Università degli studi di Padova
Department of Information Engineering
Master Degree in Computer Engineering

Development and Implementation of a Retrieval-Augmented Generation System: The Lookinglass

Supervisor
Prof. LORIS NANNI

Graduating Student
SEYED ATA OLAH SEYED JAFARI
Matricola: 2043891

Academic Year 2023/2024

Graduation Date 11/07/2024

To my wonderful family: nurturing mom, steadfast father, and spirited little brother.

Confidentiality Statement

This thesis project was conducted as part of an internship at Impssbl Corp. The research, development, and results presented in this document are based on work completed during my time at the company. The content herein is confidential and intended solely for academic and professional evaluation.

Contents

1. Introduction **17**

1.1. Importance of data retrieval and generation . . . 17
1.2. Rise of Large Language Models (LLMS) . . . 17
1.3. Limitations of LLMs in Data Retrieval and Generation . . . 18
1.4. Retrieval-Augmented Generation Systems . . . 18
1.5. Objectives and Scope of the Study . . . 19
1.5.1. Objectives . . . 20
1.5.2. Scope . . . 20
1.6. Structure of the Thesis . . . 21

2. Literature Review **23**

2.1. Large Language Models(LLMs) . . . 23
2.1.1. Transformers . . . 23
2.1.2. BERT and Its Impact . . . 24
2.1.3. GPT and Its Impact . . . 25
2.2. Retrieval-Augmented Generation (RAG) . . . 26
2.2.1. RAG vs LLM . . . 27
2.2.2. RAG concept and Structure . . . 27
2.3. Challenges with RAG . . . 31
2.4. Domain-Specific Applications of RAG Systems . . . 32
2.4.1. Healthcare Applications . . . 32
2.4.2. Business Applications . . . 34
2.4.3. Educational Applications . . . 35
2.5. Evaluation Metrics and Validation of RAG Systems . . . 36
2.5.1. Evaluation Metrics . . . 36
2.5.2. Validation Techniques . . . 37
2.6. Gaps in the Literature and Justification for the Lookinglass . . . 38
2.6.1. Identified Gaps . . . 38
2.6.2. Justification for the Lookinglass . . . 38
2.7. Summary . . . 39

3. Methodology **41**

3.1. Primary Goals and Objectives . . . 41
3.2. Requirments . . . 41
3.2.1. Functional Requirements . . . 42
3.2.2. Non-Functional Requirements . . . 42
3.3. System Architecture . . . 42
3.3.1. Data Extraction . . . 43
3.3.2. Data Upsert . . . 45
3.3.3. Query . . . 51
3.4. Visual Overview of the Lookinglass . . . 60
3.4.1. Upload Documents . . . 60
3.4.2. Query Documents . . . 62
3.5. Summary . . . 62

4. Experiments and Results **65**
4.1. Experimental Setup 65
4.1.1. Hardware and Software Environment 65
4.1.2. Datasets 65
4.1.3. Types of Queries 66
4.1.4. Measurement and Recording of Results 66
4.1.5. Steps to Ensure Reproducibility 67
4.2. Precision Results 68
4.3. Summary 70

5. Conclusions **71**
5.1. Future work 71
5.2. Conclusion 72

A. Appendix **73**

Bibliography **75**

List of Figures

2.1. Overall pre-training and fine-tuning procedures for BERT [1] 24
2.2. Zero-shot, one-shot and few-shot [2] 26
2.3. A generic RAG architecture. The user queries, spanning different modalities, serve as input to both the retriever and the generator. The retriever extracts relevant information from data sources. The generator interacts with the retrieval results and ultimately produces outcomes of various modalities. [3] 28
2.4. Taxonomy of RAG Enhancements.[3] 29
2.5. Overview of the MedWriter[4] 33
2.6. Proposed RAG generates responses to math student queries [5] 35

3.1. Lookinglass Architecture 43
3.2. Data Extraction for Unstructured data 44
3.3. Data Extraction for Structured data 45
3.4. Data Upsert for Unstructured data 45
3.5. MapReduce Technique for Summarization 47
3.6. Data Upsert for Structured Data 49
3.7. Query Module for Unstructured Data 51
3.8. Query Module for Structured Data 57
3.9. Upload Page on Lookinglass System 61
3.10. Status of a document after being uploaded 61
3.11. Query page on Lookinglass System 62

4.1. Precision for Each Question Type Across 5 Trials for Unstructured Data According to 4.3 69
4.2. Precision for Each Question Type Across 5 Trials for Structured Data According to 4.4 69

A.1. Response to a query from structured data, including the visualization diagram and the sources from which the data was extracted. 74
A.2. Response to a query from structured data, including the visualization diagram and the sources from which the data was extracted. 74

List of Tables

4.1. Document Types 66
4.2. Number of Questions for Each Type 67
4.3. Precision for Each Question Type Across 5 Trials for Unstructured Data . 68
4.4. Precision for Each Question Type Across 5 Trials for Structured Data . . 69

Sommario

Questa tesi presenta lo sviluppo e l'implementazione del sistema Lookinglass, un sistema di Generazione Aumentata dal Recupero (RAG) innovativo, progettato per potenziare le capacità dei modelli linguistici integrando avanzati meccanismi di recupero delle informazioni. I modelli linguistici tradizionali, pur essendo notevolmente avanzati nella generazione di contenuti, spesso incontrano difficoltà nel garantire l'accuratezza fattuale e nella gestione di query complesse e multi-step. Per superare queste limitazioni, il sistema Lookinglass incorpora un framework robusto che assicura alta precisione e affidabilità nella generazione delle risposte.

L'architettura del sistema è progettata meticolosamente per gestire in modo efficiente sia dati strutturati che non strutturati, sfruttando tecniche all'avanguardia di elaborazione del linguaggio naturale e database vettoriali per migliorare la gestione delle query. Tra le caratteristiche principali figurano la sintesi avanzata, l'efficace pre-elaborazione dei dati e la gestione sicura dei documenti, tutti elementi che contribuiscono alla sua performance superiore in applicazioni aziendali come la risposta a domande basate su documenti e la generazione di contenuti.

Attraverso test e sperimentazioni approfondite, il sistema Lookinglass ha dimostrato significativi miglioramenti in termini di accuratezza e rilevanza, specialmente in domini ad alta intensità di conoscenze. La sua capacità di fornire risposte dettagliate e precise, unitamente a forti misure di privacy dei dati, lo rende uno strumento potente per decisioni basate sui dati. Questa tesi sottolinea l'applicabilità del sistema in scenari reali e suggerisce direzioni per futuri miglioramenti, tra cui l'augmentazione dei dati in tempo reale e la funzionalità di agenti automatici.

Abstract

This thesis presents the development and implementation of the Lookinglass Retrieval-Augmented Generation (RAG) system, a novel approach designed to enhance the capabilities of language models by integrating advanced information retrieval mechanisms. Traditional language models, despite their impressive generative capabilities, often struggle with factual inaccuracies and the handling of complex, multi-hop queries. To address these limitations, the Lookinglass system incorporates a robust framework that ensures high precision and reliability in response generation.

The system's architecture is meticulously designed to manage both structured and unstructured data efficiently, leveraging state-of-the-art natural language processing techniques and vector databases for enhanced query handling. Key features include advanced summarization, effective data preprocessing, and secure document management, all of which contribute to its superior performance in business applications such as document-based question answering and content generation.

Through comprehensive testing and experimentation, the Lookinglass system has demonstrated significant improvements in accuracy and relevance, particularly in knowledge-intensive domains. Its ability to provide detailed and precise answers, coupled with strong data privacy measures, makes it a powerful tool for data-driven decision-making. This thesis underscores the system's applicability in real-world scenarios and suggests directions for future enhancements, including real-time data augmentation and automated agent functionality.

1

Introduction

In this chapter, a brief introduction to Retrieval-Augmented Generation (RAG) systems, their significance, primary applications, and associated challenges will be presented.

1.1 Importance of data retrieval and generation

In today's data-driven business landscape, the ability to effectively retrieve and generate relevant information is essential for strategic decision-making. As organizations accumulate massive volumes of structured and unstructured data from various sources, the challenge lies in identifying and extracting specific data points that address unique business needs. Efficient data retrieval systems play a crucial role in quickly accessing the most pertinent information from vast and complex data pools, minimizing time and effort spent on manual data exploration. The true value, however, lies in generating meaningful insights and actionable information from the retrieved data, whether through concise summaries, visualizations, or natural language responses that directly address the posed queries. These capabilities empower decision-makers to identify trends, understand market dynamics, and gain a deeper understanding of business opportunities.

For instance, a financial services company can benefit from advanced data retrieval systems to monitor and analyze market changes, helping adjust portfolios based on new data and maximizing returns. Similarly, healthcare organizations rely on data retrieval to quickly access patient histories and research findings, enabling them to deliver personalized treatment and improve patient outcomes. Thus, data retrieval and generation are the cornerstones of data-driven decision-making, enabling organizations to unlock the full potential of their data assets, uncover hidden patterns, and make informed choices that drive strategic growth and success.

1.2 Rise of Large Language Models (LLMS)

The recent introduction and rapid adoption of Large Language Models (LLMs) mark a transformative leap in how organizations handle and interact with data. These powerful models, such as GPT-3, BERT, and others, leverage deep learning to understand, process, and generate natural language text, ushering in a new era of automated content generation, customer service, and data interpretation. Trained on massive datasets, LLMs have demonstrated an unprecedented ability to generate human-like text, revolutionizing various domains, including business intelligence and decision support systems.

These models exhibit unparalleled versatility in generating coherent and contextually

relevant responses to complex queries. Their ability to comprehend and process vast data, combined with their capacity to generate high-quality text, has sparked interest across industries. For instance, marketing firms can use LLMs to analyze social media sentiment and create targeted campaigns aligned with current consumer trends. Similarly, customer service chatbots powered by LLMs can efficiently handle and respond to customer queries, improving satisfaction and streamlining interactions. As a result, these advanced language models have opened new avenues for enhancing data interaction, automating information retrieval, and improving decision-making processes within organizations.

1.3 Limitations of LLMs in Data Retrieval and Generation

While Large Language Models (LLMs) have transformed natural language processing and generation, their reliance on pre-trained data presents significant challenges in dynamic business environments. LLMs are trained on static datasets and cannot incorporate updated information into their responses. This limitation can lead to outdated or irrelevant answers, particularly in rapidly changing industries like finance and technology. Additionally, LLMs often struggle with specialized queries that require precise data interpretation or domain-specific knowledge. For instance, in healthcare, providing accurate answers about emerging treatments necessitates access to the latest clinical guidelines and research studies—something that static training data cannot accommodate.

Furthermore, LLMs are prone to "hallucination," a phenomenon where they generate plausible-sounding but factually incorrect information due to the absence of updated data. This issue is particularly concerning in data-driven business contexts, where accurate and current information is essential for effective decision-making. Depending on outdated, incomplete, or misleading information can lead to suboptimal resource allocation, strategic missteps, and potential financial losses. These limitations highlight the need to integrate the language understanding and generation capabilities of LLMs with data retrieval mechanisms that can ensure responses are grounded in the most relevant and accurate information available. A Retrieval-Augmented Generation (RAG) system offers a solution by addressing these limitations and enabling organizations to generate contextually appropriate and factually accurate responses based on up-to-date information.

1.4 Retrieval-Augmented Generation Systems

Retrieval-Augmented Generation (RAG) systems have emerged as a promising approach to enhance the capabilities of large language models (LLMs) by combining them with external knowledge retrieval mechanisms. These systems aim to address some of the inherent limitations of LLMs, such as their inability to access up-to-date information beyond their training data and their tendency to generate hallucinated or factually incorrect content when dealing with knowledge-intensive tasks. The core idea behind RAG systems is to augment the generation process of LLMs by incorporating relevant

information retrieved from external knowledge sources. This is typically accomplished through a three-step process: (1) indexing and storing information from external data sources in a searchable format, (2) retrieving relevant chunks of information based on the input query or prompt, and (3) providing the retrieved information as additional context to the language model during the generation process. By integrating external knowledge retrieval into the generation process, RAG systems can potentially overcome some of the limitations of LLMs. First, they can mitigate the issue of LLMs being limited to the knowledge present in their training data by allowing access to up-to-date and domain-specific information from external sources. Second, they can reduce the likelihood of hallucinations or factual errors by grounding the generation process in relevant retrieved information.

The potential benefits of RAG systems include:

1. **Improved factual accuracy**: By leveraging external knowledge sources, RAG systems can generate more factually accurate responses, especially in knowledge-intensive domains where LLMs may struggle due to limited training data.
2. **Access to up-to-date information**: RAG systems can retrieve and incorporate the latest information from external sources, allowing them to provide more current and relevant responses compared to LLMs relying solely on their training data.
3. **Domain adaptability**: RAG systems can be tailored to specific domains by indexing and retrieving information from domain-specific knowledge sources, enabling them to generate more domain-relevant and accurate outputs.
4. **Transparency and interpretability**: By exposing the retrieved information used in the generation process, RAG systems can provide more transparency and interpretability compared to black-box LLM generations, allowing users to verify the sources and validity of the generated content.

While RAG systems offer promising potential, they also face challenges such as efficient and accurate retrieval, effective integration of retrieved information into the generation process, and managing the trade-off between computational complexity and performance.

1.5 Objectives and Scope of the Study

The primary objective of the study is to develop and enhance the Retrieval-Augmented Generation (RAG) system, Lookinglass, aiming to integrate information retrieval mechanisms to significantly improve the accuracy and reliability of responses generated by language models. This research focuses on addressing the limitations of traditional language models by reducing the occurrence of hallucinations, effectively handling complex and multi-hop queries, and ensuring robust privacy measures to protect sensitive information. The scope of the study encompasses the comprehensive design and implementation of system requirements and specifications necessary for a robust RAG system within a business context, ensuring its applicability in various real-world business scenarios such as document-based question answering, information retrieval, and content generation.

1.5.1 Objectives

The primary objective of this thesis is to develop and implement a Retrieval-Augmented Generation (RAG) system, named "The Lookinglass," which addresses the limitations of current Large Language Models (LLMs) in business environments. Specifically, the objectives include:

1. **Enhance Language Model Capabilities:** Integrate advanced retrieval mechanisms to reduce instances of hallucinations, ensuring the language model generates factually accurate and reliable responses.

2. **Improve Handling of Complex Queries:** Develop capabilities to manage long-tail data and multi-hop queries, enabling the system to reason over multiple pieces of supporting evidence for complex business inquiries.

3. **Address Privacy Concerns:** Implement robust data handling and retrieval processes to mitigate the risk of data leakage, ensuring the confidentiality of proprietary and private datasets.

4. **Facilitate Practical Business Applications:** Design the system to be versatile and applicable across various business tasks, including document-based question answering, information retrieval, and content generation.

1.5.2 Scope

The scope of this thesis encompasses the comprehensive design, and development of the Retrieval-Augmented Generation (RAG) system, Lookinglass. It includes a detailed examination of the system architecture, focusing on the critical components necessary for integrating advanced retrieval mechanisms to improve the accuracy and reliability of language model responses. The thesis also addresses the implementation of robust privacy and security measures to protect sensitive information, ensuring the system's compliance with data protection standards.

Additionally, User interface design is another key area, aimed at providing a seamless and intuitive experience for users. Through these detailed explorations, the thesis aims to contribute to the advancement of RAG systems in practical business applications, demonstrating their potential to enhance various business tasks, from document-based question answering to content generation and information retrieval.

1. **System Design and Architecture:** Detailed exploration and documentation of the system architecture for the Lookinglass RAG system, including data extraction, data upsert processes, and the query mechanisms for both structured and unstructured data.

2. **Implementation of Retrieval Mechanisms:**: Development and integration of advanced information retrieval techniques to enhance the accuracy and reliability of responses generated by the language model. This includes designing indexing methods, efficient search algorithms, and seamless data augmentation processes.

3. **Privacy and Security Measures:**: Implementation of robust privacy and security protocols to ensure the confidentiality and protection of sensitive data. This involves developing mechanisms to prevent data leakage, enforce access controls, and utilize encryption methods for data security.

