e

Node 0
Endpoint
https://node-0-de47e8ce-0fbc-4642-82e5-259f4d929fe7.us-east

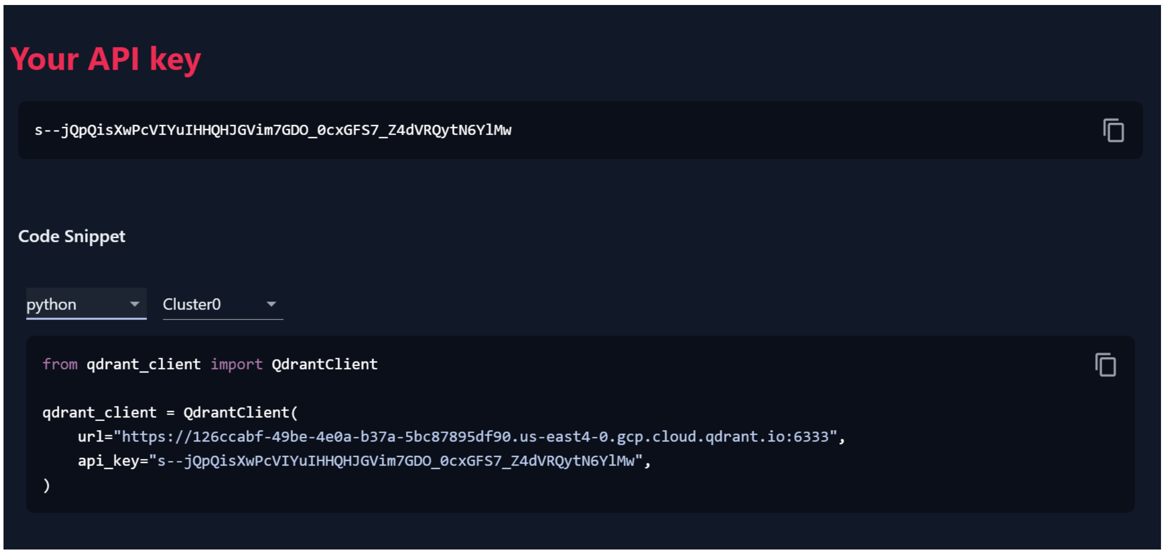
Usage

TYPE	USED	TOTAL
RAM (incl. caches)	154.68 MB	1.00 GB
RAM	153.61 MB	1.00 GB
CPU	0.04183 vCores	0.5 vCores
Disk	52.00 KB	4.00 GB

Figure 6.5: Qdrant Vector store cloud

Database Integration

Integration of the Qdrant database with our Python project is achieved by configuring the API key and cluster URL, thereby establishing a seamless connection for data handling.



```
s--jQpQisXwPcVIYuIHHQHJGVim7GDO_0cxGFS7_Z4dVRQytN6YlMw
```

```python
from qdrant_client import QdrantClient

qdrant_client = QdrantClient(
    url="https://126ccabf-49be-4e0a-b37a-5bc87895df90.us-east4-0.gcp.cloud.qdrant.io:6333",
    api_key="s--jQpQisXwPcVIYuIHHQHJGVim7GDO_0cxGFS7_Z4dVRQytN6YlMw",
)
```

Figure 6.6: Qdrant API key Configuration

Document Embedding and Storage

Documents are embedded into the vector store using the chosen embeddings after evaluating several options, including OpenAI and CamemBERT embeddings. The preferred method is OpenAI embeddings, specifically the Ada 2 model, primarily due to its superior performance in preliminary trials. The main reason for selecting the OpenAI Ada 2 model over CamemBERT is its computational efficiency. CamemBERT, being a larger model, requires significant processing time when run on a CPU, which makes the Ada 2 model more suitable for our needs due to its faster performance on less powerful hardware.

Table 6.1: Comparison of Text Embedding Models

Attribute	**OpenAI Ada-002**	**Sentence-CamemBERT Large**
Model Name	text-embedding-ada-002	sentence-camembert-large
Token Limit	8191 tokens	512 tokens
Embedding Dimension	1536	1024
Model Size	Not specified	335Billion parameter
Computational Efficiency	High (optimized for CPU)	Lower (best with GPU)

Retrieval and Generation Methods

This phase of the project explores various retrieval strategies, including semantic search enhanced by a dedicated reranking step. We employed several large language models (LLMs) for generative capabilities, such as GPT and open-source alternatives like Llama2 and Mistral. Below is a summary of the techniques and configurations used:

Table 6.2: Methods Used for Retrieval and Generation

Embedding Technique	**Document Chunking Strategy (Node Parser)**	**Retrieval Strategy**	**Language Model (LLM)**
OpenAI Embeddings	Simple Node Parser with Chunk Size of 200 and Overlap of 100	Small to big retrieval	GPT-4 and Mistral (GROC) Models
Camembert (By Hugging Face)	Simple Node Parser with Chunk Size of 200 and Overlap of 100	Small to big retrieval	GPT-4 and Mistral (GROC) Models
Mini LM (By Hugging Face)	Simple Node Parser with Chunk Size of 200 and Overlap of 100	Small to big retrieval	GPT-4 and Mistral (GROC) Models

Reranking

For the reranking process, we utilized the `SentenceTransformerRerank` model specifically designed for enhancing retrieval precision. This model, `cross-encoder/ms-marco-MiniLM-L-2-v2`, reranks the top N results to prioritize the most relevant documents, significantly refining the output of our semantic search.

