



DEPARTMENT OF MATHEMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mathematics

**Toward Optimising a Retrieval Augmented
Generation Pipeline using Large Language
Model**

Gentrit Fazlija





DEPARTMENT OF MATHEMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Mathematics

**Toward Optimising a Retrieval Augmented
Generation Pipeline using Large Language
Model**


**Optimierung einer Retrieval Augmented
Generation Pipeline unter Verwendung eines
großen Sprachmodells**

Author:	Gentrit Fazlija
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Anum Afzal
Submission Date:	15.03.2024



I confirm that this master's thesis in mathematics is my own work and I have documented all sources and material used.

Munich, 15.03.2024


Gentrit Fazlija

Abstract

This thesis introduces a Modular Retrieval Augmented Generation (RAG) framework, a response to the growing trend where over 60% of companies integrate RAG into their operations. Focusing on the incorporation of four specific modules—Multi-Query, Child-Parent-Retriever, Ensemble Retriever, and In-Context-Learning—the study aims to enhance the functionality and performance of RAG systems for academic data retrieval, specifically targeting over 150 study programs at Technical Universität München. A significant contribution of this research is the introduction of a novel evaluation approach, the RAG Confusion Matrix, designed to assess the effectiveness of various module combinations within the Modular RAG framework. By exploring the integration of both open-source (e.g., Llama 2, Mistral, Mini Orca, Vicuna) and closed-source (GPT-3.5 and GPT-4) Large Language Models, this thesis seeks to identify optimal configurations that improve query interpretation, data retrieval accuracy, and context-based response adaptation, offering valuable insights into the application and optimization of RAG frameworks in educational and business contexts.

Contents

Abstract	iii
1. Introduction	1
2. Related Work	2
3. Background	5
3.1. Large Language Models	5
3.1.1. Embedding	6
3.2. Retrieval Augmented Generation	7
3.2.1. Naive RAG	8
3.2.2. Advanced RAG	9
3.2.3. Modular RAG	12
3.3. Evaluation	19
3.3.1. RAG Evaluation	20
4. Corpora	23
4.1. TUM Studyprogram Corpora	23
4.1.1. Official TUM website	25
4.1.2. Individual Faculty website	26
4.2. Question-Answer Set	27
4.3. Evaluation Set	28
5. Methodology	31
5.1. Models	31
5.1.1. Open-Sourced Models	32
5.1.2. Closed-Sourced Models	33
5.2. Pre-Retrieval Phase	34
5.2.1. Multi Query	35
5.3. Retrieval Phase	36
5.3.1. Child Parent Retriever	37
5.3.2. Ensemble Retriever	37
5.4. Generation Phase	39
5.4.1. In-Context-Learning	39
5.4.2. Parsing Meta-Data	40
5.5. User Interface	43

5.6. Evaluation	43
5.6.1. RAG Confusion Matrix	43
6. Evaluation Results	47
6.1. Retrieval Quality	47
6.1.1. Hit Rate	48
6.2. Generation Quality	51
6.2.1. RAG Confusion Matrix	52
6.2.2. RAG AVG-Metric Evaluation	57
6.3. RAG Framework Enhancement	61
6.3.1. RAG Confusion Matrix	64
6.3.2. RAG AVG-Metric Evaluation	66
7. Conclusion and Future Work	68
A. Appendix: Evaluation Metrics for Generation Quality	70
List of Tables	72
Bibliography	73

1. Introduction

Since the 1950s, with the introduction of n-grams, scientists have been attempting to create Language Models capable of interacting with humans using natural language. A significant breakthrough came in 2017 with the publication of "Attention is All You Need," which introduced the Transformer architecture, laying the groundwork for the development of Large Language Models (LLMs). These models have demonstrated exceptional capabilities in generating human-like text, understanding language, and translating between languages.