4. **User Interface and Experience Design:**: Designing an intuitive and user-friendly interface for users to interact with the Lookinglass system. This involves creating clear input mechanisms for queries, ensuring easy-to-understand outputs, and enhancing the overall user experience.

1.6 Structure of the Thesis

This thesis is organized into five major chapters, each of which addresses specific aspects of the development and implementation of the Retrieval-Augmented Generation (RAG) system, "The Lookinglass."

Chapter 2: Literature Review The Literature Review chapter provides an overview of previous research related to Retrieval-Augmented Generation systems and their integration with Large Language Models (LLMs). This chapter highlights key advancements, existing challenges, and gaps in current literature that the Lookinglass system seeks to address.

Chapter 3: Methodology The Methodology chapter outlines the design and implementation of the Lookinglass system. It covers the overall system architecture, including data retrieval and processing pipelines, query handling mechanisms, and response generation strategies. The chapter also discusses technical aspects like metadata storage, data chunking, and embedding vector storage.

Chapter 4: Experiments and Results The Experiments and Results chapter presents a thorough evaluation of the Lookinglass system through extensive testing and validation. This chapter describes the experimental setup, including the hardware and software environments, datasets, and types of queries used. It analyzes the system's performance in terms of precision, accuracy, and reliability across various query complexities and data types. The results highlight the system's strengths in delivering accurate and relevant information and identify potential areas for improvement. This evaluation demonstrates the practical applicability of Lookinglass in real-world business scenarios and guides future development efforts.

Chapter 5: Conclusion and Future Work The Conclusion and Future Work chapter summarizes the key findings of the research and provides an analysis of the Lookinglass system's strengths and limitations. The chapter also outlines the potential implications of the system in practical business environments and identifies areas for future improvement and research.

2

Literature Review

The evolution of artificial intelligence (AI) and natural language processing (NLP) has revolutionized data interaction in business environments, driven by advancements in Large Language Models (LLMs) like BERT and GPT. While these models have significantly improved language understanding, their static nature often limits the relevance and accuracy of their responses in dynamic contexts. To overcome these limitations, Retrieval-Augmented Generation (RAG) systems have emerged, combining LLMs with data retrieval mechanisms to enhance decision-making efficiency and information accuracy, which this literature review will explore in depth.

2.1 Large Language Models(LLMs)

The development of large language models has been significantly propelled by the introduction of the Transformer model, detailed in [6], which revolutionized NLP with its self-attention mechanism. Building on this, BERT [1] brought about groundbreaking improvements in NLP by enabling deep bidirectional understanding of text. Following this, OpenAI's GPT series further advanced the field with generative pre-training, particularly with and GPT-3 [2] showcasing exceptional capabilities in text generation. These advancements set the stage for sophisticated systems like Retrieval-Augmented Generation (RAG).

2.1.1 Transformers

The introduction of the Transformer model in [6] marked a significant advancement in sequence transduction models by eliminating the need for recurrence and convolution, which are traditionally employed in neural network architectures for tasks such as machine translation and language modeling. The Transformer relies entirely on self-attention mechanisms, allowing for greater parallelization during training and significantly reducing the time required to reach state-of-the-art performance. For example, the Transformer achieved a BLEU score of 28.4 on the WMT 2014 English-to-German translation task and 41.0 on the English-to-French task, both with much shorter training times compared to previous models. This paradigm shift not only improved the efficiency of training but also demonstrated superior performance by learning global dependencies more effectively than models based on recurrent neural networks or convolutional neural networks [6]. The Transformer model's architecture consists of an encoder-decoder structure that leverages multi-head self-attention and fully connected feed-forward layers.

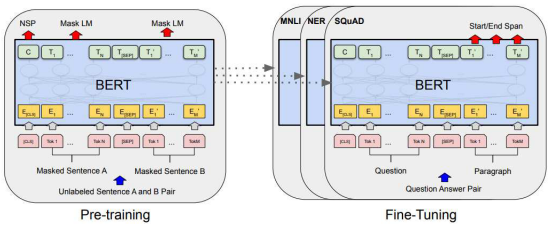

Figure 2.1.: Overall pre-training and fine-tuning procedures for BERT [1]

This design allows each position in the input sequence to attend to all other positions, facilitating the capture of long-range dependencies with constant computational complexity per layer. Additionally, the use of scaled dot-product attention and position-wise feed-forward networks within each encoder and decoder layer ensures efficient processing and high performance across various tasks. The authors report that the Transformer not only surpasses previous state-of-the-art models in translation quality but also proves to be more adaptable to different tasks by simply modifying the input embeddings and positional encodings. These innovations underscore the potential of attention-based models to revolutionize sequence modeling and transduction tasks across diverse domains [6].

2.1.2 BERT and Its Impact

BERT, which stands for Bidirectional Encoder Representations from Transformers, represents a significant breakthrough in the field of natural language processing (NLP) [1]. introduced a novel approach to pre-training language models that significantly improved the performance of a wide range of NLP tasks. BERT's main innovation lies in its use of bidirectional training of Transformer models, which allows it to consider the context from both the left and right sides of a word in all layers. The bidirectional approach enables BERT to capture deeper nuances of language and context, making it more powerful for understanding the complexities of human language.

BERT utilizes two primary tasks during its pre-training phase[1]: the Masked Language Model (MLM) and the Next Sentence Prediction (NSP). The MLM task involves randomly masking some of the tokens in the input and training the model to predict the masked tokens. This allows the model to learn bidirectional representations. The NSP task helps the model understand relationships between sentences by training it to predict if a given sentence B follows sentence A in the original text.

BERT's impact on NLP has been profound. By providing a powerful, pre-trained language representation model, it has reduced the need for task-specific architectures. This generalization capability means that BERT can be fine-tuned for a wide range of tasks, such as question answering, named entity recognition, and natural language inference, without substantial modifications. This has streamlined the process of developing high-performing models for different NLP tasks and has democratized access to advanced language understanding capabilities.

The introduction of BERT has also spurred further research and development in the field, leading to a wave of new models and techniques that build on its success. Its influence is evident in the adoption of similar pre-training and fine-tuning methodologies across various domains within NLP.

2.1.3 GPT and Its Impact

GPT (Generative Pre-trained Transformer) is a groundbreaking language model developed by OpenAI, which has significantly advanced the field of natural language processing (NLP). The core innovation of GPT lies in its transformer-based architecture, which allows for powerful language understanding and generation capabilities. GPT's primary innovation is the transformer architecture, introduced by [6]. Unlike previous models that relied on recurrent neural networks (RNNs), the transformer architecture uses self-attention mechanisms to process input data in parallel, rather than sequentially. This approach allows for the handling of long-range dependencies and the efficient processing of large datasets.

Another crucial innovation in GPT is its unsupervised pre-training approach. GPT is pre-trained on a large corpus of text using a language modeling objective, which involves predicting the next word in a sequence. This pre-training allows the model to learn a wide range of linguistic patterns, grammar, facts, and some degree of common sense knowledge. Once pre-trained, GPT can be fine-tuned on specific tasks with relatively small amounts of task-specific data, making it highly versatile and efficient for various NLP applications.

GPT-3, the third iteration of the GPT model, represents a significant leap in scale and capability. GPT-3 has 175 billion parameters, making it the largest language model ever created at the time of its release [2]. The scaling up of parameters has resulted in substantial improvements in the model's ability to understand and generate human-like text.

The performance of GPT-3 across various NLP tasks is remarkable. It has demonstrated state-of-the-art results in tasks such as translation, question answering, and cloze tasks. One of the most impressive aspects of GPT-3 is its ability to perform few-shot learning, where it can generalize from just a few examples provided in the prompt. This ability to perform tasks with minimal examples showcases the model's deep understanding and flexibility.

One of the key contributions of GPT-3 is its ability to perform few-shot learning[2], which allows the model to understand and execute tasks based on just a few examples provided in the prompt. This is achieved through "in-context learning," where the model uses the context provided by the user to adapt its responses without any gradient updates or fine-tuning.

The impact of few-shot learning is significant because it reduces the need for large task-specific datasets, which are often expensive and time-consuming to obtain. GPT-3's performance in few-shot settings often rivals or exceeds that of models that have been fine-tuned on large amounts of task-specific data. For instance, in tasks like TriviaQA and CoQA, GPT-3 has achieved impressive results with minimal example inputs[2].

Despite its advancements, GPT-3 is not without challenges. The model's reliance on vast amounts of data from the internet means it can inadvertently generate biased or inappropriate content[2]. Furthermore, while GPT-3 performs well on many tasks, it still struggles with tasks requiring deep reasoning, contextual understanding over long documents, and certain types of commonsense reasoning. These limitations highlight the ongoing need for improvements in language models to ensure they can handle a broader

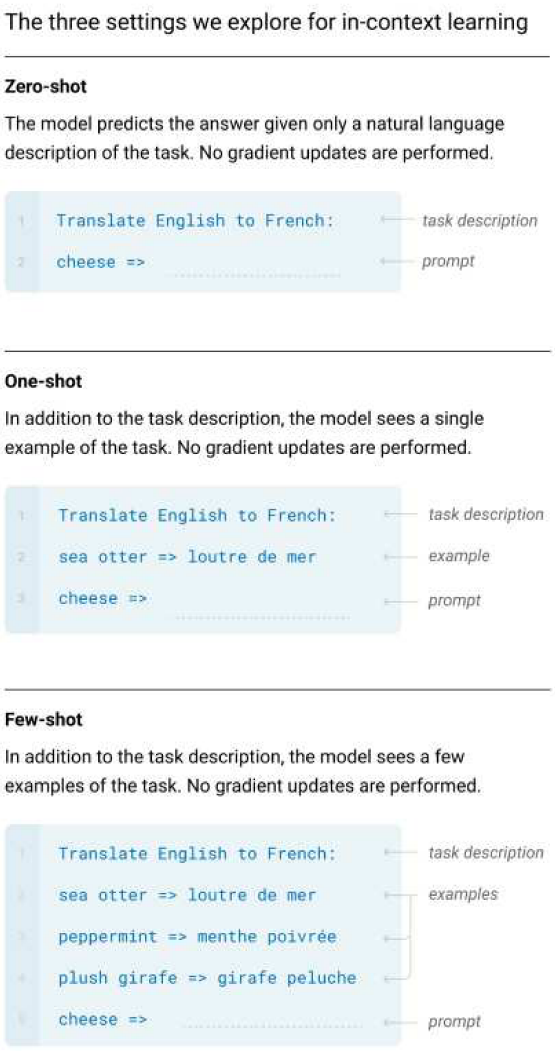

Figure 2.2.: Zero-shot, one-shot and few-shot [2]

range of tasks more accurately and ethically.

The impact of GPT and GPT-3 on NLP has been profound. By providing a robust, pre-trained language model that can be fine-tuned with minimal data, GPT has democratized access to advanced NLP capabilities. This has led to a surge in applications across various domains, from automated customer service and content generation to more sophisticated applications in healthcare and legal document analysis.

2.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) represents a significant advancement in the capabilities of language models by integrating information retrieval mechanisms directly into the generation process. Unlike traditional LLMs, which rely solely on pre-trained data, RAG models dynamically fetch relevant information from external databases during inference, thereby enhancing the accuracy and relevance of their responses. This hybrid approach combines the generative power of models like GPT with the precision

of information retrieval systems, making RAG particularly valuable for tasks requiring up-to-date knowledge and detailed information, such as complex question answering and specialized domain applications.

2.2.1 RAG vs LLM

Retrieval-Augmented Generation (RAG) models have emerged as a significant advancement over traditional Large Language Models (LLMs), particularly in handling knowledge-intensive tasks. Traditional LLMs, such as GPT-3 and BERT, store knowledge implicitly within their parameters, which limits their ability to access and manipulate external, up-to-date information effectively. This limitation is particularly evident in their struggle to provide accurate responses about recent events or rarely mentioned facts [7] [8].

RAG models address these limitations by incorporating a retrieval mechanism that allows the model to access and utilize external knowledge sources dynamically during the generation process. This integration of a dense vector index of a large corpus, such as Wikipedia, enables RAG models to condition their responses on relevant and up-to-date information retrieved from this corpus. As a result, RAG models can provide more accurate and contextually relevant answers compared to traditional LLMs that rely solely on pre-trained knowledge[7].

One of the key advantages of RAG models over traditional LLMs is their ability to generate more specific, diverse, and factual language. By leveraging retrieved passages during the response generation, RAG models significantly reduce the likelihood of generating hallucinations, a common issue with traditional LLMs where the model generates plausible but incorrect or misleading information[7]. This makes RAG models particularly useful in applications requiring high factual accuracy, such as open-domain question answering and fact verification tasks.

Moreover, the hybrid architecture of RAG models, which combines parametric and non-parametric memory, allows for more flexible and interpretable knowledge management. Knowledge in RAG models can be easily updated and expanded by modifying the external corpus, without needing to retrain the entire model. This capability addresses a major challenge faced by traditional LLMs, which require substantial retraining to incorporate new information [8].

For instance, the RAG models fine-tuned for various NLP tasks have consistently outperformed parametric seq2seq models and task-specific retrieve-and-extract architectures, setting new state-of-the-art results in several open-domain QA tasks. This performance improvement is attributed to the effective combination of retrieved context with the model's pre-existing knowledge, enhancing both the accuracy and relevance of the generated responses[7].

2.2.2 RAG concept and Structure

Retrieval-Augmented Generation (RAG) is a hybrid model that integrates the strengths of both parametric and non-parametric memory to enhance language generation. Traditional large language models (LLMs) store knowledge within their parameters, but this approach has limitations in accessing, updating, and verifying the knowledge, leading

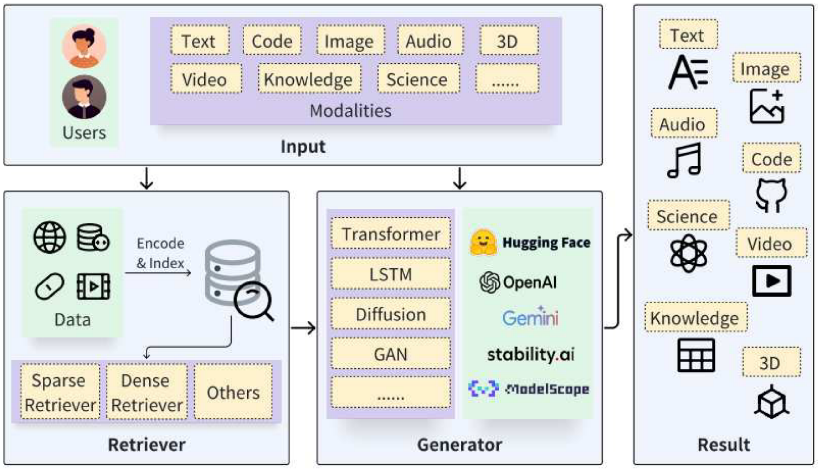

Figure 2.3.: A generic RAG architecture. The user queries, spanning different modalities, serve as input to both the retriever and the generator. The retriever extracts relevant information from data sources. The generator interacts with the retrieval results and ultimately produces outcomes of various modalities. [3]

to issues such as the generation of hallucinated content. RAG models address these challenges by incorporating a retrieval component that dynamically fetches relevant information from an external database during the generation process.

Core Components of RAG

- **Parametric Memory:**

  The parametric memory component in RAG is typically a pre-trained sequence-to-sequence (seq2seq) model. This model is responsible for the actual generation of text based on the input query and the retrieved passages. The seq2seq model utilizes the extensive knowledge encoded in its parameters to provide contextually appropriate responses [7].

- **Non-Parametric Memory:**

  The non-parametric memory consists of a dense vector index, often built from a large corpus such as Wikipedia. This index is accessed using a neural retriever, which identifies the most relevant passages for a given query. The retriever uses a pre-trained query encoder to transform the query into a dense vector and then retrieves the closest matching passages from the index using Maximum Inner Product Search (MIPS) [7].