- **Model Used:** `cross-encoder/ms-marco-MiniLM-L-2-v2`
- **Top N:** 10
- **Device:** CPU

Evaluation of LLMs: Relevancy and Faithfulness

The evaluation of large language models (LLMs) in our Retrieval-Augmented Generation (RAG) method is a critical step to ensure the integrity and effectiveness of the generated content. This subsection details the methodology used to evaluate the performance of the LLMs based on their relevancy and faithfulness.

Generation of Benchmark Dataset

The evaluation process begins with the generation of a benchmark dataset consisting of questions and their corresponding answers. This dataset is created using a high-performance LLM, specifically GPT-4 Turbo, to ensure that the generated answers are of high quality and relevance. The steps include:

1. **Question Generation:** Using the internal company documents as input, GPT-4 Turbo is tasked with generating a diverse set of questions that cover the key topics and information contained in the documents.
2. **Answer Generation:** For each generated question, GPT-4 Turbo also produces an answer, aiming to reflect a deep understanding and accurate extraction of information from the input documents.

Comparative Evaluation

Once the benchmark dataset is prepared, the answers generated by the RAG system are compared against the answers provided by GPT-4 Turbo. This comparison aims to assess two main aspects:

1. **Relevancy:** This metric evaluates how relevant the answers generated by the RAG system are in relation to the questions posed. It checks if the answers adequately address the questions based on the information available in the documents.
2. **Faithfulness:** This metric assesses the factual accuracy of the answers generated by the RAG system. It is crucial that the answers not only pertain to the questions but also are factually correct and do not introduce misinformation.

Performance Metrics with OpenAI Embeddings The table below shows the performance of different language models using OpenAI embeddings. The metrics evaluated include relevancy and faithfulness, with response times noted to assess operational efficiency.

LLM	Relevancy	Faithfulness	Number of Questions	Response Time
GPT-4	0.875	1.0	40	9–10 sec
GPT-3	0.825	0.95	40	1–2 sec
Mistral (GROC)	0.7	0.9	20	6–7 sec

Table 6.3: Evaluation results using OpenAI embeddings.

Performance Metrics with Camembert Embeddings Similarly, the following table presents the performance using Camembert embeddings from Hugging Face. The focus was on relevancy and faithfulness without the influence of response time, as it did not vary significantly from OpenAI embeddings.

LLM	Relevancy	Faithfulness	Number of Questions
GPT-4	0.9	0.925	40
Mistral (GROC)	0.625	0.95	20

Table 6.4: Evaluation results using Camembert embeddings.

6.2.2 User Context Agent

This section aims to explain the implementation steps of the user tool which has access to the user database to extract relevant information

Database Creation and Schema

We initiated by constructing a simulated user information database in a structured JSON format, essential for delivering personalized services. The database schema includes:

- **Username** (String)
- **User Index** (if applicable)
- **Machine Blocked Flag** (Boolean)
- **Unsuccessful Login Attempts** (Integer)
- **List of Installed Applications** on the Machine with Status (Blocked/Unblocked)
- **Support Ticket List**, with fields such as:
  - Primary Key (Ticket Identifier)
  - Ticket Opening DateTime
  - Ticket Closure DateTime (if closed)
  - Message DateTime
  - Sender's Name/Identifier (String)
  - Message Content
  - Closure Cause Status (String, if possible)
  - Progress Status in a Process (String, if possible)

Critical User Issue Detection Attributes are designed to identify critical issues, requiring swift responses. A chat engine enhanced with GPT-4o is integrated to manage urgent concerns effectively, with specialized prompts to ensure appropriate actions.

Embedding Technique and Storage

For effective data handling and retrieval, OpenAI embeddings were selected due to their superior performance in capturing semantic meanings. The vectorized data is stored locally using the Llama Index's storage context, which facilitates the local management of a vector database, allowing for efficient retrieval-augmented generation (RAG) operations directly on the user's machine.

Retrieval Mechanism and Chat Engine Integration

Given the complex nature of user interactions and the need for contextual understanding, a sophisticated retrieval mechanism was implemented. Beyond a basic search engine, a chat engine was integrated to provide the large language model (LLM) with greater flexibility in handling conversational queries. This setup ensures that the system can interpret and respond to user contexts dynamically

Reranking Integration

For the reranking process, we utilized the SentenceTransformerRerank model specifically designed for enhancing retrieval precision. This model, cross-encoder/ms-marco-MiniLM-L-2-v2, reranks the top N results to prioritize the most relevant documents, significantly refining the output of our semantic search.

Prompt Integration

Strategically crafted prompts guide the LLM's interactions, focusing on relevance and appropriateness, particularly for critical issues.

6.2.3 Execute Command Tool

The Execute Command Tool is a streamlined function within the multi-agent system designed to execute operational commands directed by the Meta Agent. This tool acts upon user requests that require system interactions, such as resetting passwords, updating configurations, or initiating system diagnostics.

Functionality

When activated, this tool receives command directives from the Meta Agent, which discerns the need for action based on user inputs and system context. The Execute Command Tool is capable of:

- Processing system commands securely and efficiently.
- Returning feedback to the Meta Agent about the success or failure of executed commands.

Implementation

The development of the Execute Command Tool begins with defining its primary function within the system's architecture. This involves establishing the logical framework necessary to handle and execute operational commands as directed by the Meta Agent. Once the basic function is established, it undergoes a transformation into a 'function tool' by employing a specific conversion method from the Llama Index framework. This transformation equips the tool with advanced capabilities, enabling it to:

- Directly interact with the large language model (LLM) for dynamic command execution.
- Seamlessly integrate within the multi-agent system to ensure coherent operational workflows.
- Enhance responsiveness and reliability in executing system commands.