Businesses quickly took notice, eager to harness the power of LLMs for their operations. However, the use of LLMs in a business context is not straightforward, as these models are predominantly trained on publicly available data, making them less suited for specific business needs. This challenge led to the emergence of Retrieval Augmented Generation (RAG) frameworks, which offer a solution by enabling the integration of company-specific data with the expansive knowledge base of LLMs, thereby enhancing their utility and relevance in business applications. A recent study by Databricks underscores the growing reliance on RAG frameworks, revealing that approximately 60% of such applications employ some form of RAG to enhance performance and relevance. [1]

In this thesis, we aim to construct a RAG framework equipped with a user interface to enable efficient interaction with data depicting over 150 study programs at Technical Universität München, catering to both existing and potential students. This framework integrates several modules, each contributing uniquely to the functionality and performance of the RAG system, which will be assessed using a newly developed RAG Confusion Matrix.

The background section introduces Naive RAG—a basic version without extra data handling—and explains the roles of modules in our Modular RAG framework. We incorporate Multi-Query for improved query interpretation, Child-Parent-Retriever for better hierarchical data retrieval, Ensemble Retriever for combining BM25 and dense retrievers, and In-Context-Learning for context-based response adaptation. Additionally, our development uses various LLMs, including open-source models Llama 2 7B, Llama 2 13B, Mistral 7B, Mini Orca 7B, and Vicuna 7B, and closed-source models GPT 3.5 and GPT 4.

The methodology section of the thesis outlines the strategic decisions made throughout the framework's development, detailing the selection and integration of the various modules. The evaluation chapter presents an analysis of the framework's performance, focusing on how different combinations of modules and iterations of LLMs influence the effectiveness of the RAG system. Through this comprehensive approach, the thesis aims to identify the most efficient configurations, thereby contributing valuable insights into the application of RAG frameworks in enhancing information retrieval and generation.

2. Related Work

Early approaches to RAG involved relatively simple methods of retrieving relevant documents based on the query and then processing these documents to generate a response. However, recent advancements have seen more sophisticated integration of retrieval and generation processes. Notable implementations of RAG, such as those by Lewis et al., have demonstrated the ability to dynamically retrieve and incorporate relevant information during the generation process, thereby significantly enhancing the quality and relevance of the generated text. These advancements have been facilitated by improvements in both the retrieval mechanisms, which have become more efficient and effective at finding relevant information, and the generative models, which have become better at integrating and contextualizing the retrieved information.

Beyond question answering, RAG is also used in content creation tasks, including article writing, code generation, and creative storytelling and many more. By accessing a diverse range of sources, RAG models can generate content that is not only relevant and informative but also rich in detail and variety. For instance, in content creation, RAG can draw upon various articles, blogs, and databases to produce comprehensive and nuanced articles on topics ranging from environmental science to geopolitical analyses. [2, 3]

Over the last months, notable RAG examples have emerged. Some of which are Google's RETRO (Retrieval-Enhanced Transformer), Forward-Looking Active Retrieval Augmented Generation (FLARE), and System 2 Attention (S2A). Google's RETRO represents an advancement in leveraging external knowledge bases by integrating a retrieval component into the Transformer architecture. This enables the model to dynamically incorporate relevant information from a vast corpus during the generation process, resulting in more accurate and information-rich text generation across various tasks, ranging from complex question answering to content creation with nuanced context. [4]

FLARE, on the other hand, harnesses the internet as a dynamic knowledge base to ensure the accuracy and timeliness of responses. Through an iterative process, FLARE uses the initial response from the LLM to query internet searches, integrating fresh information into the answer. This approach underscores the importance of accessing real-time data to enhance LLM outputs, particularly for questions requiring up-to-date information. [5] Similarly, S2A, developed by META, focuses on refining the context within RAG systems by regenerating it to remove irrelevant details while retaining essential information. By instructing the LLM to reconstruct its context, S2A ensures that only pertinent elements are considered, marking a significant step towards improving the relevance and clarity of responses generated by RAG systems. [6]

Multiple Multimodal RAG frameworks utilize CLIP embeddings to integrate text and image data, enabling the retrieval of relevant multimodal contexts. This framework synthesizes

responses using a multimodal model, demonstrating the benefits of combining textual and visual information. The incorporation of both data types showcases the potential for multimodal approaches to enrich and enhance the accuracy of RAG-generated responses.