- **Integration Mechanism:**

  RAG models combine the retrieved passages with the original query to form an augmented input for the seq2seq model. There are different approaches to integrating these passages. One method conditions the entire generated sequence on the same set of retrieved passages, while another allows different passages to be used for generating each token, enhancing the specificity and diversity of the generated content [7].

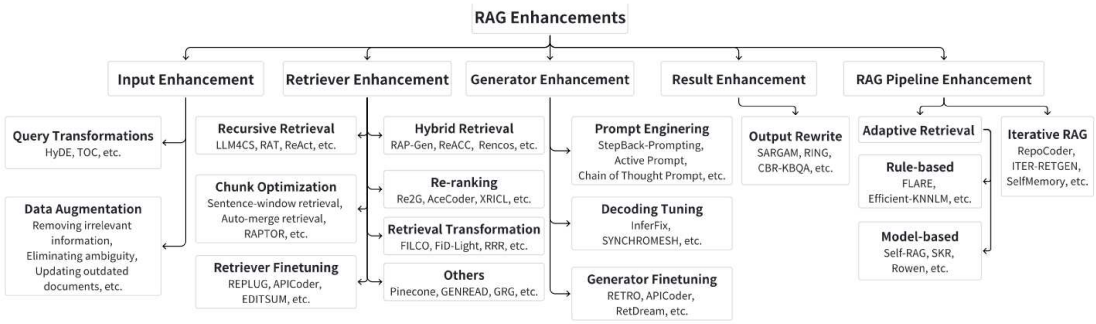

Figure 2.4.: Taxonomy of RAG Enhancements.[3]

Advantages of RAG

RAG models offer several advantages over traditional LLMs:

- **Dynamic Knowledge Access:** By retrieving real-time data from external sources, RAG models can generate more accurate and up-to-date responses.
- **Reduced Hallucinations:** The non-parametric memory provides a grounding for the generated content, reducing the risk of generating incorrect or hallucinated information [7].
- **Improved Specificity and Diversity:** The ability to condition on different passages for each token allows RAG models to produce more nuanced and contextually relevant responses [7].

Evaluation and Performance

RAG models have been shown to outperform traditional seq2seq models and other task-specific architectures in various knowledge-intensive NLP tasks, including open-domain question answering. They provide more specific, diverse, and factual language generation, setting new benchmarks in several areas [7].

Technological advancements in Retrieval-Augmented Generation (RAG) systems have significantly enhanced their performance and applicability across various domains. These advancements can be categorized into enhancements targeting input, retriever, generator, result, and the entire pipeline. Each category addresses specific aspects of the RAG process, contributing to the overall effectiveness and efficiency of the system.

Input Enhancements

Input enhancements primarily focus on improving the quality of the initial query fed into the retriever. Two notable methods in this category are query transformation and data augmentation. Query transformation involves modifying the input query to generate a pseudo document that contains richer relevant information, thereby improving the accuracy of the retrieval results. An example of this is the HyDE method, which generates a pseudo document using the original query [3] [9], [10]. Data augmentation techniques, on the other hand, involve updating or synthesizing new data to enhance the quality of

the retrieval process. Methods like Make-An-Audio use captioning and audio-text retrieval to generate captions for language-free audio, addressing data sparsity issues and improving the retrieval outcome [3].

Retriever Enhancements

Enhancements to the retriever are critical as they directly influence the quality of the content fed into the generator. Recursive retrieval and chunk optimization are two significant techniques in this area. Recursive retrieval involves performing multiple searches to gather richer and higher-quality content, as seen in the ReACT method, which uses Chain-of-Thought (CoT) to break queries down for recursive retrieval [3] [11]. Chunk optimization adjusts the size of the data chunks retrieved to improve the retrieval results, with methods like LlamaIndex employing a 'small to big' principle to fetch finer-grained content initially and then returning richer information [3].

Generator Enhancements

Enhancements to the generator are crucial for improving the quality of the final output. Prompt engineering and decoding tuning are two important methods in this category. Prompt engineering involves designing prompts that better utilize the retrieved data, with techniques like Chain of Thought (CoT) prompting and active prompt design significantly improving the output quality [3]. Decoding tuning adjusts the hyperparameters of the generator to balance diversity and quality, as demonstrated by InferFix, which balances the temperature in the decoder to achieve optimal results [3].

Result Enhancements

Result enhancements focus on refining the generated output to meet the specific needs of downstream tasks. Output rewrite is a key technique here, where the generated content is rewritten to align better with the task requirements. SARGAM, for example, refines outputs in code-related tasks by employing a specialized transformer that aligns generated content with real-world code context [3].

Pipeline Enhancements

Pipeline enhancements aim to optimize the entire RAG process from retrieval to generation. Adaptive retrieval and iterative RAG are prominent methods in this category. Adaptive retrieval dynamically decides whether retrieval is necessary based on the complexity of the query, using classifiers to guide this decision-making process. Self-RAG, for instance, uses a trained generator to determine the necessity of retrieval based on the input query [3]. Iterative RAG refines results through multiple cycles of retrieval and generation, allowing for progressively improved outputs. RepoCoder employs this approach to enhance code completion by refining queries with previously generated code [3][12].

2.3 Challenges with RAG

Despite the significant advancements in Retrieval-Augmented Generation (RAG) systems, there remain numerous challenges that limit their full potential. These challenges span various aspects of the technology, from operational efficiency to robustness and scalability. Addressing these issues is crucial for enhancing the performance and applicability of RAG systems in diverse domains. Moreover, identifying and exploring future directions for research and development will be key to overcoming these obstacles and driving further innovation in this field. This section discusses the primary challenges faced by RAG systems and outlines potential avenues for future advancements.

RAG vs. Long Context

One of the primary challenges for RAG is managing long contexts. With advancements in LLMs, handling extensive contexts exceeding 200,000 tokens has become feasible. However, this capability raises questions about the necessity of RAG when LLMs can incorporate entire documents directly. Providing LLMs with a large amount of context at once can significantly slow down inference speeds. Moreover, RAG's chunked retrieval and on-demand input improve operational efficiency and allow for observable retrieval and reasoning processes, making RAG indispensable for verifying generated answers and managing complex problems requiring substantial material to answer[13].

Robustness Against Noise and Contradictory Information

RAG systems often struggle with robustness when confronted with noise or contradictory information during retrieval. The quality of RAG output can deteriorate significantly due to the presence of such adversarial inputs. Studies have shown that including irrelevant documents can sometimes unexpectedly increase accuracy, highlighting the complexity of developing strategies that effectively integrate retrieval with generation models. Enhancing the robustness of RAG against such inputs is a key area of ongoing research[13].

Hybrid Approaches

Combining RAG with fine-tuning (FT) emerges as a leading strategy to optimize performance. The challenge lies in determining the optimal integration approach—whether sequential, alternating, or through end-to-end joint training. This hybrid method aims to harness both parameterized and non-parameterized advantages, yet balancing these aspects requires further exploration. The introduction of small language models (SLMs) with specific functionalities into the RAG system and fine-tuning them based on RAG results is a promising direction[13].

Scaling Laws of RAG

The scalability of RAG models remains an open question. While scaling laws are well-established for LLMs, their applicability to RAG is less certain. Initial studies suggest that smaller models might sometimes outperform larger ones, an intriguing possibility

that warrants further investigation. Understanding how to scale RAG models effectively is crucial for enhancing their capabilities and performance[13].

Production-Ready RAG

Practical deployment of RAG systems involves several engineering challenges. Enhancing retrieval efficiency, improving document recall in large knowledge bases, and ensuring data security are critical issues. Preventing inadvertent disclosure of document sources or metadata by LLMs is particularly challenging. Developing robust, efficient, and secure RAG systems that can be readily adopted in production environments is essential for the broader application of this technology[13].

Multi-modal RAG

Expanding RAG beyond text to incorporate multi-modal data, including images, audio, and video, presents additional challenges. Developing models that can effectively retrieve and generate across various data modalities is complex and requires sophisticated integration strategies. This expansion holds significant potential but also demands robust methodologies to manage the increased complexity and ensure consistent performance across different types of data[13].

2.4 Domain-Specific Applications of RAG Systems

Retrieval-Augmented Generation (RAG) systems represent a significant advancement in the field of artificial intelligence, combining the strengths of dynamic information retrieval with sophisticated generative models. This integration allows RAG systems to access and utilize current, domain-specific knowledge, enhancing their relevance and accuracy across various applications. The versatility of RAG makes it a powerful tool in diverse fields such as healthcare, finance, and education, where the need for precise, up-to-date information is paramount. By bridging the gap between static data and dynamic retrieval, RAG systems provide innovative solutions to complex challenges, making them indispensable in today's data-driven world.

2.4.1 Healthcare Applications

Retrieval-Augmented Generation (RAG) systems have shown significant promise in enhancing various healthcare applications by combining the strengths of retrieval-based and generation-based models. These systems leverage external knowledge bases to generate accurate, contextually relevant, and up-to-date responses, making them particularly useful in the healthcare domain.

Medical Report Generation

One of the primary healthcare applications of RAG is in the generation of medical reports. Traditional approaches to medical report generation often rely on predefined templates, which can limit the diversity and specificity of the generated reports. RAG

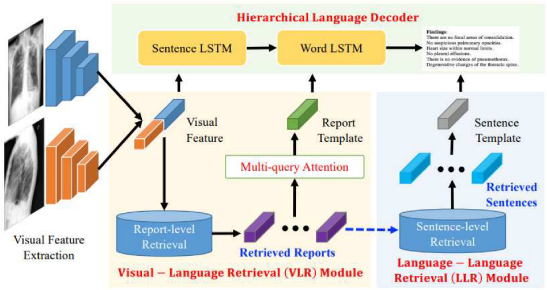

Figure 2.5.: Overview of the MedWriter[4]

systems, such as MedWriter 2.5, incorporate a hierarchical retrieval mechanism that automatically extracts both report and sentence-level templates from data. This ensures that the generated reports are not only clinically accurate but also tailored to the specific medical images being analyzed. MedWriter employs a Visual-Language Retrieval (VLR) module to retrieve the most relevant reports based on visual features and a Language-Language Retrieval (LLR) module to ensure logical coherence between sentences [4]. MedWriter's approach has been validated on large-scale datasets such as Open-I and MIMIC-CXR, demonstrating superior performance compared to state-of-the-art baselines. The system achieves better results in terms of CIDEr, ROUGE-L, and BLEU scores, indicating its ability to generate more accurate and natural descriptions for medical images [4].

Question Answering in Medical Contexts

Another critical application of RAG in healthcare is in the field of medical question answering. These systems can dynamically retrieve relevant information from extensive medical literature and databases to provide precise and contextually appropriate answers to medical queries. For instance, QA-RAG systems are designed to handle complex medical questions by integrating retrieval mechanisms that fetch relevant documents during the answering process. This approach ensures that the answers are not only based on pre-trained knowledge but also incorporate the latest research and clinical guidelines [14].

Optimizing Clinical Decision Support

RAG systems also play a crucial role in optimizing clinical decision support systems. By combining retrieval and generation capabilities, these systems can provide healthcare professionals with up-to-date and evidence-based recommendations. For example, the integration of RAG in hepatological clinical optimization involves using retrieval-augmented mechanisms to fetch relevant clinical guidelines and research articles, thereby supporting physicians in making informed decisions about patient care [15].

Enhancements in Radiology Report Generation

In radiology, the generation of detailed and accurate reports is critical for patient diagnosis and treatment planning. RAG systems like MedWriter improve the quality of radiology reports by incorporating multi-query attention mechanisms that fuse retrieved

information with image features. This results in reports that are not only accurate but also capture subtle and significant medical findings that may be missed by generation-only models [4].

2.4.2 Business Applications

Retrieval-Augmented Generation (RAG) systems have shown substantial potential in enhancing various business applications by integrating the capabilities of information retrieval with the generative power of large language models (LLMs). This combination allows businesses to access, process, and utilize vast amounts of data more effectively, leading to improved decision-making, customer service, and data management.

Financial Data Analysis

RAG systems are particularly effective in the financial sector, where they can be used to analyze large volumes of financial statements and reports. By dynamically retrieving relevant data and generating insightful summaries, these systems help financial analysts and decision-makers extract critical information efficiently. For instance, the use of RAG in financial statement analysis enables the system to parse through extensive financial documents, retrieve pertinent data points, and generate comprehensive reports that highlight key financial metrics and trends [16] .

Customer Service Enhancement

In the realm of customer service, RAG systems have revolutionized the way companies handle customer inquiries and support tickets. By integrating knowledge graphs with retrieval-augmented generation, RAG systems can provide accurate and contextually relevant responses to customer queries. This approach not only improves the quality of customer support but also significantly reduces the time required to resolve issues. The integration of RAG with knowledge graphs ensures that the system maintains the structure and relationships within customer service data, leading to more precise and useful answers [17].

For example, LinkedIn's deployment of RAG for customer service involved creating a knowledge graph from historical issue tickets and using it to enhance the retrieval and answering process. This method improved retrieval accuracy by 77.6% in Mean Reciprocal Rank (MRR) and boosted BLEU scores for answer quality by 0.32, demonstrating significant gains in both retrieval performance and response quality [17].

Table-to-Answer Systems

Another critical application of RAG systems in business is in generating answers from large, structured data sets like tables. The ERATTA (Extreme RAG for Table-To-Answers) system is a prime example, where multiple LLMs are used to authenticate user access, route queries, retrieve data, and generate natural language responses. This system is particularly beneficial for enterprise-level data products, enabling real-time responses to user queries about highly varying and large data tables. ERATTA has

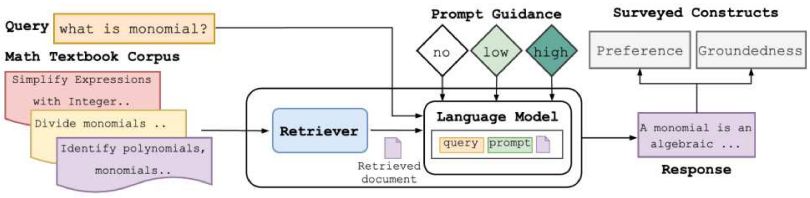

Figure 2.6.: Proposed RAG generates responses to math student queries [5]

shown over 90% confidence scores in various domains, including sustainability and financial health, by ensuring accurate data retrieval and minimizing hallucinations in the generated responses [18].

Enhancing Business Intelligence

RAG systems also enhance business intelligence by providing deeper insights through data retrieval and generation. By processing large datasets, these systems can generate detailed reports, predictive analyses, and prescriptive recommendations that are vital for strategic planning and operational efficiency. The ability to dynamically retrieve up-to-date information and generate contextually relevant insights helps businesses stay competitive and make informed decisions [18].

2.4.3 Educational Applications

Retrieval-Augmented Generation (RAG) systems have shown significant potential in transforming educational applications by enhancing the quality of automated responses and personalized learning experiences. By integrating retrieval mechanisms with generative models, RAG systems can provide more accurate and contextually relevant information, which is particularly beneficial in educational settings.

Improving Math Question-Answering

One of the key educational applications of RAG is in improving math question-answering systems. Traditional large language models (LLMs) can sometimes generate incorrect or contextually inappropriate responses to math questions. RAG systems address this issue by incorporating vetted external knowledge sources into the LLM prompts to increase the accuracy and relevance of the responses. For instance, a study implemented a RAG system using content from a high-quality open-source math textbook to generate responses to middle-school algebra and geometry questions. The results indicated that while humans preferred responses generated using RAG, the responses needed to strike a balance between being grounded in the textbook content and meeting student preferences [5].

Uncovering Knowledge Gaps

RAG systems can also be used to uncover knowledge gaps in educational content. By simulating user search behavior, RAG models identify and address gaps in information

retrieval systems. This approach is crucial for enhancing the efficacy of information retrieval in educational contexts, as it helps educators and content developers understand what information is missing or inadequately covered. For example, the AskPandi system uses RAG to mimic user behavior and generate relevant follow-up questions, thus identifying areas where educational content can be improved [19].