6.2.4 Conversation Saving Tool

The Conversation Saving Tool is essential for maintaining a record of all interactions handled by the system. This tool captures and stores the details of each conversation, aiding in compliance, training, and system improvement efforts.

Functionality

This tool is designed to:

- Automatically log all dialogues between users and the system.
- Store conversation data in a structured format that includes timestamps, user IDs, and the content of interactions.

Implementation

The initial setup for the Conversation Saving Tool involves programming the essential functionality to capture and store interaction data systematically within the system. This foundational function is crafted to automatically log all dialogues, incorporating detailed metadata such as timestamps and user identifiers. Following its basic configuration, this function is transformed into a 'function tool' through a conversion process facilitated by the Llama Index framework. This enhancement allows the tool to:

- Interface effectively with the large language model (LLM) to utilize contextual data from user interactions.
- Integrate smoothly into the multi-agent framework, aligning with the data retention and retrieval protocols.
- Improve the system's capability to access and analyze historical interaction data for continuous learning and system refinement.

6.2.5 Meta Agent Implementation

Having established the foundational tools—User Context Agent, Documentation Tool, Conversation Saving Tool, and Execution Command Tool—we are now positioned to integrate these components into the Meta Agent. This section details the implementation of the Meta Agent, which orchestrates interactions between the user queries and our multi-agent system.

Integration with OpenAI's GPT-4o Model

The core of the Meta Agent's functionality is driven by OpenAI's GPT-4o model, chosen for its exceptional speed and performance metrics. The integration process involves several key steps:

Model Selection and Integration Using Langfuse

The selection of the GPT-4o model from the Llama Index's suite of tools was made due to its cutting-edge capabilities in understanding and generating human-like text, which are critical for handling complex user interactions in our system. To ensure the optimal selection of the language model for our multi-agent system, we utilized Langfuse, a comprehensive LLM Engineering Platform, to conduct a comparative analysis among several leading models, including GPT-4, GPT-4o, Llama3, and GPT-3.5.

Comparative Analysis of Models Using Langfuse

Langfuse provided the necessary tools to systematically track and analyze various performance metrics such as latency and cost efficiency of each model. Here's how Langfuse facilitated the comparison:

1. **Latency Tracking:** We used Langfuse to measure the response times of each model, providing a clear benchmark of their performance under operational conditions.

   Our findings, detailed through Langfuse's comprehensive analytics, revealed that GPT-4o offers the best trade-off between response time and cost, while maintaining high standards of performance. These results are visually represented in the comparative latency chart below, which underscores the performance advantages of GPT-4o over its counterparts.

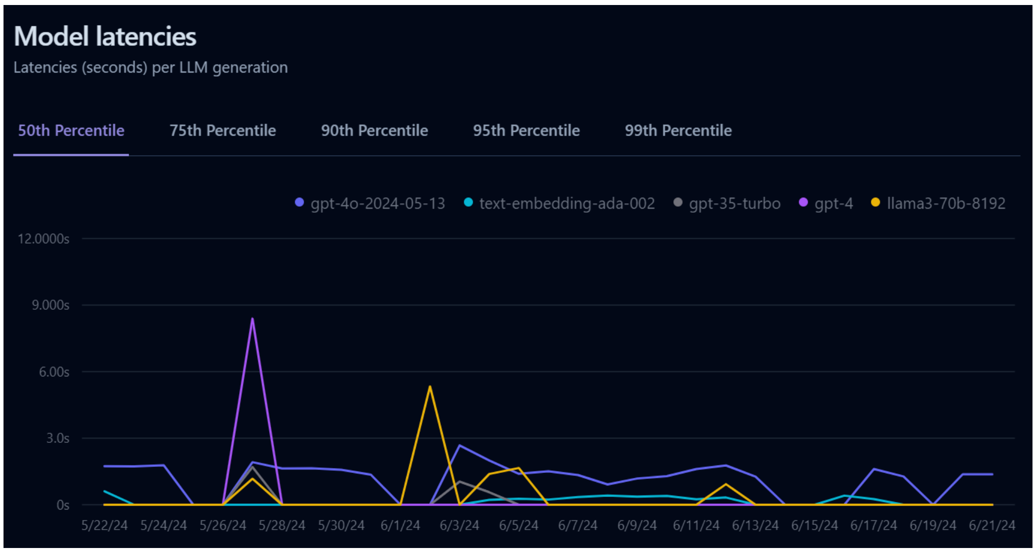


Figure 6.7: Latency Comparison Among Selected Models Facilitated by Langfuse

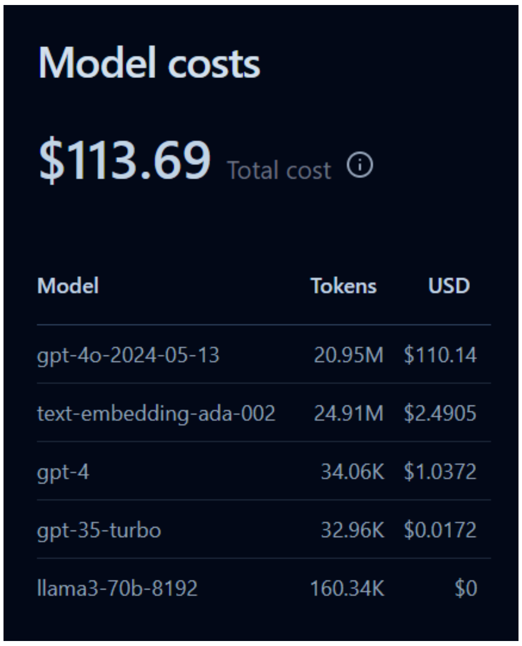
2. **Cost Analysis Using Langfuse**

   We utilized Langfuse to perform a detailed cost analysis for each model considered. The following table presents the cost per million tokens for input and output processes of each model, providing a clear comparative view of their cost efficiency.