Considering the diversity of RAG pipelines, a clear insight emerges: a RAG framework is inherently a synthesis of various components, collaboratively functioning to achieve a singular objective. Consequently, altering even a minor detail within any component of the RAG framework can result in significant modifications to the entire system. Thus, a critical aspect of research in this area involves discerning the specific impact each component has on the overall framework. This analysis will be a focal point throughout this thesis.

The evaluation of RAG systems necessitates a comprehensive approach that transcends conventional language model evaluation metrics, incorporating both automated metrics and human assessment to capture the nuanced performance of these hybrid models. Automated metrics such as BLEU, ROUGE, and F1 scores serve to quantitatively assess aspects like fluency, coherence, and the accuracy of generated text against reference standards. However, these metrics, while useful for scalability and initial benchmarking, often fall short in capturing the full complexity of tasks RAG systems are deployed for, particularly in evaluating the relevance and integration of retrieved information within the generated content. Given that LLMs like ChatGPT achieve superior performance across these metrics, their effectiveness raises questions about the utility of traditional scores, suggesting that they may no longer be as indicative of true performance in the nuanced contexts RAG systems are designed for.

In the evaluation of RAG systems, our focus narrows down to two critical aspects: retrieval metrics and evaluation metrics. Retrieval metrics, including Hit Rate, Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR), provide quantitative measures of a system's ability to retrieve relevant documents. Evaluation metrics, on the other hand, can involve human judgment or the comparison of outputs from different models, often using a larger model to assess the output of a smaller one. [7]

Hit Rate measures the proportion of queries for which the correct answer is retrieved within the top-k results, serving as an indicator of the system's effectiveness in identifying relevant answers quickly. Mean Reciprocal Rank (MRR) offers insight into the average rank position of the first relevant document across queries, with a higher MRR reflecting better performance in surfacing relevant documents at the top of the retrieval list. Mean Average Precision (MAP) extends this by evaluating the precision of retrieval across all ranks, emphasizing the importance of the order in which relevant documents appear, with higher values indicating better performance in ranking relevant documents more highly.

The distinction between retrieval and evaluation metrics is crucial for assessing RAG systems comprehensively. Retrieval metrics (Hit Rate, MRR, MAP) focus on the system's ability to find and rank relevant information accurately, assessing effectiveness and precision in document retrieval. Conversely, evaluation metrics provide a broader perspective, leveraging human judgment or comparisons with larger models to gauge the quality of generated responses. This dual approach ensures a balanced evaluation of a RAG system's retrieval efficiency and the relevance and accuracy of its outputs.

The Technical University of Munich has recently undertaken a project to develop a chat-

bot designed to address student inquiries effectively. Launched during the "TUM MGT @ Microsoft" hackathon in December 2023, this initiative brought together student teams to create a RAG chatbot. This development aims to streamline the management of the estimated 600 weekly emails received by the TUM School of Management's Student Support, thereby enhancing efficiency and providing timely responses to students. The successful implementation of this chatbot signifies TUM's commitment to leveraging artificial intelligence for improving administrative processes and student support services. The author's participation in this project underscores its relevance and contribution to the exploration of AI applications in educational contexts. [8]

3. Background

3.1. Large Language Models

The evolution of Large Language Models (LLMs) represents a significant trajectory in the field of artificial intelligence, tracing back to the early 2000s with the introduction of neural probabilistic language models. [9] These foundational models, which utilized neural networks to predict the next word in a sequence, set the stage for the development of more complex systems. Key milestones in this journey include the introduction of sequence-to-sequence (seq2seq) models [10] which revolutionized machine translation by encoding a source sentence and decoding it into a target language, and the advent of attention mechanisms that allowed models to focus on relevant parts of the input sequence for better context understanding.

Key milestones in LLM research have significantly shaped the landscape of artificial intelligence and natural language processing. The introduction of the Transformer architecture by Vaswani et al. in 2017 marked a pivotal breakthrough, introducing a model based solely on attention mechanisms, eliminating the need for recurrent layers. This innovation not only improved the efficiency and effectiveness of language models but also paved the way for subsequent advancements. The development of BERT (Bidirectional Encoder Representations from Transformers) by Devlin et al. further exemplified the power of pre-training, enabling models to understand context from both left and right of a word, a notable departure from previous models that processed text in a unidirectional manner. These innovations have significantly improved models' understanding of language nuances and context, setting new standards for performance across various language tasks.