Enhancing Interactive Learning

Interactive learning platforms can significantly benefit from RAG systems by providing more relevant and accurate responses during student interactions. For example, in the context of a math chatbot used by middle-school students, RAG was employed to retrieve relevant content from textbooks and generate responses that are both accurate and pedagogically sound. This method ensures that the chatbot's answers are aligned with the educational curriculum and provide meaningful learning support to students [5].

2.5 Evaluation Metrics and Validation of RAG Systems

In the evaluation of Retrieval-Augmented Generation (RAG) systems, it is crucial to focus on several key metrics and validation methodologies to ensure the effectiveness and reliability of these systems. The following sections provide an overview of the most important evaluation metrics and validation techniques for RAG systems, drawing insights from several key papers in the field.

2.5.1 Evaluation Metrics

Retrieval Evaluation

Retrieval performance in RAG systems is primarily assessed using metrics that gauge the relevance and accuracy of the retrieved documents. Key metrics include:

- **Mean Average Precision at K (MAP@K):** This metric evaluates the average precision of the top-K retrieved documents. It is calculated as the mean of the precision scores at different recall levels for the top-K retrieved documents. Higher MAP@K values indicate better retrieval performance [20].
- **Mean Reciprocal Rank at K (MRR@K):** This metric measures the rank position of the first relevant document in the top-K retrieved documents. The reciprocal rank of the first relevant document is averaged over all queries. MRR@K is particularly useful for understanding how quickly a relevant document appears in the ranked list[20].
- **Hit Rate at K (Hit@K):** This metric indicates the proportion of queries for which at least one relevant document is found within the top-K retrieved documents. A higher hit rate suggests that the retrieval system is effective in fetching relevant documents early in the ranking[20].

Several recent studies provide valuable insights into the evaluation of RAG systems:

- **MultiHop-RAG:** This study highlights the challenges of answering multi-hop queries, which require reasoning over multiple pieces of evidence. The authors developed a novel dataset, MultiHop-RAG, to benchmark the retrieval and reasoning capabilities of various state-of-the-art language models. Their findings indicate that existing RAG methods struggle with multi-hop queries, underscoring the need for improved retrieval strategies[21].

- **RAGGED:** This paper focuses on the evaluation of RAG systems in noisy environments. It introduces the RAGGED dataset, which includes queries with varying levels of noise and complexity. The study emphasizes the importance of robust retrieval mechanisms to handle noisy inputs effectively [22].

Generation Evaluation Metrics

Generation quality in RAG systems is evaluated based on the accuracy and relevance of the generated responses. Important metrics include:

- **Accuracy:** This metric measures the proportion of correct answers generated by the RAG system compared to the ground truth answers. It is a direct indicator of the system's ability to produce correct and relevant responses[20].

- **BLEU Score:** This metric assesses the quality of the generated text by comparing it to reference texts. BLEU (Bilingual Evaluation Understudy) score considers the overlap of n-grams between the generated and reference texts, providing a quantitative measure of textual similarity[20].

- **ROUGE Score:** Similar to BLEU, ROUGE (Recall-Oriented Understudy for Gisting Evaluation) evaluates the overlap of n-grams, but it also considers recall and precision. ROUGE is particularly useful for summarization tasks where the completeness of the content is important[20].

Several recent studies provide valuable insights into the evaluation of generation quality in RAG systems:

2.5.2 Validation Techniques

To ensure the robustness and generalizability of RAG systems, various validation techniques are employed:

Cross-Validation: This technique involves partitioning the dataset into multiple subsets and training the model on each subset while validating it on the remaining data[20]. Cross-validation helps in assessing the model's performance across different data splits, ensuring that it is not overfitting to a particular subset.

Human Evaluation: Although automated metrics are useful, human evaluation remains the gold standard for assessing the quality of generated responses[20]. Human evaluators rate the responses based on relevance, coherence, and factual accuracy. This subjective assessment provides insights into the practical usability of the system.

Benchmark Datasets: Utilizing standardized benchmark datasets, such as MultiHop-RAG, is essential for consistent evaluation and comparison of RAG systems[20]. These

datasets provide a diverse set of queries and corresponding ground truth answers, enabling comprehensive performance evaluation.

2.6 Gaps in the Literature and Justification for the Lookinglass

Current challenges such as handling complex queries, minimizing factual inaccuracies, and ensuring data privacy highlight the necessity for innovative solutions. This study introduces Lookinglass, a robust RAG system designed to address these critical gaps and enhance the practical utility of language models in real-world scenarios.

2.6.1 Identified Gaps

- Factually Incorrect Responses: Traditional language models often generate responses that are factually incorrect or misleading. Existing research highlights the issue of "hallucinations" where models produce inaccurate outputs.
- Handling Complex Queries: There is a significant challenge in managing long-tail data and multi-hop queries that require reasoning over multiple pieces of supporting evidence. Most systems struggle with such complex queries.
- Privacy Concerns: Implementing robust mechanisms to prevent data leakage is crucial. The system must ensure that sensitive information is protected and only accessible to authorized users. This involves developing strong access controls, encryption, and other security measures.
- Practical Business Applications: While there is a growing interest in applying language models to business contexts, many existing systems are not versatile or robust enough to handle diverse business tasks effectively.

2.6.2 Justification for the Lookinglass

The gaps identified underscore the necessity for an advanced RAG system, like Lookinglass, with the following justifications:

- Reducing Hallucinations: By integrating sophisticated retrieval mechanisms, Lookinglass aims to minimize instances of hallucinations, thereby enhancing the accuracy and reliability of language model outputs. This improvement is critical for applications requiring high factual correctness.
- Complex Query Management: The ability to handle multi-hop queries and long-tail data is essential for addressing complex business inquiries. Lookinglass is designed to manage such complexity, providing accurate responses by reasoning over multiple pieces of evidence.
- Ensuring Privacy and Security: The system allows users to set customizable security levels, ensuring that only authenticated users with the appropriate level of security can access sensitive information. This feature is particularly useful for

organizations that need to maintain strict confidentiality while allowing controlled access to critical data.

- Versatile Business Applications: The system's design focuses on practical business applications, such as document-based question answering, information retrieval, and content generation. This versatility ensures that Lookinglass can be effectively utilized across various real-world business scenarios.

Overall, the development of Lookinglass addresses the critical gaps in current RAG systems by enhancing factual accuracy, managing complex queries, ensuring security, and supporting a wide range of business applications. This study is justified by the need to overcome the limitations of existing language models and provide a more reliable, secure, and versatile solution for business contexts.

2.7 Summary

This chapter provides a comprehensive analysis of the advancements and challenges in artificial intelligence (AI) and natural language processing (NLP), focusing on the evolution of Large Language Models (LLMs) like BERT and GPT, and the emerging paradigm of Retrieval-Augmented Generation (RAG). The chapter highlights the transformative impact of LLMs, enabled by innovations such as the Transformer model, BERT's bidirectional training, and GPT's generative capabilities. Despite these advancements, traditional LLMs face limitations in dynamic and contextually accurate information generation, addressed by RAG systems that integrate retrieval mechanisms to enhance response accuracy and relevance. Detailed comparisons between RAG and LLMs emphasize RAG's superior ability to handle complex, knowledge-intensive tasks by dynamically accessing external information. The review also explores the structural components, advantages, and technological enhancements of RAG, along with its domain-specific applications in healthcare, business, and education. Challenges such as handling long contexts, robustness against noise, and scalability are discussed, underscoring the need for ongoing research. The chapter concludes by identifying gaps in current systems, justifying the development of the Lookinglass system to address these issues and improve practical applications in real-world scenarios.

3

Methodology

The ultimate goals of the Retrieval-Augmented Generation (RAG) system, Lookinglass, are centered around enhancing the capabilities of language models by integrating information retrieval mechanisms to improve response accuracy and reliability. This document outlines the system requirements and specifications essential for developing an effective and robust RAG system in a business context.

3.1 Primary Goals and Objectives

The Lookinglass RAG system aims to overcome several challenges associated with traditional language models. The key goals include:

1. **Enhancing Language Model Capabilities:** By integrating retrieval mechanisms, the system aims to reduce instances of hallucinations where language models generate factually incorrect or misleading responses [23]. This enhancement is critical for ensuring that the outputs of the language model are both accurate and reliable.

2. **Improving Handling of Complex Queries:** The system is designed to better manage long-tail data and multi-hop queries, which require reasoning over multiple pieces of supporting evidence [21]. This capability is vital for addressing complex business inquiries that go beyond simple, single-hop responses.

3. **Addressing Privacy Concerns:** The Lookinglass includes mechanisms to mitigate the risk of data leakage, ensuring that the use of proprietary and private datasets does not inadvertently expose sensitive information. This is achieved through robust data handling and retrieval processes that prioritize confidentiality.

4. **Facilitating Practical Business Applications:** The system is intended to be versatile and applicable across various business tasks such as document-based question answering, information retrieval, and content generation, thereby enhancing the practical utility of language models in real-world business scenarios.

3.2 Requirments

The requirements for the Lookinglass RAG system encompass both functional and non-functional aspects. Functionally, it must accurately retrieve relevant documents, seam-

lessly integrate retrieved data into the generation process, produce coherent and factually accurate responses, and offer a user-friendly interface. Non-functionally, it must ensure high performance, reliability, scalability, and security, handling a high volume of queries efficiently while protecting sensitive data.

3.2.1 Functional Requirements

- **Accurate Information Retrieval:** The system must be capable of retrieving relevant documents and data efficiently. This involves sophisticated indexing techniques and the ability to search large datasets quickly.
- **Effective Augmentation of Retrieved Data:** Once data is retrieved, the system must integrate it seamlessly into the generation process. This includes preprocessing the retrieved information to ensure it is in a suitable format for the language model to use.
- **Robust Response Generation:** The language model component should be capable of generating coherent, contextually appropriate, and factually accurate responses based on the retrieved information. This requires advanced natural language processing techniques and fine-tuning of the language model.
- **User-Friendly Interface:** The system should provide an intuitive interface for users to interact with. This includes clear input mechanisms for queries and easy-to-understand outputs.

3.2.2 Non-Functional Requirements

- **Performance:** The system must be able to handle a high volume of queries without significant latency. This includes efficient data retrieval and response generation processes to ensure quick turnaround times.
- **Reliability:** The system should operate consistently under varying conditions. It should maintain a high level of accuracy and dependability in its outputs.
- **Scalability:** The architecture should support scaling to accommodate increasing data volumes and user demands. This involves designing the system to be extensible and capable of integrating additional resources as needed.
- **Security:** The system must prioritize the protection of sensitive data. This includes implementing strong access controls to prevent unauthorized access and data breaches.

3.3 System Architecture

The Lookinglass system architecture 3.1 integrates advanced language models with robust information retrieval mechanisms, utilizing MongoDB for extracted data storage

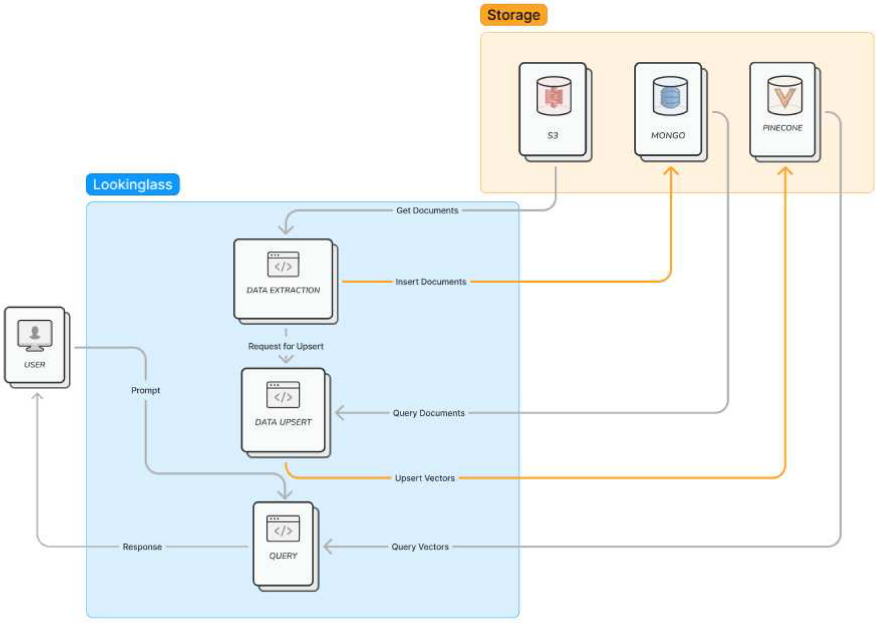

Figure 3.1.: Lookinglass Architecture

and Pinecone for vector storage. Unstructured and structured data undergo preprocessing and vectorization to enable efficient querying and accurate responses.

3.3.1 Data Extraction

The concept of data extraction within the Lookinglass system is integral to the effective functioning of applications that rely on structured and unstructured data. This discussion will focus on two core extraction methods used in the Lookinglass: contextual and tabular data extraction. Both methods are implemented as API endpoints designed to handle incoming data requests, process them accordingly, and insert them into the MongoDB. Below, we will delve into the mechanisms of detailed unstructured data extraction processes, including the handling of documents loaded from S3 and URLs, text splitting, and data aggregation.

Unstructured Data Extraction

Unstructured data extraction is designed to handle a broader range of data types.

- **Significance:** Contextual data extraction is essential for handling unstructured data, which is common in many real-world applications within the Lookinglass system. By providing a flexible mechanism for extracting and processing this data, the system can support a wide range of use cases, from document analysis to content aggregation.
- **Process:** The detailed textual data extraction process depicted in 3.2 involves mechanisms to load, split, and aggregate text data from documents. This is crucial for applications requiring granular control over text data, such as document summarization, text analysis, and content extraction within Lookinglass.

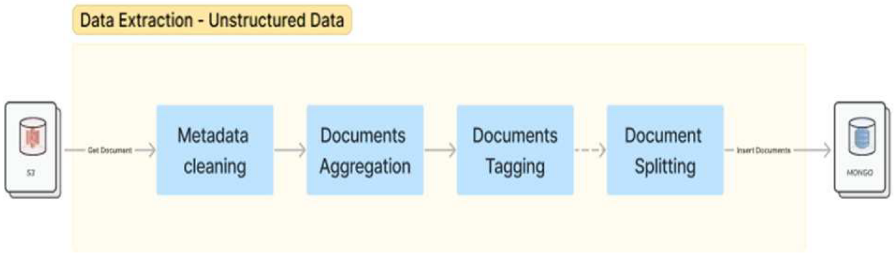

Figure 3.2.: Data Extraction for Unstructured data

1. **Loading Documents:** Documents are loaded from an Amazon S3 bucket, ensuring they are not empty before proceeding.
2. **Metadata cleaning:** The metadata of the documents is cleaned and updated, like setting and check if each document got the page number in its metadata or not.
3. **Aggregation:** Documents are aggregated based on the number of pages. This step combines multiple pages into a single document if necessary, ensuring that each chunk of text is appropriately sized for further processing.
4. **Document Tagging:** tagging the beginning and end of each page. In this module each document's content would be over-written using by following format:

   ```
   ***** Page \{page_number\} beginning *****
   \{page_content\}
   ***** Page \{page_number\} Ending *****
   ```

5. **Document Splitting:** Text is split into chunks based on specified parameters. This helps manage large text documents by breaking them into smaller, manageable pieces.

Structured Data Extraction

Tabular data extraction is critical when dealing with datasets that are organized into rows and columns, such as spreadsheets or database tables. This method involves fetching tabular data from cloud storage, and transforming it into a format that can be easily used by Lookinglass for further analysis.

- **Significance:** This method is vital for applications that need to process large amounts of tabular data efficiently. By extracting and transforming this data, the system can seamlessly integrate it into the broader data processing pipeline.
- **Process:** This process at 3.3 shows the following steps:
  1. **Loading Data:** The function utilizes an S3TableLoader to load the data from the specified location in a paginated manner, ensuring efficient handling of potentially large datasets.
  2. **Data Transformation:** Once loaded, the tabular data is processed into a list of document objects, each containing page content and metadata. This transformation is crucial for downstream processing and integration.