Table 6.5: Cost Comparison of Language Models

Model	Cost per 1M Tokens (Input)	Cost per 1M Tokens (Output)
GPT-4o	$5.00	$15.00
GPT-3.5 Turbo (0125)	$0.50	$1.50
GPT-4	$30.00	$60.00
Llama3	Free	Free

This cost analysis highlights the economic efficiency of using GPT-4o and its variants, which offer significant cost savings for both input and output operations compared to GPT-4, while maintaining high standards of performance. Although Llama3 is a free model and presents an appealing option for projects with limited budget constraints, our comparative analysis raised concerns regarding its stability and performance consistency. The performance of Llama3 is not always guaranteed, with significant variations that could impact service quality. This inconsistency led us to favor GPT-4o, which provides more reliable and stable performance across various operational conditions. This decision underscores our commitment to delivering a dependable and efficient service to our users, where the slightly higher expense is justified by superior consistency and user satisfaction.


Model costs

$113.69 Total cost

Model	Tokens	USD
gpt-4o-2024-05-13	20.95M	$110.14
text-embedding-ada-002	24.91M	$2.4905
gpt-4	34.06K	$1.0372
gpt-35-turbo	32.96K	$0.0172
llama3-70b-8192	160.34K	$0

Figure 6.8: The total cost of the usage during the testing for a mounth

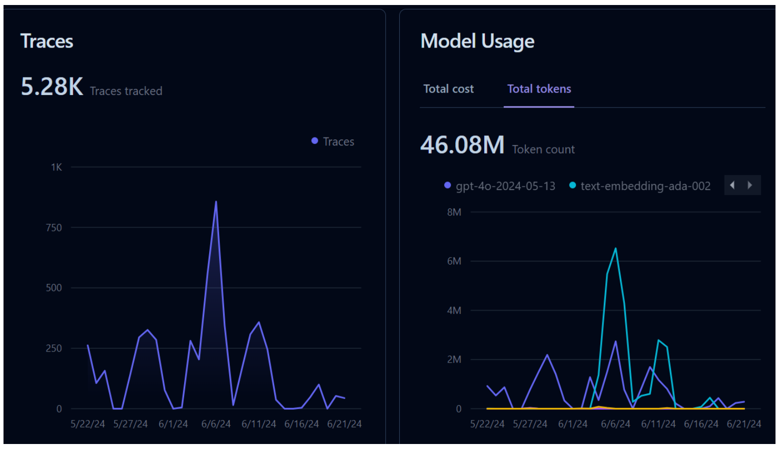


Figure 6.9: Total tokens during testing

Integration and Configuration

Following the selection of GPT-4o, we integrated the model into our system's existing infrastructure. This process involved configuring API endpoints for seamless communication, ensuring data security through robust encryption and access controls, and tuning performance parameters to optimize cost and efficiency.

System Testing and Validation

With the integration complete, we conducted extensive testing to validate GPT-4o's performance. Langfuse's tracing and logging features were instrumental in monitoring the model's responses in real-time, allowing us to refine its functionality based on empirical data.

By leveraging Langfuse for a detailed comparative analysis and utilizing its tools for ongoing monitoring and optimization, we have ensured that our system not only selects the most appropriate model but also maintains high performance standards to meet the evolving demands of our users.

Prompt Refinement

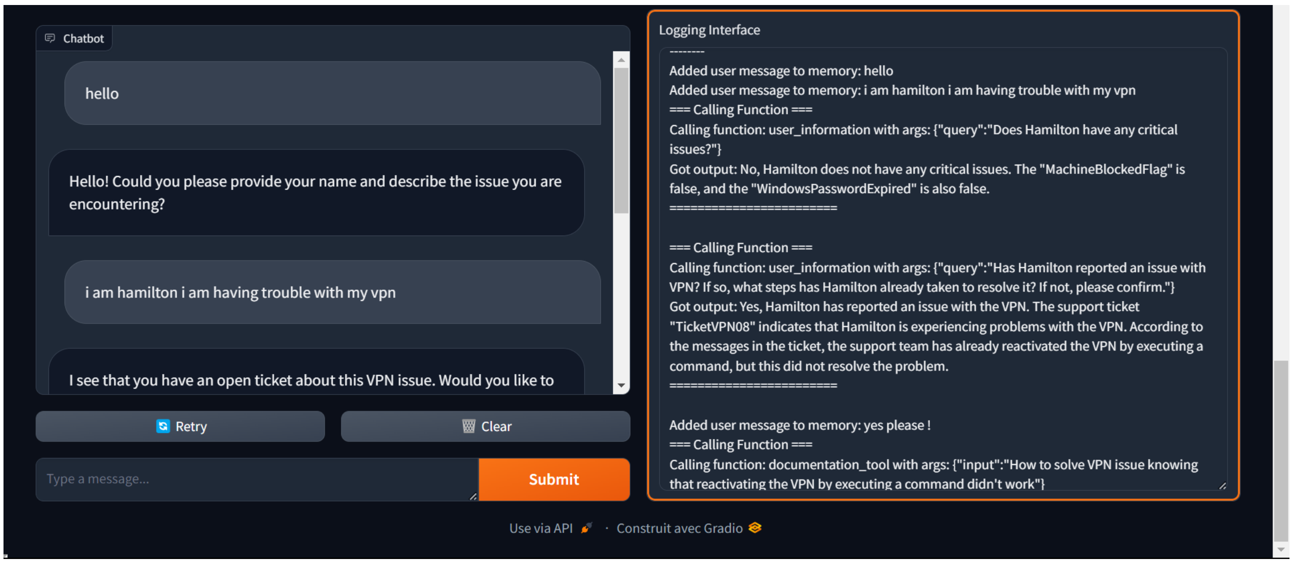
To tailor the behavior of the GPT-4o model to our specific needs, we engaged in prompt engineering. This process involved crafting detailed prompt instructions that guide the model to respond in a manner aligned with our objectives for user interaction. The prompts were designed to leverage the model's capabilities to interpret context, make informed decisions, and generate responses that are not only relevant but also concise and user-friendly.