Following these foundational advancements, the GPT series by OpenAI took the spotlight, showcasing the potential of scaling up Transformer models. GPT-3, with its 175 billion parameters, pushed the boundaries of language models' capabilities, demonstrating an ability to perform a wide range of tasks without task-specific training. This leap in model size and capability highlighted the scalability of Transformer architectures and their impact on the field's understanding of model capacity, generalization, and task adaptability. The progression from GPT-1 to GPT-3 reflects a broader trend in LLM research towards larger, more powerful models capable of increasingly sophisticated language understanding and generation, a trend that continues to drive innovation in AI research and application.

Regarding LLMs, it's widely acknowledged that larger and more complex models are perceived as more capable. An important advancement in optimizing the transformer architecture is the implementation of a technique known as a Mixture of Experts (MoE), which essentially allows the model to dynamically select from a pool of specialized sub-models (the "experts") for processing different parts of the input data. This optimization enables more efficient computation and potentially greater scalability by leveraging the

strengths of individual experts for specific tasks or data types, thereby enhancing the overall performance and flexibility of LLMs.

Additionally, a distinction can be made between Closed-Sourced and Open-Sourced models, with the Llama series standing out as a prominent example of open-source LLMs that have achieved remarkable results and serve as foundational components for other open LLMs. The key difference lies in the accessibility of the models' weights; open-source models allow for the inspection and local or cloud-based fine-tuning of their weights. This approach only incurs the cost of electricity, offering transparency in data flow. This distinction will be crucial, as both Open-Sourced and Closed-Sourced models will be employed in the methodology chapter of this thesis.

3.1.1. Embedding

In the intricate process of training LLM from the ground up, embedding spaces serve a critical function. Initially, the model's architecture is designed such that its weights encapsulate the rich tapestry of linguistic and contextual nuances. Specifically, within the transformer architecture, the embedding step transforms sentences into vector representations, facilitating the model's understanding and manipulation of language. This mechanism is particularly vital in the context of RAG frameworks, where embedding spaces assume a unique and enhanced role. Unlike general LLM operations, a RAG framework incorporates a bespoke embedding space, often termed a vector store, pre-populated with domain-specific data in vectorized form. This vector store is integral during the retrieval step of the RAG process, where it enables the precise selection of vectors that encapsulate relevant contextual information. These selected vectors are then utilized in the generation part of the framework, guiding the LLM to produce responses that are not only pertinent but also richly informed by the domain-specific context previously retrieved. This delineation between the retrieval and generation phases underscores the efficiency and specificity that embedding spaces bring to the RAG framework, ensuring that the generated content is both relevant and contextually nuanced.

Embeddings are a foundational technique in natural language processing (NLP) that convert words, phrases, or even entire documents into numerical vectors of a fixed dimensionality, allowing machines to process and understand language. By representing textual elements in a high-dimensional space, these embeddings enable various applications, notably in search functionalities, where the semantic similarity between the query and potential results is often measured using cosine similarity among their vector representations. Although cosine similarity is a prevalent metric due to its effectiveness in identifying angular proximity between vectors, other distance metrics can also be employed depending on the specific requirements of the task. The dimensionality of these embedding spaces varies, with common sizes being 768 or 1536 dimensions. This variation in dimensions reflects a balance between capturing enough semantic detail and computational efficiency.

The creation of embeddings involves mapping words or phrases to vectors in such a way that the semantic relationship between words is reflected in the geometric arrangement of their vectors. A typical method involves using an "anchor" word and comparing it with other

words that are either similar or dissimilar, leading to a process known as contrastive learning. During this process, the model is trained to bring the vectors of similar words closer together and push those of dissimilar words further apart in the embedding space. This training can be accomplished through various algorithms, including neural network models that adjust the vector representations based on the context in which words appear, thereby encapsulating both the syntactic and semantic nuances of language use. Through iterative adjustments during the training phase, the model learns to accurately embed words in a way that mirrors their real-world usage and relationships. [11]