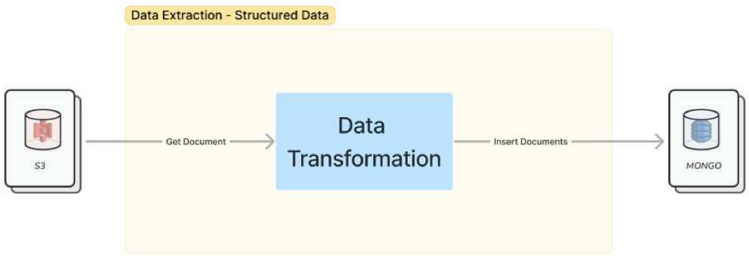

Figure 3.3.: Data Extraction for Structured data

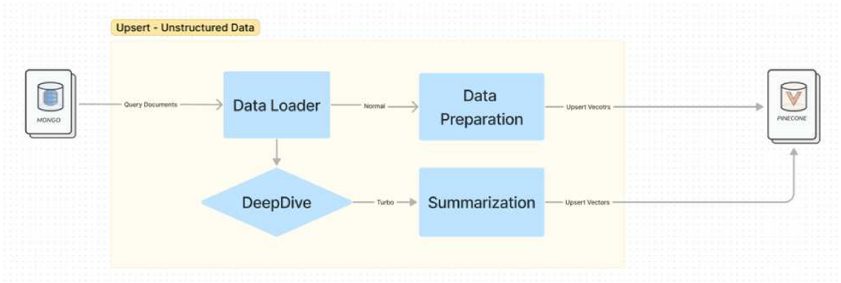

Figure 3.4.: Data Upsert for Unstructured data

The data extraction mechanisms within the Lookinglass system are foundational components that enable the efficient processing and utilization of both structured and unstructured data. The robust handling of structured and unstructured data ensures that the system can meet the demands of various applications.

3.3.2 Data Upsert

The data upsert process within the Lookinglass system is essential for integrating newly extracted data into the Pinecone vector database. This process involves retrieving data from MongoDB, vectorizing it, and then upserting it into Pinecone to ensure efficient retrieval and analysis.

Unstructured Data Upsert

The unstructured data upsert workflow 3.4 in the Lookinglass system involves several key steps to ensure the data is correctly processed and stored. The process can be divided into two main paths: normal upsert and turbo upsert, each serving different purposes and needs.

Process

- **Data Loader:** The process begins by querying MongoDB to retrieve the extracted data chunks stored during the data extraction phase. These chunks represent portions of text data that need to be further processed.
- **Data Preparation:**The retrieved data undergoes a preparation process to make

it suitable for embedding. This involves cleaning and formatting the data to ensure it is ready for word embedding using the OpenAI ADA model.

Embedding

Embedding is a crucial step in the data preparation process because it transforms textual data into numerical vectors. These vectors capture the semantic meaning of the text, allowing the Lookinglass system to perform efficient similarity searches and advanced data analysis. By converting text into a numerical format, embeddings enable the system to understand and process the underlying patterns and relationships within the data, which is essential for various applications such as recommendation systems, semantic search, and natural language processing tasks.

Why We Chose OpenAI ADA

We have chosen the OpenAI ADA model for embedding due to its high performance and versatility. ADA is known for its ability to produce high-quality embeddings that effectively capture the nuances and context of the text. It offers several advantages:

1. Accuracy: ADA provides highly accurate embeddings, which improve the overall performance of the Lookinglass system in terms of search relevance and data analysis [24].
2. Efficiency: The model is optimized for efficiency, allowing for fast processing of large volumes of data without compromising on quality [24].
3. Scalability: ADA can handle a wide range of text types and sizes, making it a scalable solution for diverse data sets and applications within Lookinglass [24].
4. Robustness: The embeddings generated by ADA are robust, meaning they maintain their quality and relevance even when applied to varied and complex text data. By leveraging the capabilities of the OpenAI ADA model, the Lookinglass system ensures that the data is accurately and efficiently embedded, enabling powerful and precise search and analysis functionalities [24].

DeepDive: Enhanced Upsert with Summarization

The DeepDive feature provides an enhanced upsert option that includes summarization. This feature is optional and can be enabled by the user for additional insights and detailed query capabilities The turbo upsert process involves generating summaries of the original documents using a GPT model. These summaries, along with the original data vectors, are then upserted into the Pinecone database.

Summarization

The summarization process in the Lookinglass system is a critical component of the DeepDive feature, which provides enhanced query capabilities by generating detailed summaries of documents. This process leverages a MapReduce strategy to efficiently

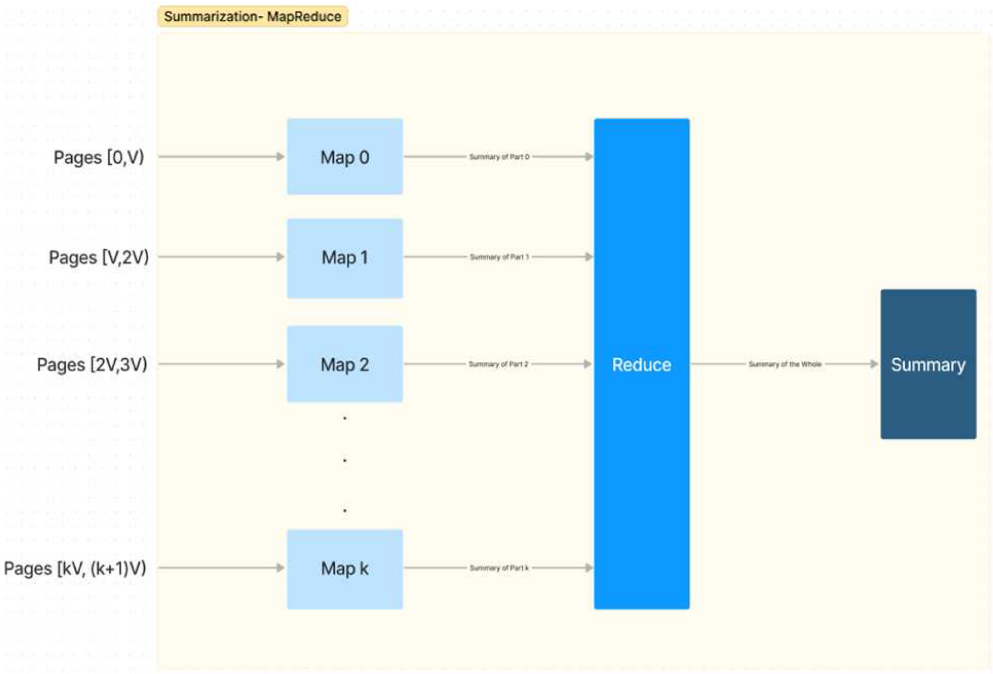

Figure 3.5.: MapReduce Technique for Summarization

handle and summarize large text data sets, utilizing the GPT-3.5 model for generating high-quality summaries.

The summarization process begins by querying the MongoDB to retrieve the extracted data splits. These splits are then divided into multiple parts and processed in parallel using a MapReduce strategy 3.5. This approach allows for efficient handling of large volumes of text data, ensuring that the summarization process is both scalable and effective.

- **Map Phase:**
  1. **Data Splitting:** The retrieved document splits from MongoDB are divided into several smaller parts. Each part is assigned to a separate map task for parallel processing.
  2. **Individual Summarization:** Each map task is responsible for summarizing its assigned part of the document using the ChatGPT-3.5 model. This involves condensing the content into a concise summary while retaining the essential information and context.
  3. **Table of Contents Generation:** In addition to summarizing the text, each map task extracts and generates a table of contents for its respective part. This helps in organizing the summarized content.
  4. **Output Generation:** The output of each map task is a summarized document that includes both the summary and the table of contents for that part. These outputs are then collected for further processing.
- **Reduce Phase:**
  1. **Aggregation of Summaries:** The outputs from all the map tasks are fed

into a single reduce task. This task is responsible for aggregating the individual summaries and tables of contents generated by the map tasks.

2. **Final Summarization:** The reduce task combines the individual summaries into a comprehensive summary that encapsulates the entire original document. It also consolidates the tables of contents into a cohesive structure that reflects the summarized content.
3. **Output of the Whole Summary:** The final output of the reduce task is a document that includes the complete summary of the original text along with a detailed table of contents. This document provides a comprehensive and organized overview of the original content, making it easier for Lookinglass to navigate.

The MapReduce-based summarization process in the Lookinglass system offers several advantages:

1. Scalability: By dividing the document into smaller parts and processing them in parallel, the system can efficiently handle large volumes of text data.
2. Efficiency: The parallel processing approach reduces the time required to generate summaries, making the system more responsive and capable of handling high loads.
3. Comprehensiveness: The generation of detailed summaries and tables of contents ensures that Lookinglasss have access to concise yet comprehensive overviews of the original documents.
4. Enhanced Query Capabilities: The detailed summaries generated by the DeepDive feature enable the system to handle complex and specific queries, providing users with more precise and relevant information.

Use Cases of DeepDive

The DeepDive summarization feature provides additional information that enhances the system's ability to respond to detailed and specific queries. Some example use cases include:

1. **Character Development Analysis:** For example, analyzing the development of a specific character in a novel over multiple chapters.
2. **Comparative Analysis:** Comparing themes, topics, or figures across different documents or sections within a document.

Costs of DeepDive Upsert

While the DeepDive provides significant benefits, it also comes with certain costs:

1. **Time:** The process takes more time compared to a normal upsert due to the additional steps involved in summarization using the GPT-3.5 model.
2. **Cost:** The summarization process incurs costs associated with the use of the GPT-3.5 model provided by OpenAI.

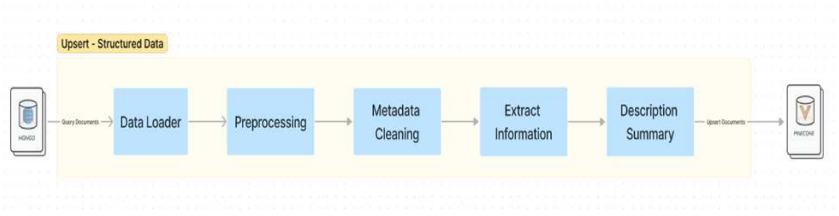

Figure 3.6.: Data Upsert for Structured Data

Structured Data Upsert

The upsert process for structured data in the Lookinglass system involves several key stages to ensure that tabular data is correctly processed and integrated into the Pinecone vector database. This comprehensive workflow includes data retrieval, preprocessing, metadata cleaning, information extraction, and summary generation. The diagram 3.6 illustrates this detailed process.

1. **Data Loader:** The Data Loader is responsible for fetching tabular data from MongoDB, similar to the unstructured data upsert process.

2. **Preprocessing:** Preprocessing prepares the data to ensure it is clean and suitable for analysis and querying. This stage involves several key actions to clean and normalize the data.

   - *Clean Column Headers:* Remove SQL and MongoDB-related keywords from column headers to prevent errors during query generation.
   - *Ensure Unique Column Names:* Check and enforce the uniqueness of column names to avoid conflicts.
   - *Replace NaN Values:* Substitute NaN (Not a Number) values with None to standardize the data.
   - *Drop Empty Columns:* Identify and remove columns that do not contain any data to streamline the dataset.

   Preprocessing ensures that the data is clean, consistent, and ready for subsequent stages, reducing the risk of errors during querying and analysis.

3. **Metadata Cleaning:** Metadata cleaning refines the metadata associated with each table. This step ensures that only relevant and useful metadata is retained, while unnecessary information is discarded.

   - *Remove Unnecessary Metadata:* Strip out irrelevant metadata such as coordinates and file directory information.
   - *Add Relevant Metadata:* Introduce essential metadata such as table names and other descriptive attributes.

   This step is essential for maintaining a clean and useful metadata structure, which aids in better understanding and utilization of the data.

4. **Extract Information** Extracting information from the DataFrame provides a detailed overview of the tabular data, making it easier to understand and analyze.

- *DataFrame Overview:* Extract comprehensive information such as: RangeIndex, Number of columns, Column names, Number of non-null values, Data type of each column
- *First and Last Rows :* Extract the first and last rows of the DataFrame

This detailed overview is crucial for data analysis and validation, providing query module with a clear understanding of the structure and content of the data.

5. **Description Summary** The description summary generation process in the Lookinglass system utilizes the capabilities of the GPT-4 model to produce detailed and informative summaries of tables. This is achieved by providing the model with a metadata of the table, which it then uses to create a natural language description. This description includes key components, columns, statistics, and other notable characteristics of the table.

   The GPT-4 model reviews the provided table metadata to understand the key elements such as the number of rows and columns, data types, and any notable statistics or characteristics. Based on this review, the model generates a coherent and informative description. For example, the description may highlight the number of rows and columns, the types of data present in each column, key statistics such as mean values and standard deviations, and any significant data points or observations.

   - *Infer Summary:* Use the metadata to create a descriptive summary of the table, highlighting key aspects and characteristics.
   - *Generate Summary with GPT-4:* Employ the advanced capabilities of the GPT-4 model to produce a high-quality summary that captures the essence of the table. The description summary provides Lookinglass with a quick and insightful overview of the table, enhancing its ability to effectively utilize the data in the table.

6. **Vectorization and Upsert**: After the summarization process, the structured data, is vectorized using the Ada embedding model. The vectorized data is then upserted into the Pinecone vector database, ensuring it is indexed and searchable for efficient retrieval and analysis.

The upsert process for both structured and unstructured data within the Lookinglass system ensures that data is accurately and efficiently integrated into the Pinecone vector database. This process involves meticulous steps including data retrieval, preprocessing, metadata cleaning, information extraction, summarization, and vectorization using the Ada embedding model. By following these comprehensive workflows, the Lookinglass system guarantees that data is clean, well-organized, and readily available for advanced analysis and application. This robust integration enhances the system's capability to provide precise and relevant information to its users.

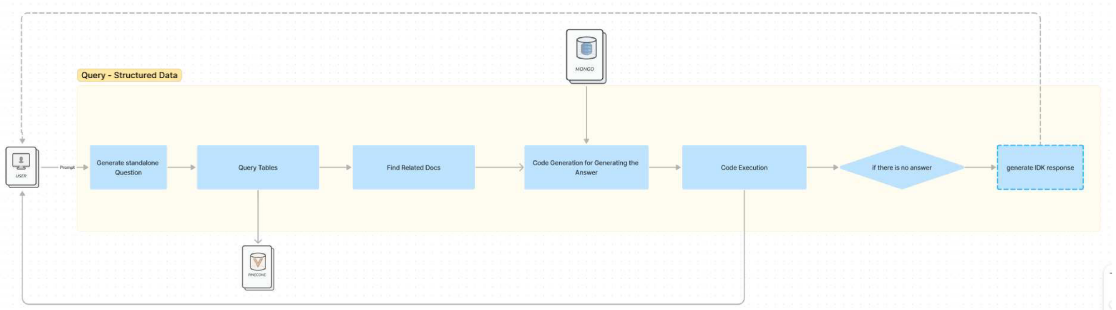

Figure 3.7.: Query Module for Unstructured Data

3.3.3 Query

The query process in the Lookinglass system is designed to leverage the structured and unstructured data upserted into the Pinecone vector database. This section will delve into the mechanisms by which the system retrieves and processes data to answer complex queries efficiently and accurately. By utilizing advanced vector search capabilities and metadata filtering, the Lookinglass system can provide detailed and relevant responses to a wide range of user queries.

Unstructured Data

1. **Generate standalone question:** The initial phase of the query process involves converting the user's prompt into a standalone question. This step ensures that the question is clear, specific, and unambiguous, which is crucial for retrieving accurate and relevant information. The user's prompt, along with the chat history, is sent to this module. Using GPT model, the system rephrases the follow-up question into a standalone question. The process includes:
   - *Identifying the language of the follow-up question.*
   - *Rephrasing the question to be standalone while maintaining the same language.*
   - *Adjusting any relative dates mentioned to exact dates based on the current date.*
   - *Ensuring that ambiguous pronouns are replaced with specific names or entities mentioned in the conversation history.*

   This rephrasing step guarantees that the question is self-contained and fully comprehensible, enabling more precise and effective querying of the database.