Behavioral Tuning

The prompts were iteratively refined based on trial interactions to perfect the model's responses. We monitored the model's performance closely, adjusting the nuances of the prompts to better capture the desired behavior and ensure that the Meta Agent acts in full compliance with our operational guidelines and user expectations.

6.2.6 Testing and Demonstrations Using Gradio Interface

For testing and demonstrating the capabilities of the Meta Agent, we utilized the Gradio interface. This platform enabled us to create a user-friendly interface for real-time interaction with the Meta Agent, facilitating easy demonstration and rapid prototyping,We have also integrated a changelog to transparently demonstrate the background processes and track the tools the agent is utilizing.. Gradio helped us simulate realistic user scenarios to evaluate the agent's ability to handle diverse and complex requests accurately and efficiently.



hello

Hello! Could you please provide your name and describe the issue you are encountering?

i am hamilton i am having trouble with my vpn

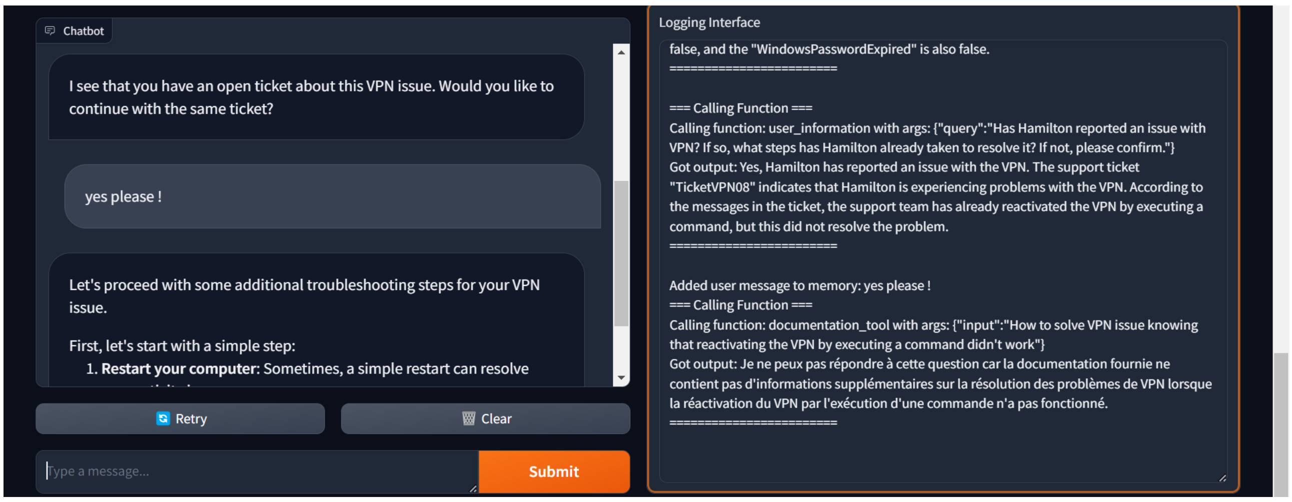
I see that you have an open ticket about this VPN issue. Would you like to

```
--------
Added user message to memory: hello
Added user message to memory: i am hamilton i am having trouble with my vpn
=== Calling Function ===
Calling function: user_information with args: {"query":"Does Hamilton have any critical
issues?"}
Got output: No, Hamilton does not have any critical issues. The "MachineBlockedFlag" is
false, and the "WindowsPasswordExpired" is also false.
======================

=== Calling Function ===
Calling function: user_information with args: {"query":"Has Hamilton reported an issue with
VPN? If so, what steps has Hamilton already taken to resolve it? If not, please confirm."}
Got output: Yes, Hamilton has reported an issue with the VPN. The support ticket
"TicketVPN08" indicates that Hamilton is experiencing problems with the VPN. According to
the messages in the ticket, the support team has already reactivated the VPN by executing a
command, but this did not resolve the problem.
======================

Added user message to memory: yes please !
=== Calling Function ===
Calling function: documentation_tool with args: {"input":"How to solve VPN issue knowing
that reactivating the VPN by executing a command didn't work"}
```

Figure 6.10: Conversation initialization



I see that you have an open ticket about this VPN issue. Would you like to continue with the same ticket?

yes please !

Let's proceed with some additional troubleshooting steps for your VPN issue.