3.2. Retrieval Augmented Generation

RAG combines the generative capabilities of LLMs with information retrieval techniques to enhance the model's ability to generate responses based on a wider range of knowledge than what is contained in its parameters alone. This approach significantly improves the model's performance on tasks requiring specific factual information or domain-specific knowledge, marking a pivotal development in the field of artificial intelligence and natural language processing. The significance of RAG lies in its hybrid nature, leveraging both the depth of pre-trained models for language understanding and generation, and the breadth of external databases or documents to pull in precise information. The pre-trained model acts as a foundation, utilizing its understanding of basic semantic structuring and patterns to grasp the context of queries. Meanwhile, the external data, when provided as context to the LLM, serves as the ground truth, enabling the generation of answers that are not only contextually appropriate but also factually accurate, particularly in question-answering scenarios. This dual approach allows RAG systems to deliver highly relevant and accurate information by combining the LLM's broad linguistic capabilities with the specific, detailed knowledge contained within external data sources.

The conceptual framework of a RAG system can be effectively visualized as being partitioned into three distinct parts: Ingestion, Retrieval, and Generation. Each of these components plays a pivotal role in the overall functioning of the RAG pipeline, and they can each be engineered with varying degrees of complexity. The Ingestion part is responsible for assimilating and processing the initial input data, preparing it for subsequent retrieval. The Retrieval part then takes over, utilizing the processed data to fetch the most relevant information from a vast repository of knowledge. Finally, the Generation part synthesizes the retrieved information to produce coherent and contextually relevant responses. Moreover, the RAG framework includes specific connection blocks that facilitate seamless interaction between these components, such as linking the Ingestion part with the Retrieval part, ensuring a fluid and efficient flow of information throughout the system.

In the subsequent chapters, this thesis will explore both a naive, or simple, RAG approach and an advanced RAG approach, highlighting the variability and adaptability of the RAG framework to different levels of complexity and sophistication. By the end of this chapter, it will become evident that the components of a RAG system are inherently modular, allowing for the addition or discarding of parts as required. This modularity is a critical feature that

will be leveraged in the methodology chapter, where different modules are connected, and their combined output is evaluated. This exploration underscores the flexibility of the RAG architecture, illustrating how it can be customized to meet specific requirements or objectives, thereby enhancing the system's effectiveness and applicability across a range of tasks and settings.

3.2.1. Naive RAG

In the realm of building a simple RAG, the concept of a naive RAG represents an elementary yet foundational approach, gaining traction shortly after the widespread adoption of technologies like ChatGPT. Characterized by its straightforward "Retrieve-Read" framework, the naive RAG delineates a basic structure comprising ingestion, retrieval, and generation phases, providing a scaffold for more complex RAG implementations. [12]

The first phase, known as digestion within the context of naive RAG, plays a pivotal role in the preparation of data for subsequent retrieval and generation tasks. This process unfolds offline and encompasses several key stages aimed at standardizing and optimizing the data for efficient processing. Initially, the digestion phase involves the cleaning and extraction of data from diverse formats, including PDF, HTML, Word, and Markdown, converting them into a uniform plain text format. To accommodate the context limitations inherent in language models, this text is further segmented into smaller, manageable chunks, a process referred to as chunking. These chunks are then encoded into vector representations using an embedding model. This transformation is crucial for enabling similarity comparisons during the retrieval stage. Completing the digestion process, a vector store is constructed to store these text chunks and their corresponding vector embeddings as key-value pairs, thereby laying the groundwork for a scalable and efficient search capability essential for the retrieval phase.

Upon receiving a user query, the naive RAG system leverages the encoding model previously used during the digestion phase to convert the input into a comparable vector representation. This step is critical for aligning the query with the pre-processed data structure of the RAG system. The system then calculates similarity scores between the query vector and the vectorized chunks stored in the indexed corpus, identifying the top K chunks that exhibit the highest similarity to the query. These selected chunks are not just retrieved; they are expanded upon to serve as a contextual foundation for formulating a response to the user's request.

Following the retrieval of these topically relevant chunks, the naive RAG system embarks on the generation phase. Here, the original query and the retrieved documents are amalgamated into a unified prompt, setting the stage for a large language model to craft an appropriate response. The model's response strategy may vary, influenced by task-specific requirements. Furthermore, in scenarios involving ongoing dialogues, the system has the capability to incorporate any existing conversational history into the prompt. This allows for a seamless continuation of multi-turn dialogue interactions, ensuring that responses are not only contextually accurate but also coherent within the broader scope of the conversation.