2. **Querying Data from Pinecone:** Once the standalone question is generated, the system queries both the original documents and their summaries stored in the Pinecone vector database. This dual-query approach allows the system to leverage the detailed information in the original documents and thorough insights of text using the summaries.

   Semantic Search: The query process uses a similarity-based method for semantic search within the Pinecone vector database. This method is particularly

effective for understanding the context and meaning behind the question, rather than relying solely on keyword matching.

- *Query Summaries:* For summaries, the system retrieves the top 5 documents that are most semantically similar to the standalone question. This ensures that the concise, high-level insights provided by the summaries are relevant to the user's query.
- *Query Original Documents:* For original documents, the system retrieves the top 10 documents that are most semantically similar to the standalone question. This provides a deeper and more comprehensive understanding of the topic in question.

3. **Removing Similar Documents:** After the initial retrieval of documents, both summaries and original documents undergo a process to remove similar documents. This step is essential for ensuring that the final set of results is diverse and not cluttered with redundant information. Removing similar documents helps in maintaining the uniqueness of the responses. It ensures that the information provided to the user is varied and comprehensive, rather than repetitive.

4. **Pruning Original Documents:** Pruning Original Documents is a critical phase in the query process that ensures the most relevant documents are retained while less pertinent ones are discarded. This step is executed using GPT model to assess the relevance of each document to the user's question.

   Process

   a) *Context and Question Review:* The system first reviews the content of each document (referred to as the context) alongside the user's standalone question. This involves a thorough examination of both the document content and the question to understand their respective details.

   b) *Relevance Assessment:* Using GPT model, the Lookinglass determines if the document contains information that answers the user's question. This assessment is not limited to exact matches but also considers partial relevance. If any part of the document addresses the question, it is deemed relevant.

   c) *Filtering Based on Relevance:*

      - True Relevance: If the document is found to be relevant (even partially), it is marked as relevant.
      - False Relevance: If the document does not contain any pertinent information, it is marked as irrelevant and excluded from further processing.

   d) *JSON Output:* The relevance determination is output as a JSON object with a boolean result. This object indicates whether each document should be retained ('true') or discarded ('false') based on its relevance.

   By filtering out irrelevant documents, this pruning phase ensures that the subsequent steps in the query process focus only on the most pertinent and useful information. This enhances the efficiency and accuracy of the system's responses, providing users with highly relevant and precise answers to their queries.

5. **Pruning Summaries:** In this phase, OpenAI model is used to assess the relevance of each structured summary of the documents to the user's question. The process involves a detailed review of both the document summary and the user's query to ensure that only pertinent documents are retained.

   Process

   a) *Review Structured Summary and Table of Contents:* The content of the structured summary and the table of contents are carefully examined to understand the key topics and sections covered by the document.

   b) *Analyze User's Question:* The user's question is thoroughly reviewed to identify the specific information or topics being queried.

   c) *Compare Topics and Sections:* The pages and topics covered in the original document are identified and compared with those mentioned in the structured summary. This comparison helps determine if the relevant information can be found within the pages covered by the summary.

   d) *Determine Relevance:* If the system finds that the answer to the question can be found within the range of pages that the summary covers, it marks the document as relevant. This relevance is determined based on several criteria:

      - Specific topics, page numbers, or sections mentioned in the query that match the structured summary.
      - The query references the name of the source document and matches the source of the structured summary.
      - General themes or topics in the query that are broadly covered in the structured summary.

      If there is no connection between the query and the summary, the document is marked as irrelevant.

   e) *Generate Result:* The system responds with a JSON object indicating whether the document is relevant (true) or not (false), ensuring that only the most pertinent documents are kept for further processing.

   By employing this detailed pruning process, the Lookinglass system effectively filters out irrelevant documents, focusing on those that are most likely to contain the answers to the user's questions. This enhances the accuracy and relevance of the final results provided to the user.

6. **Filter Chain:** The filter chain phase is a crucial step in the query process for unstructured data within the Lookinglass system. This phase ensures that the most relevant portions of the documents are identified and utilized to answer the user's question accurately. Using GPT model, the system reviews the summaries and generates specific filters for pinpointing the relevant sections within the documents.

   Process

   During the filter chain phase, the retrieved unique pruned summary documents are analyzed to determine which sections of the original documents are most pertinent to the user's query. This is achieved by generating MongoDB filters that target

specific pages or sections within the documents based on their relevance to the query.

a) *Reviewing Content:* The system carefully examines the table of contents and the structured summary of each document. This helps in understanding the scope and details covered in different sections of the document.

b) *Analyzing the Question:* The user's question is reviewed to identify the specific information or topics being queried. If the language of the question and the summary differ, the question is translated to match the language of the summary.

c) *Answer Generation:* Based on the summary, the system attempts to answer the question. If it is possible to answer the question directly from the summary, the system generates a concise response. If the summary does not contain sufficient information to answer the question, the system states "NO_ANSWER" to indicate this.

d) *Page Identification:* The system identifies the pages within the original document that are most likely to contain the relevant information needed to answer the question. This is done by correlating the topics and sections mentioned in the table of contents and the structured summary with the query.

e) *Filter Construction:* A MongoDB filter is constructed to target the fields "source" and "pages". The filter specifies the document source and includes a list of page numbers that are relevant to the query. This filter helps in retrieving only the necessary sections of the document for further processing.

f) *Output:* The system responds with a JSON object that includes the answer to the query (if available) and the constructed filter. If the document is not relevant to the question, the filter value is set to "NO_FILTER".

Example

To illustrate how the filter chain phase works, consider the following example:


```
{
  "context": "<<<source: The Great Gatsby (text).pdf\n>>>\n########
  Document's Metadata:\n- Title: The Great Gatsby\n- Author(s):
  F. Scott Fitzgerald\n- Date of Publication: 1925\n- Upload and/or
  Last Modification: Uploaded on 2024-05-14\n- Source/URL: Not
  provided\n- Total Pages (Document): 193\n- Uploader: User with
  email ata@impssbl.com\n\n### Table of Contents (ToC):\n-
  Chapter 1: \"In my younger and more vulnerable years\" -
  Page 3-25\n- Chapter 2: \"About half way between West Egg and
  New York\" - Page 26-42\n- Chapter 3: \"Party at Gatsby's\" -
  Page 43-65\n- Chapter 4: - Page 66-86\n- Chapter 5: - Page
  87-103\n- Chapter 6: - Page 104-119\n- Chapter 7: - Page 120-156
  \n- Chapter 8: - Page 157-170\n- Chapter 9: - Page 174-180\n-
  Post-Gatsby's Death Events: - Page 181-185\n- Final Reflections
```

```
  and Conclusion: - Page 186-193\n\n### Summaries and Key Points b
  Section:\n#### Chapter 6: \"In my younger and more vulnerable years
  \"\n-87-103\n- Summary Points:**\n  - Narrator returns home to find
  Gatsby's house brightly lit, initially mistaking it for a fire or a
  party.
  "question": "How did Daisy Buchanan see Jay Gatsby?"
}
```


Filter chain output:


```
{
    "answer": "Daisy Buchanan's view of Jay Gatsby is complex and
    evolves over the course of the novel. Initially, she is charmed and
    intrigued by Gatsby's  wealth and the aura of mystery surrounding him.
    Their reunion is marked by emotional tension and awkwardness, but they
    eventually reconcile, and Gatsby's demeanor changes to one of well-
    being. However, as the story progresses, Daisy's feelings become
    conflicted, especially during the confrontation with Tom Buchanan.
    She admits to having loved both Tom and Gatsby, but ultimately,
    she withdraws from Gatsby, overwhelmed by the situation and the
    revelations about Gatsby's past.",
    "filter": "{'source': 'The Great Gatsby (text).pdf',
    'pages': { '$in': ['87', '88', '90', '91', '92', '93', '94', '95',
    '127', '143', '144', '146', '148'] }}"
}
```


This process ensures that the query system efficiently identifies and retrieves the most relevant sections of documents, providing accurate and contextually appropriate responses to user queries.

7. **Query Docs and Generate Initial Responses:** In these phases, the system takes the answers generated using the filter chain and combines them with other potential sources and page numbers identified during the filter chain process to create initial responses. The key steps involved are:

   a) *Query Documents:* The system queries the necessary documents from the Pinecone vector database using the filters generated in the previous steps. This ensures that only the most relevant documents are retrieved for answering the query.

   b) *Combine Generated Answers:* The retrieved documents are combined with the answers generated during the filter chain phase. This combination includes mixing the content from the filter chain's generated answers with the new information obtained from the vector database queries.

   c) *Remove Similar Documents:* After gathering all relevant documents and generated answers, the system performs a deduplication process to remove any similar documents. This step ensures that the initial responses are unique

and free from repetition.

d) *Integrate Data from Summaries and Originals:* Once the process is completed for all documents derived from the summaries, the system integrates these with the data retrieved from the original documents. This integration creates a comprehensive set of initial responses that include insights from both summaries and original texts.

e) *Final Deduplication:* After combining the integrated data, a final deduplication process is performed to remove any remaining similar documents. This final step ensures that the responses are diverse and cover a wide range of information without redundancy.

8. **QA Chain:** The QA chain is the final phase in generating an answer for the user's query. This phase ensures that if there is an answer, it is accurately provided to the user. If no relevant answer can be found, the system returns an "IDK" (I don't know) response. During this phase, all the potential answers gathered up to this point are consolidated and sent to the QA module, which utilizes the OpenAI GPT model to produce a coherent and accurate response.

   Process

   a) *Gathering Potential Answers:* All the answers identified during the previous phases are collected. These answers come from both the initial responses generated from the filter chain and the documents queried from the Pinecone vector database.

   b) *Utilizing Context:* The gathered answers and the context in which they were found are sent to the QA module. This module uses the OpenAI GPT model to review the context and formulate a comprehensive answer.

   c) *Answer Generation:* The GPT model processes the provided context to generate a helpful and relevant answer to the user's question. It strictly uses the information available in the context and avoids fabricating any details.

   d) *Listing Sources:* The model also lists the source files utilized to craft the answer. This ensures transparency and traceability of the information provided.

   e) *IDK Response:* If the model cannot generate an answer from the provided context, it returns an empty array, indicating that no relevant information was found. In such cases, the system responds with "IDK" (I don't know) to the user.

   f) JSON Output: The final response is formatted as a JSON object containing the answer and the list of source files used. This structured format ensures clarity and consistency in the responses provided to the user.

   Example:

   - **Question:** What mitigation strategies for climate change are discussed in the document?
   - **Context:** The context provided includes summaries and sections from various documents that may contain relevant information about climate change mitigation strategies

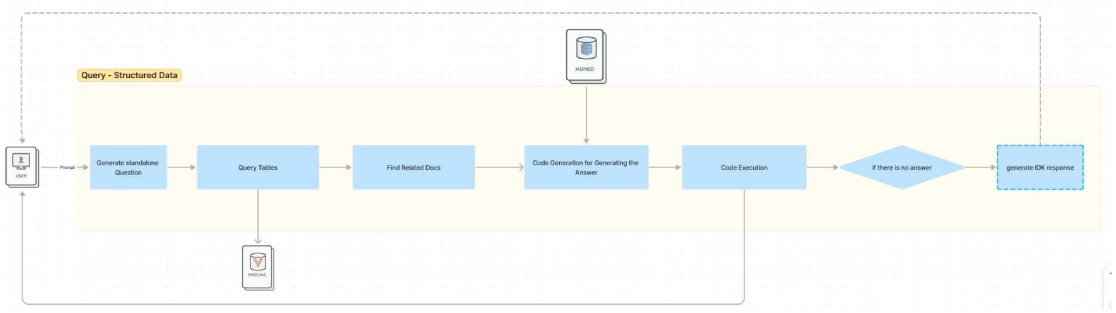

Figure 3.8.: Query Module for Structured Data

QA output:(if there is any answer)

```
{ "answer": "The document discusses various mitigation strategies for
climate change, specifically focusing on renewable energy initiatives,
carbon taxing and trading, and the implementation of global agreements
and policies. These strategies are detailed in the section titled
'Mitigation Strategies' and include approaches such as promoting
renewable energy sources, establishing carbon pricing mechanisms, and
strengthening international cooperation to reduce emissions.",
"source": ["cc_2024_env_report.pdf"] }
```

QA output:(No answer is available)

```
{ "answer": "IDK", "source": [] }
```

By following this thorough process, the QA chain ensures that the user receives the most accurate and relevant information available or a clear indication when no answer can be provided.

Structured Data

The query process for structured data in the Lookinglass system is designed to handle tabular data efficiently 3.8, enabling users to extract meaningful insights from large datasets stored in the Pinecone vector database. This comprehensive process involves several stages: generating a standalone question, querying tables, finding related documents, generating code to formulate the answer, executing the code, and, if necessary, generating an "IDK" (I don't know) response when an answer cannot be found. This structured approach ensures that the system can effectively respond to complex queries by leveraging advanced data processing and natural language generation techniques.

Process

1. **Generation of Standalone Question:** The initial step in the query process for structured data involves generating a standalone question from the user's prompt. This module utilizes the capabilities of GPT model to rephrase the user's follow-up question into a clear, unambiguous standalone question. To achieve this, the system:
   a) Rephrases the follow-up question to ensure it stands alone, maintaining the original language.

b) Adjusts any relative dates mentioned in the conversation to their exact dates based on the current date.

c) Ensures the rephrased question eliminates ambiguity by replacing pronouns with specific entities mentioned in the conversation history. The result is a concise and contextually clear standalone question that accurately reflects the user's intent, setting the stage for an effective query of the structured data.

2. **Query Tables:** In the Query Tables phase, the system utilizes the standalone question generated in the previous step to search for relevant tabular data within the Pinecone vector database. This is done using an similarity search method that efficiently retrieves the most relevant documents. The process involves:

   a) *Executing a Similarity Search:* The system performs a similarity search within the vector database using the standalone question as the query. This search method leverages the semantic understanding of the query to find documents that are contextually similar.

   b) *Applying Filters:* To ensure the relevance of the results, the search is filtered to target only tabular data. Additional access filters are applied as specified by the data access parameters.

   c) *Retrieving Results:* The system retrieves the top results from the search, typically the four most relevant documents. These documents are then used in the subsequent steps of the query process to generate accurate and comprehensive answers to the user's question.

   By utilizing this similarity search approach, the Lookinglass system ensures that the most pertinent tabular data is identified and used to provide precise responses to complex queries.

3. **Find Related Documents:** The Find Related Documents phase involves two key steps to ensure that only the most relevant documents are considered for generating the final response.

   a) *Removing Similar Documents:* Similar to the process used for unstructured data, this step involves removing documents that are highly similar to each other. This deduplication ensures that the final set of documents is diverse and does not contain redundant information. The system analyzes the semantic content of the retrieved documents and filters out those that are too similar, retaining only unique and relevant entries

   b) *Relevance Check:* After removing similar documents, the system conducts a relevance check to determine which documents are most pertinent to the user's query. This step involves evaluating each document summary to identify the one that best matches the query's requirements.

      The process includes:

      - Reviewing Summaries: Each DataFrame summary is reviewed to understand its content and context, which includes the table's source and name.

- Assessing Relevance: The summaries are assessed for relevance by comparing their content with the user's query. The system looks for mentions of specific tables, file names, or other relevant details that are directly addressed in the query.
- Identifying the Most Relevant Summary: The system identifies the index of the most relevant DataFrame summary, along with its source and table name, based on how well it matches the query.

For instance, if a user's query asks about specific details within a table, the system will evaluate the query against the summaries of available DataFrames. It then selects the summary that most accurately aligns with the query, ensuring that the selected document is highly relevant.

By removing duplicates and verifying the relevance of each document, the system ensures that the final set of documents used to generate responses is both unique and directly related to the user's question, thereby enhancing the accuracy and effectiveness of the query results.