First, let's start with a simple step:

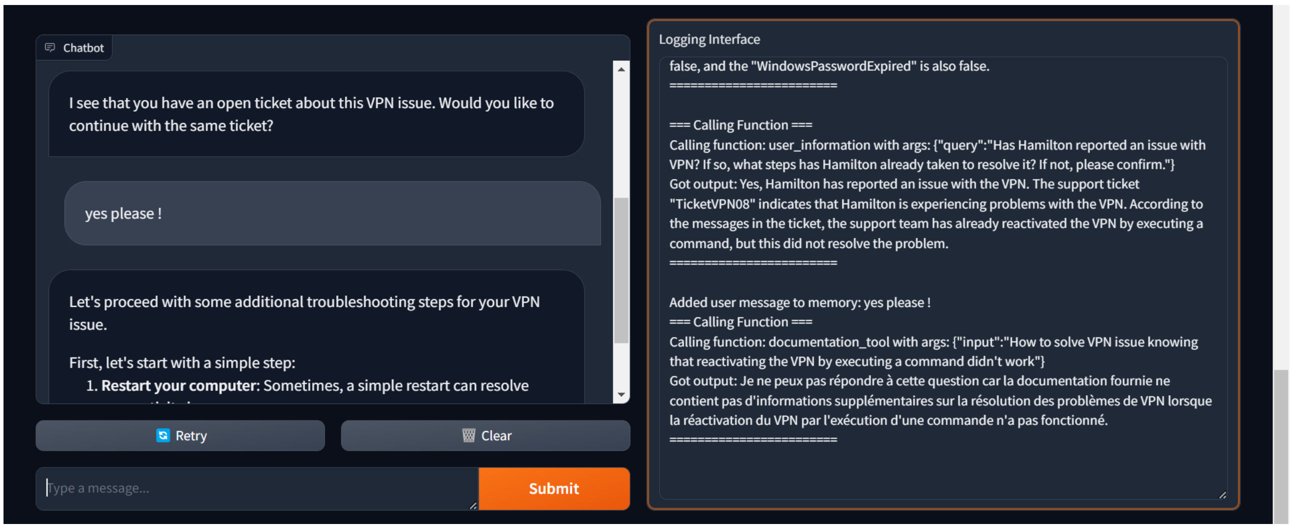
1. **Restart your computer:** Sometimes, a simple restart can resolve

```
false, and the "WindowsPasswordExpired" is also false.
======================

=== Calling Function ===
Calling function: user_information with args: {"query":"Has Hamilton reported an issue with
VPN? If so, what steps has Hamilton already taken to resolve it? If not, please confirm."}
Got output: Yes, Hamilton has reported an issue with the VPN. The support ticket
"TicketVPN08" indicates that Hamilton is experiencing problems with the VPN. According to
the messages in the ticket, the support team has already reactivated the VPN by executing a
command, but this did not resolve the problem.
======================

Added user message to memory: yes please !
=== Calling Function ===
Calling function: documentation_tool with args: {"input":"How to solve VPN issue knowing
that reactivating the VPN by executing a command didn't work"}
Got output: Je ne peux pas répondre à cette question car la documentation fournie ne
contient pas d'informations supplémentaires sur la résolution des problèmes de VPN lorsque
la réactivation du VPN par l'exécution d'une commande n'a pas fonctionné.
======================
```

Figure 6.11: Looking for critical or existing issues.



I see that you have an open ticket about this VPN issue. Would you like to continue with the same ticket?

yes please !

Let's proceed with some additional troubleshooting steps for your VPN issue.

First, let's start with a simple step:

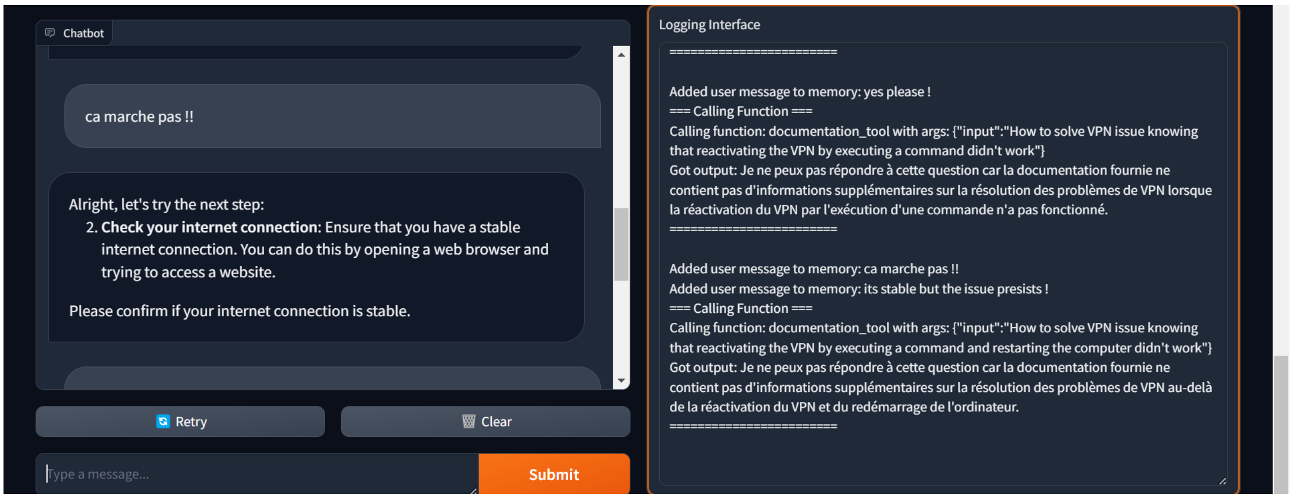
1. **Restart your computer:** Sometimes, a simple restart can resolve

```
false, and the "WindowsPasswordExpired" is also false.
=======================

=== Calling Function ===
Calling function: user_information with args: {"query":"Has Hamilton reported an issue with VPN? If so, what steps has Hamilton already taken to resolve it? If not, please confirm."}
Got output: Yes, Hamilton has reported an issue with the VPN. The support ticket "TicketVPN08" indicates that Hamilton is experiencing problems with the VPN. According to the messages in the ticket, the support team has already reactivated the VPN by executing a command, but this did not resolve the problem.
=======================

Added user message to memory: yes please !
=== Calling Function ===
Calling function: documentation_tool with args: {"input":"How to solve VPN issue knowing that reactivating the VPN by executing a command didn't work"}
Got output: Je ne peux pas répondre à cette question car la documentation fournie ne contient pas d'informations supplémentaires sur la résolution des problèmes de VPN lorsque la réactivation du VPN par l'exécution d'une commande n'a pas fonctionné.
=======================
```