The naive RAG implementation, despite its innovative approach to blending retrieval and generation processes, grapples with considerable challenges across "Retrieval," "Generation,"

and "Augmentation" facets. In retrieval, issues like low precision may result in the selection of misaligned chunks, diminishing retrieval quality and potentially leading to hallucinations or narrative discontinuity. Compounded by low recall, the system might not gather all important information, restricting the LLM's capacity for generating detailed and accurate responses. During generation, the model faces hurdles in maintaining output relevance and coherence, with risks of producing content that is irrelevant, toxic, or biased due to the model's limitations in content refinement. The augmentation phase, crucial for weaving retrieved context into the generation task, also presents challenges, especially when navigating redundancy in similar passages, which risks producing repetitive and bloated content. Achieving a seamless integration of diverse passages and aligning writing styles and tones for consistent output further complicates the generation process, underscoring the complexity of creating engaging and coherent responses within the naive RAG framework.

3.2.2. Advanced RAG

The Advanced RAG framework represents a significant evolution from its Naive counterpart, designed to specifically address and rectify the limitations identified in the earlier model and more. However, up to this date, there is no specific configuration of RAG modules that universally defines what constitutes an Advanced RAG framework. Various sources highlight improvements over the shortcomings of the naive RAG approach by enhancing different segments of the RAG pipeline at specific junctures. Yet, since these enhancements are often incremental and scattered across different aspects of the framework, it's challenging to pinpoint a singular permutation of modules within the RAG architecture that could be distinctly recognized as an Advanced RAG. These modifications, while individually contributing to the overall efficacy and sophistication of RAG systems, do not coalesce into a standardized advancement that can be uniformly labeled as 'Advanced RAG.' This situation underscores the dynamic and evolving nature of RAG development, where continuous, iterative improvements are made in pursuit of optimizing performance and relevance, rather than achieving a definitive, one-size-fits-all enhancement model.

Given the diversity of data structures, domains, and levels of complexity encountered in various applications, the integration and implementation of different RAG frameworks become essential. This diversity necessitates a flexible approach to the design and evaluation of RAG systems, allowing them to adapt and perform optimally across varied contexts. Such an approach involves a top-down conceptualization of the RAG framework as comprising interconnected building blocks — Digestion, Retrieval, and Generation. The implementation of specific modules within these phases is crucial for tailoring the framework to the unique demands of different data landscapes, thereby ensuring a smooth flow of information and effective processing across the entirety of the RAG system.

In this section, we will embark on a top-level overview, exploring the potential enhancements to the Naive RAG framework. This exploration aims to identify areas of improvement that can further refine and elevate the system's capabilities. Subsequently, in the next chapter "Modular RAG," we delve deeper into the granular details of the specific modules comprising the building blocks of the RAG architecture.

At the heart of these enhancements is the digestion process, focusing primarily on optimizing data indexing to improve the content’s quality before it even enters the retrieval phase. This optimization encompasses several strategies: enhancing data granularity, optimizing digestion structures, adding metadata, alignment optimization, and mixed retrieval methods to name a few. Enhancing data granularity ensures the text is standardized, consistent, factually accurate, and context-rich, which involves removing irrelevant data, clarifying ambiguities, and updating outdated documents. Optimizing digestion structures entails adjusting chunk sizes to better capture relevant contexts and leveraging graph structures to exploit relationships between data nodes, thereby enriching the context available for retrieval. The addition of metadata to chunks enhances filtering capabilities and retrieval efficiency by providing additional contextual clues.