4. **Code Generation for Generating the Answer:** In this phase, the system processes the relevant tables and documents to generate the necessary code that will answer the user's query. This process involves several detailed steps to ensure that the generated code is accurate, efficient, and capable of handling the data appropriately.

   a) *Extracting Information from Tables:* The system starts by gathering general information about the DataFrame. This includes summarizing the range index, listing the columns along with their data types and non-null counts, and providing a markdown representation of the DataFrame. If the DataFrame is small, the entire table is included; for larger DataFrames, only the head (first few rows) and tail (last few rows) are used.

   b) *Preparing for Code Generation:* Using the extracted information, the system prepares the DataFrame summary and the user's question for further processing. This setup ensures that the context needed for code generation is well-defined and comprehensive.

   c) *Generating Python Code:* The system then formulates Python code to perform the necessary operations on the DataFrame. this step is done using OpenAI model.

   d) *Handling Data and Outputs:* The generated code includes steps to how to store the final answer. For questions involving visualizations, the code uses Plotly to create aesthetically appealing and professional graphs.

   e) *Ensuring Code Quality:* To maintain code quality, the system ensures that special characters, new lines, and quotes are properly handled and escaped.

   This meticulous approach to code generation ensures that the system can effectively process the DataFrame and provide accurate and relevant answers to the user's questions. The generated code is designed to be robust, readable, and capable of producing professional-quality outputs, whether they are textual answers, or visualizations.

5. **Code Execution and Final Response:** Once the code is generated, it is executed along with the fetched table to derive the final answer. The execution process involves running the Python code within the specified environment, utilizing the data from the relevant DataFrame. This step ensures that all calculations, data manipulations, and visualizations are performed accurately based on the user's query. The system captures the output, whether it is a text-based answer, a table, or a graphical representation, and prepares it for delivery to the user.

   If the execution of the code yields a valid answer, it is formatted appropriately and returned to the user. This ensures that the response is both accurate and relevant to the query. However, if the code execution does not produce a meaningful answer, the system defaults to returning an "IDK" (I don't know) response. This mechanism ensures that the user is always provided with a clear and honest answer, maintaining the integrity and reliability of the Lookinglass system. By handling both successful and unsuccessful queries in this manner, the system upholds its commitment to providing precise and dependable information.

the query module of the Lookinglass system is a sophisticated and robust framework designed to handle both structured and unstructured data queries with precision and efficiency. Through a series of meticulously designed phases, including generating standalone questions, querying databases, removing duplicate documents, checking relevance, and generating executable code, the system ensures that users receive accurate and relevant answers to their queries. The integration of advanced natural language processing and data manipulation techniques enables Lookinglass to provide high-quality responses, whether they are textual answers, detailed tables, or professional-grade visualizations. This comprehensive approach not only enhances the user experience but also ensures that the information provided is reliable and actionable, solidifying Lookinglass as a powerful tool for data-driven decision-making.

3.4 Visual Overview of the Lookinglass

This section presents images of the Lookinglass environment, showcasing its user interface and key features. These visuals illustrate how users interact with the system and navigate its functionalities, providing a clear understanding of the platform's design and capabilities.

3.4.1 Upload Documents

The image 3.9 illustrates how users can upload documents into the Lookinglass with special configuration settings. The interface allows users to customize the upload process to meet specific needs, ensuring that documents are properly secured, categorized, and prepared for advanced processing.

- **Security:** Users can define who has access to the uploaded documents. This setting ensures that sensitive information is protected and only accessible to authorized personnel. By configuring the security options, users can control permissions and maintain data confidentiality.

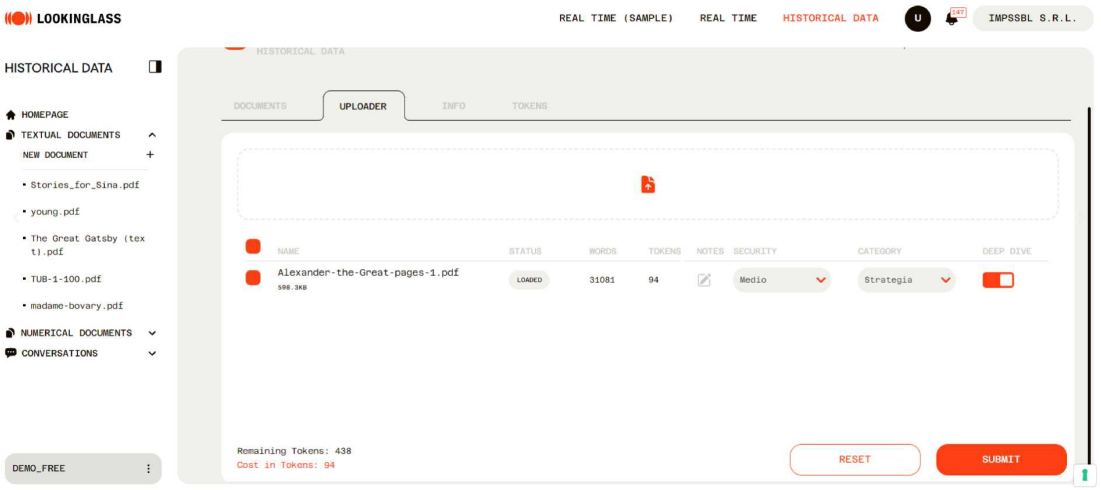

Figure 3.9.: Upload Page on Lookinglass System

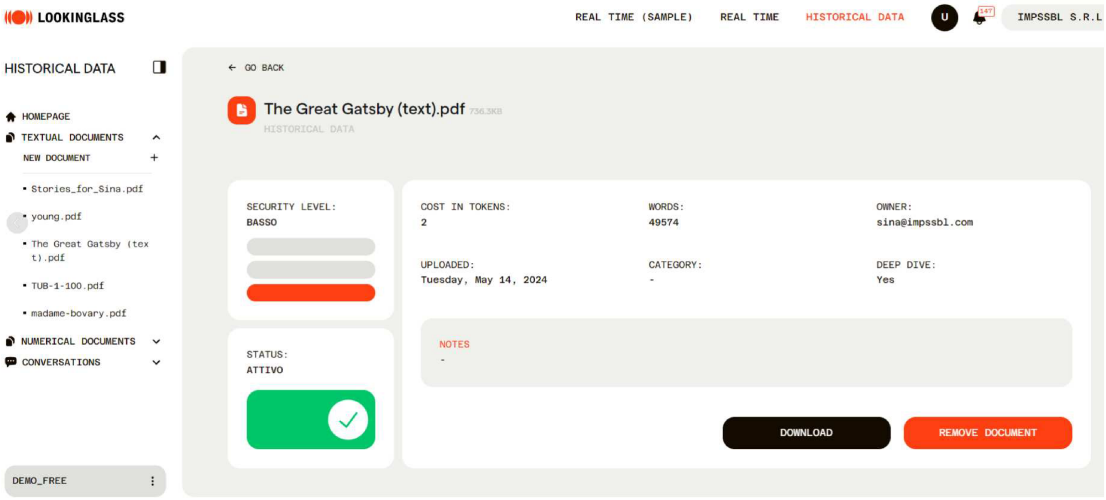

Figure 3.10.: Status of a document after being uploaded

- **Category:** During the upload process, users can categorize their documents. This feature allows for efficient organization and retrieval of documents based on predefined categories. Categorizing documents helps in maintaining a structured repository, making it easier to manage and locate files as needed.

- **Deep Dive:** The Deep Dive option, which is described in detail on page 44, can be enabled during the upload process. This setting activates the summarization of the file along with the original document and upserts both into the Pinecone vector database. Enabling Deep Dive ensures that the document is thoroughly analyzed and that detailed summaries are available for advanced querying and analysis.

At the end of this process, the original document is stored in Amazon S3, and then the data is extracted and inserted into MongoDB. Subsequently, it is upserted to the Pinecone database as well. This comprehensive approach ensures that the uploaded documents are not only securely stored and well-organized but also readily available for sophisticated data querying.

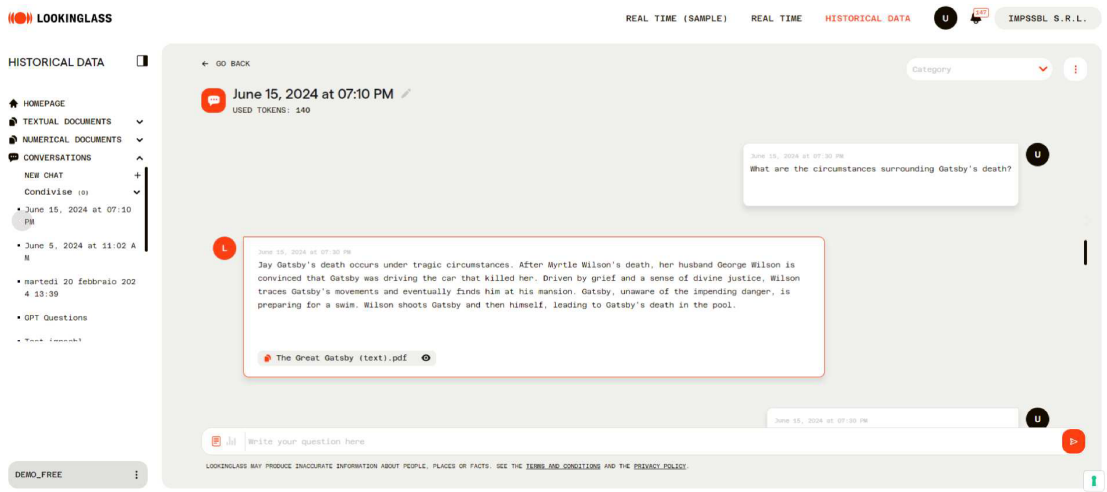

Figure 3.11.: Query page on Lookinglass System

3.4.2 Query Documents

The query environment 3.11 is designed to be user-friendly and efficient, providing a seamless experience for users to input their questions and receive answers based on the uploaded documents. Users can leverage the advanced search capabilities of Lookinglass to obtain precise and relevant responses to their queries.

Key features of the query environment include:

- *Question Input:* Users can enter their questions and interact with the system.
- *Answer Display:* The answers generated from the uploaded documents are displayed clearly. Users can see the answers if they exist, along with the sources from which the information was derived.
- *Source Selection:* Before starting a query, users have the option to select specific sources. This allows them to limit the scope of the search to particular documents within Lookinglass, ensuring that the results are highly relevant to their needs.

3.5 Summary

In this chapter, we have outlined the methodology behind the development and functioning of the Lookinglass Retrieval-Augmented Generation (RAG) system. Our primary goals include enhancing language model capabilities, improving the handling of complex queries, addressing privacy concerns, and facilitating practical business applications.

The system's requirements were divided into functional and non-functional categories, focusing on accurate information retrieval, effective data augmentation, robust response generation, and ensuring high performance, reliability, scalability, and security.

The system architecture details the processes for data extraction, upsertion, and query handling, encompassing both structured and unstructured data. Unstructured data extraction involves text processing, while structured data extraction focuses on tabular

data. The upsert process ensures that newly extracted data is efficiently integrated into the Pinecone vector database, with special mechanisms for handling both data types. The query module leverages advanced techniques to provide accurate and relevant responses to complex user queries.

Finally, the visual overview of the Lookinglass environment illustrates the user interface and key functionalities, emphasizing ease of document upload and efficient query handling. The system's design ensures that users can securely upload, categorize, and query documents, facilitating advanced data-driven decision-making in a business context.

4

Experiments and Results

In this chapter, we present the results of our comprehensive testing and experimentation on the Lookinglass Retrieval-Augmented Generation (RAG) system. The primary goal is to evaluate the system's effectiveness in enhancing language model capabilities through integrated information retrieval mechanisms. We conducted a series of tests to assess the accuracy of information retrieval, the quality of generated responses, performance under different conditions, and the system's ability to handle complex queries. Additionally, we evaluated data privacy and security measures to ensure sensitive information is adequately protected. These results will provide a thorough understanding of the system's strengths and areas for improvement, demonstrating its applicability in real-world business contexts.

4.1 Experimental Setup

In this section, we detail the experimental setup used to evaluate the Lookinglass Retrieval-Augmented Generation (RAG) system. This includes the hardware and software environment, the datasets, the types of queries, the methodology for measuring and recording results, and the steps taken to ensure reproducibility.

4.1.1 Hardware and Software Environment

The experiments were conducted using a system hosted on Digital Ocean, utilizing a microservice architecture. Each major component of the system (extraction, upsert, and query) was hosted on a dedicated endpoint. This setup ensures scalability and efficient handling of tasks by distributing the load across multiple services.

4.1.2 Datasets

To comprehensively evaluate the Lookinglass system, we used two types of datasets: unstructured and structured data.

1. Unstructured Data:

We selected 10 different types of unstructured data to simulate the diverse real-world applications of the system:

#	Documents Type
1	Research Paper
2	News Article
3	Technical Manual
4	Legal Document
5	Books
6	Blog Post
7	Corporate Report
8	Medical Record
9	Product Review
10	Email

Table 4.1.: Document Types

2. Structured Data:

We used 10 files consisting of both CSV and Excel formats, with some Excel files containing multiple sheets and requiring preprocessing to clean special characters and annotations. This setup was intended to test the system's ability to handle structured data effectively.

4.1.3 Types of Queries

We designed 90 questions, categorized into three levels of complexity to evaluate the system's performance comprehensively:

- **Simple Questions:**

  Focus on basic facts directly stated in the document.
  Example: "What is the main focus of the document?"[1]

- **Intermediate Questions:**

  Require contextual understanding and integration of multiple pieces of information.
  Example: "What is the significance of transformers in the development of LLMs?"[2]

- **Advanced Questions:**

  Involve in-depth analysis, synthesis, and critical thinking.
  Example: "Compare the architectural differences between encoder-decoder and causal decoder models in LLMs."[3]

4.1.4 Measurement and Recording of Results

The testing process involved sending each question as a query to the system using a Python script, storing the responses, and manually scrutinizing them for accuracy. Precision was calculated separately for simple, intermediate, and advanced questions. Given

[1] This questions is from [23]
[2] This questions is from [23]
[3] This questions is from [23]

Type of Questions	Number of Questions
Simple Questions	30
Intermediate Questions	30
Advanced Questions	30

Table 4.2.: Number of Questions for Each Type

the system's deployment in a real environment, manual examination of responses ensured a thorough evaluation.

Precision Calculation

The precision for each question type is calculated using the following formula:

$$\text{Precision}_{\text{type}} = \frac{\text{Number of Correct Answers}}{\text{Total Number of Questions}}$$

Where:

- $\text{Precision}_{\text{type}}$ is the precision for the specific question type (simple, intermediate, or advanced).
- Number of Correct Answers is the count of correct responses for the specific question type.
- Total Number of Questions is the total number of questions for the specific type (in this case, 30).

Since each question type has 30 questions, the precision for each type can be calculated as:

$$\text{Precision}_{\text{simple}} = \frac{\text{Number of Correct Answers}_{\text{simple}}}{30}$$

$$\text{Precision}_{\text{intermediate}} = \frac{\text{Number of Correct Answers}_{\text{intermediate}}}{30}$$

$$\text{Precision}_{\text{advanced}} = \frac{\text{Number of Correct Answers}_{\text{advanced}}}{30}$$

4.1.5 Steps to Ensure Reproducibility

To enhance the reproducibility of the experiments and account for the inherent nondeterminism of LLM-based applications like the Lookinglass, we implemented the following procedure:

1. Multiple Attempts per Question: Each question was sent to the Lookinglass five times. This approach helps in observing the consistency of the system's responses and mitigates the effects of any random variations in the output.
2. Recording Consistent Answers: For each of the five attempts, the system's response was recorded. We specifically tracked whether the response was correct or incorrect for each attempt.

3. Calculating Precision per Attempt: Precision was calculated separately for each of the five attempts.

4. Average Precision Calculation: After calculating precision for each of the five attempts, the average precision for each question type was determined by taking the mean of the five precision values. This provides a more robust measure of the system's performance, accounting for variability across multiple runs.

   The average precision for each question type is calculated as:

$$\text{Average Precision}_{\text{type}} = \frac{1}{5} \sum_{i=1}^{5} \text{Precision}_{\text{type, attempt}_i}$$

   Where:

   - Average $\text{Precision}_{\text{type}}$ is the average precision for the specific question type (simple, intermediate, or advanced).
   - $\text{Precision}_{\text{type, attempt}_i}$ is the precision for the i^{th} attempt.