Figure 6.12: Proposing Solutions



ca marche pas !!

Alright, let's try the next step:

2. **Check your internet connection:** Ensure that you have a stable internet connection. You can do this by opening a web browser and trying to access a website.

Please confirm if your internet connection is stable.

```
=======================

Added user message to memory: yes please !
=== Calling Function ===
Calling function: documentation_tool with args: {"input":"How to solve VPN issue knowing that reactivating the VPN by executing a command didn't work"}
Got output: Je ne peux pas répondre à cette question car la documentation fournie ne contient pas d'informations supplémentaires sur la résolution des problèmes de VPN lorsque la réactivation du VPN par l'exécution d'une commande n'a pas fonctionné.
=======================

Added user message to memory: ca marche pas !!
Added user message to memory: its stable but the issue presists !
=== Calling Function ===
Calling function: documentation_tool with args: {"input":"How to solve VPN issue knowing that reactivating the VPN by executing a command and restarting the computer didn't work"}
Got output: Je ne peux pas répondre à cette question car la documentation fournie ne contient pas d'informations supplémentaires sur la résolution des problèmes de VPN au-delà de la réactivation du VPN et du redémarrage de l'ordinateur.
=======================
```

Figure 6.13: Proposing other solutions

6.3 Continuous Improvement

Following the initial tests, the Meta Agent enters a continuous cycle of performance evaluation and enhancement. Utilizing feedback from these interactions and system performance metrics, we periodically refine the model's prompts and adjust operational parameters. This iterative improvement ensures that the Meta Agent remains effective as user needs evolve and new challenges arise.

This section demonstrates the critical role of the Meta Agent in our multi-agent system, highlighting the technical sophistication and adaptability required to manage and direct complex automated interactions in client services. Through the use of advanced AI models and strategic implementation, the Meta Agent stands as a cornerstone of our system, ensuring high-quality, efficient service delivery.

Furthermore, the development team is actively working on the production phase , of our solutions. These efforts have already culminated in the successful sale of the system to a client, marking a significant milestone in the project's commercial deployment.

General conclusion

This thesis explores the practical application of Large Language Models (LLMs) in enhancing customer service operations, focusing on a project undertaken at Voxist for one of its clients. It addresses the significant challenges LLMs face, such as handling real-time data, managing sensitive information securely, and maintaining interaction context.

The proposed multi-agent chatbot system, now in the production phase by the development team, aims to mitigate these challenges by improving interaction quality, response accuracy, and operational efficiency. Through a detailed examination of foundational concepts in machine learning and the evolution of LLMs, the thesis lays a solid theoretical groundwork. It then transitions into practical insights, detailing Voxist's application of LLMs, the challenges encountered, and the strategies developed to overcome these obstacles.

In the course of this thesis, we explored the limitations of frameworks like Llama Index and conducted a comparative analysis between OpenAI models and open-source models such as Llama 3 and Mistral. This comparison provided a solid understanding of these models' strengths and weaknesses. Additionally, we delved into vector databases, prompt engineering, and compared multiple embeddings, enhancing the chatbot system's capabilities.

By leveraging advanced techniques like Retrieval-Augmented Generation (RAG) systems and focusing on prompt engineering, the thesis identifies effective methods to enhance the capabilities of LLMs. The design and implementation of the multi-agent system are thoroughly discussed, highlighting how each component contributes to enhanced client services.

This thesis demonstrates that with the right framework, the integration of LLMs into existing systems can be simplified, allowing for more personalized and effective customer interactions. The thesis underscores the potential of advanced AI technologies to transform customer service operations by making them more scalable and adaptable to complex user interactions and sensitive data management.

Bibliography

[1] Otterly AI. Title or description of the photo, 2023.

[2] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. A survey on hallucination in large language models. *Journal Name Here*, 2023.

[3] Ines Almeida. Improving the reasoning capabilities of large language models: Chain-of-thought prompting. https://www.nownextlater.ai/Insights/post/Improving-the-Reasoning-Capabilities-of-Large-Language-Models, 8 2023. Accessed: 2023-08-08.

[4] Ben Lorica. Best practices in retrieval augmented generation. https://gradientflow.substack.com/p/best-practices-in-retrieval-augmented, 2023.

[5] Towards Data Science. Rag vs finetuning — which is the best tool to boost your llm application? https://towardsdatascience.com/rag-vs-finetuning-which-is-the-best-tool-to-boost-your-llm-application-94654b1eaba7, 2023.