When looking at the connection between the digestion and retrieval stage we see significant enhancements through the implementation of dynamic and fine-tuned embeddings, each aimed at refining the system’s ability to identify and utilize the most relevant context. Dynamic embeddings represent a pivotal improvement, offering sensitivity to the contextual nuances of language by adjusting the vector representation of words based on their surrounding context. This approach allows for a richer, more precise understanding of queries, as the meaning of words can shift dramatically depending on their use, thus ensuring that the retrieval process is informed by the nuanced meanings words acquire in different settings. Following this, fine-tuning embeddings emerges as a crucial strategy for tailoring the retrieval process to specific domain needs. By customizing embedding models, the system can significantly enhance the relevance of retrieved content, particularly in specialized fields where terms may evolve rapidly or be uncommon. This process involves adjusting the embeddings based on domain-specific data, ensuring that the system can effectively handle a wide range of queries with high precision, even in the face of challenging, dynamic vocabularies.

Within the Advanced RAG framework, the retrieval process encounters notable improvements, particularly addressing two main challenges inherent in traditional approaches. Firstly, the reliance on dot or cosine similarity for matching query vectors with ingested document vectors in the multi-dimensional space underpins a significant limitation. This method presupposes a semantic similarity between the question and the potential answers, an assumption that does not always hold true. Various factors can disrupt this presumed similarity: the contextual knowledge required to understand the query may be distributed across multiple vectors, necessitating their combination for a complete answer; or simply, the semantic relationship between the question and its answer may not be direct, making it challenging for the cosine similarity approach to accurately retrieve the most relevant documents. Additionally, innovative retrieval mechanisms, such as the ColBERT model, explore beyond simple dot or cosine similarity by introducing a late interaction architecture. This approach allows for the independent encoding of queries and documents, followed by a cost-effective interaction step to model fine-grained similarity, demonstrating the potential for more efficient and nuanced retrieval strategies that capitalize on the expressiveness of deep language models while significantly reducing computational demands. [13]

The second challenge concerns the creation of vector representations during the inges-

tion phase. This process involves a delicate balance between the breadth of information encapsulated within a single vector and the precision of the information it represents. When a vector is tasked with representing a vast array of information, there's a risk of diluting its relevance with noise, making it less effective for precise retrieval. Conversely, vectors representing too narrow a scope of information may lack the necessary contextual breadth, failing to capture nuanced relationships or broader contextual cues essential for accurate retrieval. These shortcomings highlight the complexity of optimizing the retrieval process within the RAG framework, underscoring the need for advanced strategies that can navigate the intricacies of semantic similarity and vector representation to enhance the accuracy and relevance of retrieved content.

Upon retrieving contextually relevant information from the database, integrating this data with the user query into LLMs poses significant challenges, not just due to context window limitations, but also concerning the effective use of long contexts. Even if it were possible to bypass these limitations and input all retrieved documents directly into the LLM, two main issues remain closely intertwined. Firstly, the inclusion of a large volume of retrieved information risks introducing substantial noise alongside the relevant data. Without proper compression or summarization, this noise can significantly hinder the model's ability to identify and focus on the crucial information needed for generating accurate responses. Secondly, as highlighted by recent research, language models exhibit limitations in effectively utilizing long input contexts. Performance notably deteriorates when the position of relevant information shifts within the input, particularly for information located in the middle of long contexts. This phenomenon suggests that, despite their capacity to process extensive contexts, current language models do not robustly leverage the entirety of their input, showing a preference for information presented at the beginning or end.

During the generation phase of integrating contextually relevant information into LLMs, significant challenges arise, notably in generating concise and accurate responses to user queries. Two primary issues are prevalent in this phase. The first is the occurrence of hallucination when the model fails to retrieve the desired chunks of information, leading to the generation of content not grounded in the retrieved data or significantly divergent from the provided context. This issue is fundamentally different from instances where the model provides inaccurate or incorrect responses due to misinterpretation or misuse of correct information within the context window, highlighting a discrepancy in the model's processing rather than an absence of relevant data. This nuance will be further addressed inside the chapter RAG Evaluation.

The second challenge is ensuring the consistency and satisfaction of the model's responses. Addressing this requires an effective strategy like in-context learning, which involves furnishing the model with examples of the expected output or behavior directly within the input. This approach not only "shows" the LLM the anticipated quality and relevance of responses but also aids in mitigating issues of hallucination and inconsistency. By demonstrating desired outputs, in-context learning becomes a crucial mechanism for harmonizing the generation capabilities of LLMs with the nuanced requirements of specific tasks, thereby enhancing the overall success and reliability of the model's output.