By implementing this approach, we ensure that our evaluation of the Lookinglass system is comprehensive and reproducible. The repeated trials help to smooth out any anomalies and provide a clearer picture of the system's true performance.

4.2 Precision Results

In this section, we present the precision results for the Lookinglass Retrieval-Augmented Generation (RAG) system across different types of queries: simple, intermediate, and advanced, for both structured and unstructured data. Each question type was tested through five trials to ensure robustness and account for the system's variability. The table below shows the precision for each trial and the average precision for each question type, providing a clear assessment of the system's performance.

Unstructured Data

Question Type	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Average Precision**
Simple	0.97	0.97	0.94	0.97	0.94	0.96
Intermediate	0.94	0.97	0.94	0.94	0.90	0.94
Advanced	0.94	0.90	0.90	0.94	0.90	0.91

Table 4.3.: Precision for Each Question Type Across 5 Trials for Unstructured Data

Structured Data

The precision results in 4.4 and 4.3 indicate that the Lookinglass system performs consistently well across different types of queries, with higher precision observed for simpler queries. The variability across trials is minimal, demonstrating the system's robustness. However, advanced queries, particularly with structured data, showed slightly lower precision, highlighting an area for potential improvement. These results underscore the

Question Type	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Average Precision
Simple	0.97	0.97	0.97	0.94	0.97	0.97
Intermediate	0.94	0.97	0.97	0.94	0.90	0.94
Advanced	0.87	0.90	0.90	0.90	0.90	0.89

Table 4.4.: Precision for Each Question Type Across 5 Trials for Structured Data

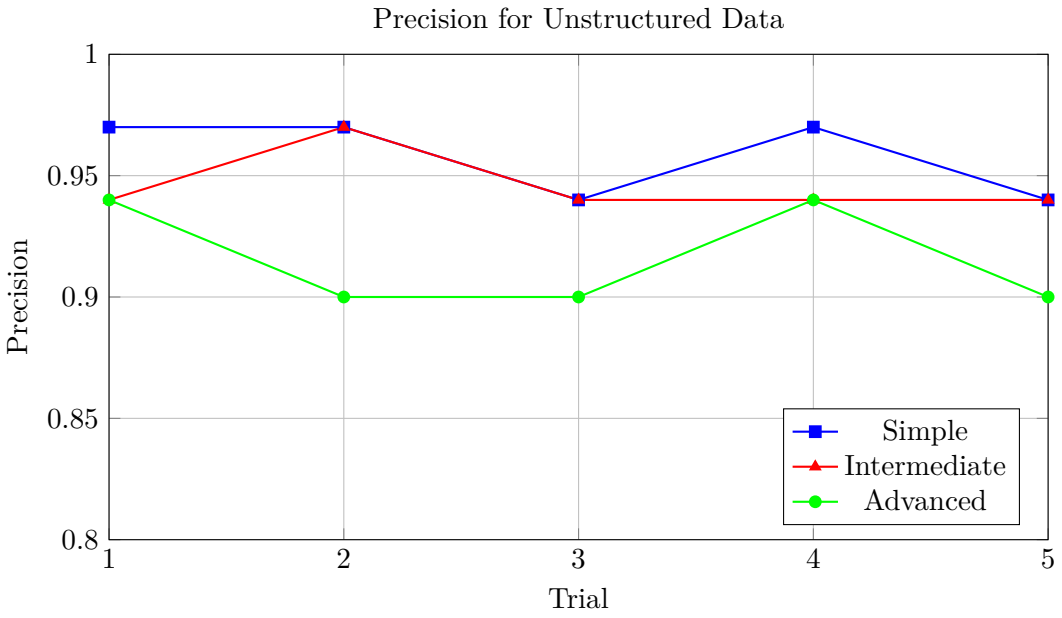

Figure 4.1.: Precision for Each Question Type Across 5 Trials for Unstructured Data According to 4.3

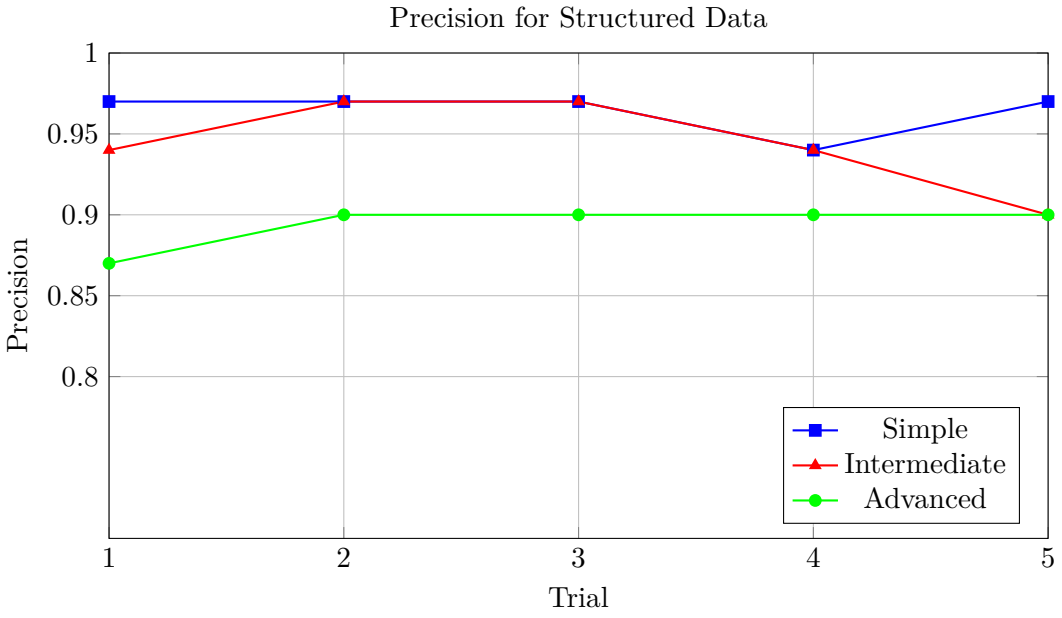

Figure 4.2.: Precision for Each Question Type Across 5 Trials for Structured Data According to 4.4

system's applicability in real-world scenarios, while also providing a direction for further refinement.

4.3 Summary

In this chapter, we presented a comprehensive evaluation of the Lookinglass Retrieval-Augmented Generation (RAG) system. Our experiments were designed to assess the system's effectiveness in enhancing language model capabilities through integrated information retrieval mechanisms. We conducted tests using both structured and unstructured datasets, covering various types of documents and queries of differing complexities. The results demonstrated the system's robust performance in terms of precision across multiple trials, with consistent accuracy for simple and intermediate queries, and satisfactory precision for advanced queries. Additionally, we ensured the reproducibility of our experiments by repeating each query multiple times and calculating average precision. These findings provide a clear understanding of the system's strengths and potential areas for improvement, affirming its applicability in real-world business contexts and guiding future development efforts.

5

Conclusions

This thesis successfully developed the Lookinglass Retrieval-Augmented Generation (RAG) system, enhancing language model capabilities through advanced information retrieval mechanisms. The system demonstrated significant improvements in generating accurate, reliable responses, handling complex queries, and ensuring data privacy. Future work will focus on integrating real-time data augmentation, implementing automated agent functionality. These advancements will further solidify Lookinglass as a powerful tool for data-driven decision-making in various business contexts.

5.1 Future work

While the Lookinglass system has shown promising results, there are several areas for future improvement and expansion to further enhance its capabilities:

1. **Realtime Data Augmentation:**

   Integrating real-time data augmentation capabilities would allow users to access updated information, such as recent laws or live updates from the stock market, directly through the system. This could be achieved by developing APIs that connect to real-time data sources. For example, integrating financial APIs for live stock market data or legal databases for the latest legislative changes. The system would then retrieve, process, and integrate this data into the response generation process in real-time.

2. **Automated Agent Functionality:**

   Implementing automated agent functionality where users can set predefined jobs that the system executes independently, notifying users upon completion. This would enhance user convenience and efficiency.

   Develop a job scheduling module that allows users to define tasks and set criteria for execution. The system would monitor these criteria, execute the tasks as conditions are met, and notify users via their preferred communication channels (e.g., email, SMS) once the tasks are completed. For instance, a user could set a job to monitor stock prices and perform calculations, with the system delivering the final analysis when ready.

By addressing these future work areas, the Lookinglass system can further solidify its position as a powerful tool for data-driven decision-making, offering users even more robust, accurate, and timely information across various domains and applications.

5.2 Conclusion

The objective of this thesis was to develop and enhance the Retrieval-Augmented Generation (RAG) system, Lookinglass, to improve the accuracy and reliability of responses generated by language models. The research focused on integrating information retrieval mechanisms to address the limitations of traditional language models, particularly in business contexts.

Throughout this thesis, we have explored the fundamental aspects of RAG systems, from the importance of data retrieval and generation to the rise and limitations of Large Language Models (LLMs). We designed and implemented a comprehensive system architecture for Lookinglass, incorporating advanced data extraction, upsertion, and querying techniques. Our experiments and results demonstrated the system's ability to handle complex queries.

Key findings from our research include:

- **Enhanced Language Model Capabilities:** By integrating advanced retrieval mechanisms, Lookinglass significantly reduced instances of hallucinations, ensuring the generation of factually accurate and reliable responses.
- **Improved Handling of Complex Queries:** The system effectively managed long-tail data and multi-hop queries, providing accurate responses to complex business inquiries.
- **Robust Privacy Measures:** The implementation of strong data handling and retrieval processes ensured the confidentiality of proprietary and private datasets.
- **Practical Business Applications:** The system proved versatile and applicable across various business tasks, including document-based question answering, information retrieval, and content generation.

Overall, the Lookinglass RAG system demonstrated its potential to enhance data-driven decision-making in business environments by providing precise and relevant information quickly and efficiently.

A

Appendix

In this appendix, additional images from the system environment are presented to provide a comprehensive visual overview of the system's interface and functionalities.

Query: Give me map of the world with its according number of internet users. A.1

Query: Is there a connection between gender and age in the deaths from the Titanic disaster? A.2

Figure A.1.: Response to a query from structured data, including the visualization diagram and the sources from which the data was extracted.

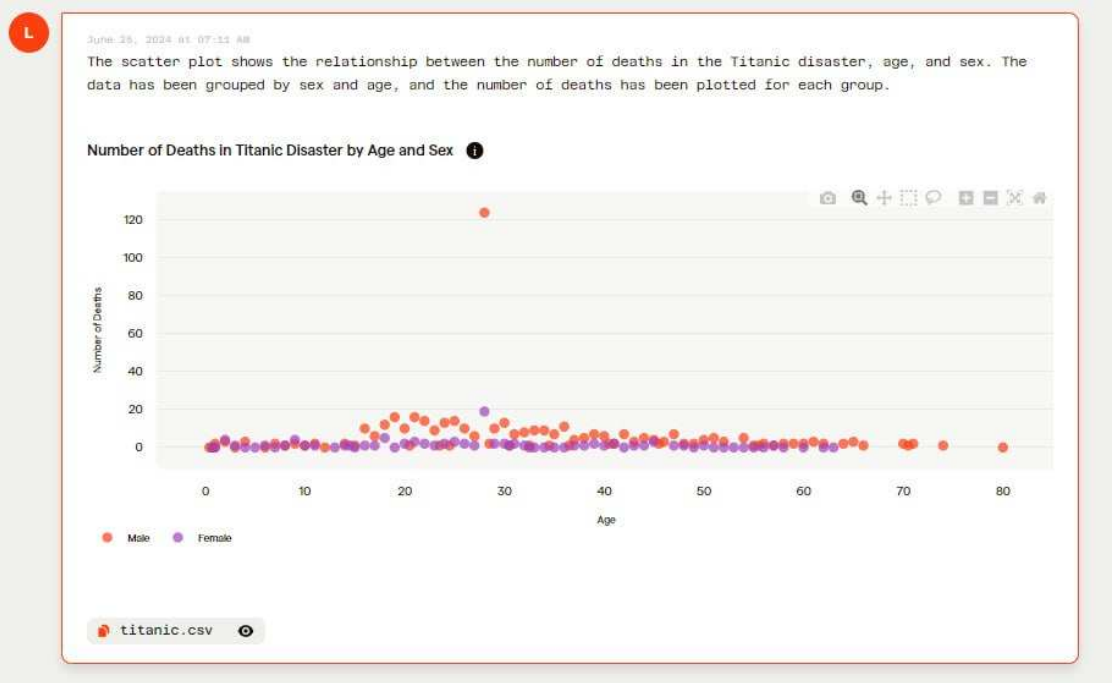

Figure A.2.: Response to a query from structured data, including the visualization diagram and the sources from which the data was extracted.

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[3] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui, "Retrieval-augmented generation for ai-generated content: A survey," 2024.

[4] X. Yang, M. Ye, Q. You, and F. Ma, "Writing by memorizing: Hierarchical retrieval-based medical report generation," 2021.

[5] Z. Levonian, C. Li, W. Zhu, A. Gade, O. Henkel, M.-E. Postle, and W. Xing, "Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference," 2023.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[7] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021.

[8] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," 2023.

[9] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," 2023.

[10] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," 2022.

[11] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023.

[12] F. Zhang, B. Chen, Y. Zhang, J. Keung, J. Liu, D. Zan, Y. Mao, J.-G. Lou, and W. Chen, "Repocoder: Repository-level code completion through iterative retrieval and generation," 2023.

[13] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," 2022.

[14] J. Kim and M. Min, "From rag to qa-rag: Integrating generative ai for pharmaceutical regulatory compliance process," 2024.

[15] S. Kresevic, M. Giuffrè, M. Ajcevic *et al.*, "Optimization of hepatological clinical guidelines interpretation by large language models: a retrieval augmented generation-based framework," *npj Digital Medicine*, vol. 7, p. 102, 2024.

[16] A. J. Yepes, Y. You, J. Milczek, S. Laverde, and R. Li, "Financial report chunking for effective retrieval augmented generation," 2024.

[17] Z. Xu, M. J. Cruz, M. Guevara, T. Wang, M. Deshpande, X. Wang, and Z. Li, "Retrieval-augmented generation with knowledge graphs for customer service question answering," *arXiv preprint arXiv:2404.17723*, 2024. [Online]. Available: https://arxiv.org/abs/2404.17723

[18] S. Roychowdhury, M. Krema, A. Mahammad, B. Moore, A. Mukherjee, and P. Prakashchandra, "Eratta: Extreme rag for table to answers with large language models," 2024.

[19] J. F. Hurtado, "Harnessing retrieval-augmented generation (rag) for uncovering knowledge gaps," 2023.

[20] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, "Evaluation of retrieval-augmented generation: A survey," 2024.

[21] Y. Tang and Y. Yang, "Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries," 2024.

[22] J. Hsia, A. Shaikh, Z. Wang, and G. Neubig, "Ragged: Towards informed design of retrieval augmented generation systems," 2024.

[23] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024.

[24] OpenAI. (2023) Openai embeddings use cases. Accessed: 2024-06-15. [Online]. Available: https://platform.openai.com/docs/guides/embeddings/use-cases

Acknowledgements

I would like to express my deepest gratitude to Professor Loris Nanni for accepting me as his student and providing me with the opportunity to pursue my research. His endorsement was invaluable in enabling me to embark on this academic journey, and I am profoundly thankful for his support in this regard.

I would also like to extend my heartfelt thanks to Dr. Sina Lessani Bahri, who supervised me tirelessly at Impssbl company. His guidance, expertise, and mentorship have been instrumental in my learning and growth throughout this project. I am equally grateful to Alessandro Botteon, the CEO of Impssbl, for providing me with this incredible opportunity, without which this project would not have been possible. A special thanks to my wonderful family—my mother, father, and brother—whose unwavering love and support have been a constant source of strength and inspiration throughout my journey. Lastly, I am thankful to all my friends who have directly and indirectly helped me throughout this project.