[6] Leo Pauly, Harriet Peel, Shan Luo, and Raul Fuentes. Deeper networks for pavement crack detection.

[7] Sagar Sharma. Activation functions in neural networks: Sigmoid, tanh, softmax, relu, leaky relu explained!!! https://towardsdatascience.com/activation-functions-in-neural-networks-1c53fdb8df08, 2017.

[8] O'Reilly Media. Intelligent projects using python: 9 real-world ai projects leveraging machine learning and deep learning with tensorflow and keras. https://www.oreilly.com/library/view/intelligent-projects-using/9781788996921/0cce5e73-bfa7-4219-97dd-19a6b7d40d35.xhtml, 2019. Image reference.

[9] DSA. Introduction to recurrent neural network. https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/.

[10] Krzysztof Zarzycki and Maciej Ławryńczuk. Lstm and gru neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[12] Shervin Minaee et al. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024. [cs.CL].

[13] Roie Schwaber-Cohen. What is a vector database & how does it work? use cases + examples. https://www.pinecone.io/learn/vector-database/, 2024.

[14] Advanced rag techniques: an illustrated overview. https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6.

[15] Leonie Monigatti. Advanced retrieval-augmented generation: From theory to llamaindex implementation. *Towards Data Science*.

[16] Michael Paluszek and Stephanie Thomas. What is deep learning, 2020.

[17] Michael Paluszek and Stephanie Thomas. What is deep learning, 2020.

[18] Michael Paluszek and Stephanie Thomas. What is deep learning, 2020.

[19] Nicholas G. Polson and Vadim Sokolov. Deep learning, 2018.

[20] Nicholas G. Polson and Vadim Sokolov. Deep learning, 2018.

[21] Activation functions: Comparison of trends in practice and research for deep learning, 2021.

[22] A survey on activation functions and their effects on deep learning, 2022.

[23] Optimization algorithms for stability and convergence in natural language processing using deep learning algorithms, 2023.

[24] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. 3rd ed. draft edition, 2021. Retrieved from https://web.stanford.edu/ jurafsky/slp3/.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[26] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *International conference on machine learning*, pages 1310–1318, 2013.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Q. Zhang and Y. Wang. When large language models meet citation: A survey. *Semantic Scholar*, 2023.

[29] Another N. Author. Transformative approaches in nlp using llms. In *Proceedings of the 2023 Conference of the Association for Computational Linguistics*, 2023.

[30] A. N. Author. Exploring the capabilities of llms in nlp. *arXiv*, 2023.

[31] National Institutes of Health. Advancements in biomedical nlp using llms. *PubMed Central*, 2023.

[32] Open Reviewer. Deep insights into llms: A review. *OpenReview*, 2023.

[33] Large language model applications. https://aloa.co/blog/large-language-model-applications, 2022.

[34] Large language model apps. https://indatalabs.com/blog/large-language-model-apps, 2022.

[35] What are large language models used for? https://blogs.nvidia.com/blog/what-are-large-language-models-used-for/, 2022.

[36] Renu Khandelwal. Vector database: Empowering next-gen applications – unlocking the future of efficient data retrieval in an ai-driven world. https://medium.com/GoPenAI/vector-database-empowering-next-gen-applications-123456789, Sep 2023.

[37] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *Department of Computer Science And Engineering, Indian Institute of Technology Patna and Stanford University and Amazon AI*, 2023. Emails: {pranab_2021cs25, ayush_2211ai27, sriparna, samrat}@iitp.ac.in, hi@vinija.ai, hi@aman.ai.

[38] Ph.D. Cameron R. Wolfe. Chain of thought prompting for llms: A practical and simple approach for "reasoning" with llms, 2023.

[39] Saumajit Saha. Prompt engineering techniques — brief survey, 2024.

[40] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*, 2024.

[41] IBM Research. Retrieval-augmented generation (rag). https://research.ibm.com/blog/retrieval-augmented-generation-RAG, 2024. Accessed: [insert today's date].

[42] Retrieval augmented generation (rag) for llms. https://www.promptingguide.ai/research/rag, 2024. Accessed: [insert today's date].

[43] L. Monigatti. Amélioration de la performance de récupération dans les pipelines rag avec la recherche hybride. *Towards Data Science*, 2023. Accessed: [insert today's date].

[44] Query transformations. https://blog.langchain.dev/query-transformations/, Oct 2023.

[45] Metrics. https://docs.ragas.io/en/stable/concepts/metrics/index.html.

[46] Ravi Theja. Evaluate rag with llamaindex. https://cookbook.openai.com/examples/evaluation/evaluate_rag_with_llamaindex, Nov 2023.

[47] Tula Masterman, Sandi Besen, Mason Sawtell, and Alex Chao. The landscape of emerging ai agent architectures for reasoning, planning, and tool calling: A survey. *Neudesic, an IBM Company*, 2023.

[48] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. 2023.

[49] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberative reasoning in language models. 2023.

[50] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. 2023.

[51] Ziyue Li et al. Title of the paper. *Journal Name*, 2023.

[52] Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Guoqing Du, Shiwei Shi, Hangyu Mao, Ziyue Li, Xingyu Zeng, and Rui Zhao. Title of the paper. *Journal Name*, 2023.

[53] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[54] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning. 2023.

[55] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktaschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Journal Name*, 2021.

[56] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. 2023.

[57] Weize et al. Chen. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. *arXiv preprint arXiv:2308.10848*, 2023.