3.2.3. Modular RAG

In the preceding chapter on Advanced RAG, it has become evident that the landscape of RAG is rich with potential for enhancement through various building blocks. This realization stems from the dynamic nature of engineering solutions within the field, which continue to evolve and materialize at a rapid pace. Acknowledging the complexity and the multifaceted components of RAG systems, the research community has gravitated towards a modular approach. This approach entails the assembly, activation, and optimization of distinct modules, each contributing to the overall efficacy of the RAG framework. Such decisions surrounding module integration are informed by a multitude of factors, including inference time, budget constraints, domain specificity, information structure, target audience, and potential integrations with other technological tools.

As discussed in the Advanced RAG chapter, enhancing a RAG system can be approached through two primary avenues: the integration of new modular building blocks and the fine-tuning of existing embedding, retrieval, or generation models. The former focuses on the structural aspect of RAG systems, introducing modular components that can be tailored or replaced to improve performance. The latter, meanwhile, delves into fine-tuning of the models themselves to better align with the specific needs of the task at hand. For the purpose of this paper, our discussion will primarily center on the concept of modular RAG in the context of a "frozen" RAG scenario.

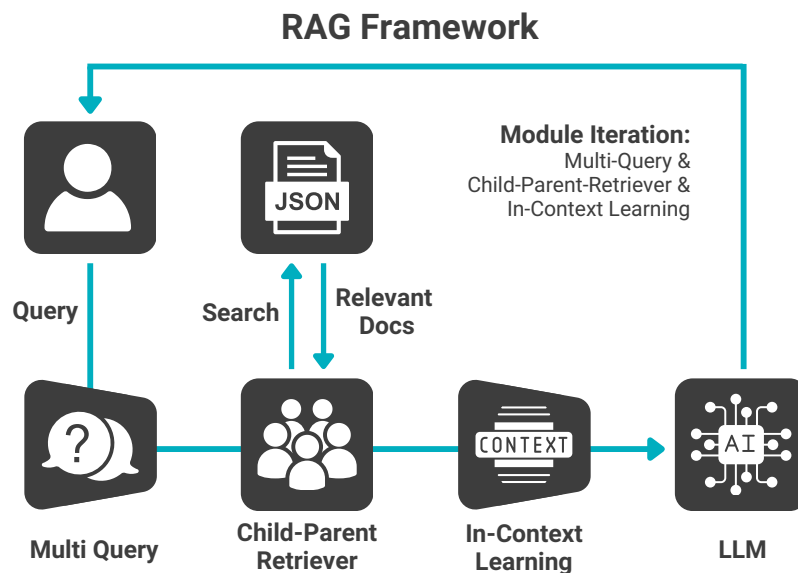


Figure 3.1.: RAG Framework of a certain Module Iteration

In the paragraphs that follow, we will explore various modules that can be integrated into the RAG framework, with a particular focus on those that align with the mentioned identified for specific factors. This examination will be presented in a sequence that mirrors the typical utilization of these modules within the RAG framework. However, it is important to emphasize that the application of these modules is not restricted to a single phase of the RAG process. For example, while we begin by discussing modules predominantly associated with the Digestion phase, it is feasible for some modules to be applied in subsequent stages of the RAG framework. The decision to employ modules at different stages requires careful consideration and evaluation to ensure that they contribute positively to the overall objectives of the RAG system. One such RAG framework is depicted in figure 3.1.

When integrating modules into the RAG framework, maintaining a coherent flow of information is paramount. It is essential to understand how different modules interact with one another. Some modules, when combined, may produce a synergistic effect, enhancing the overall performance of the RAG system beyond what would be achieved through their independent operation. Conversely, other modules may function independently, without direct interactions or dependencies on other components of the system. This diversity in module interplay necessitates a thoughtful approach to integration, ensuring that each module serves its intended purpose without disrupting the cohesive operation of the RAG framework.

3.2.3.1. Digestion Phase

- *Recursive Chunking* Recursive Chunking is an advanced method that blends straight-forward Character Chunking with the strategic use of special inputs such as enters, periods, and other punctuation marks to segment text. This technique operates under the premise that semantically similar content is typically contained within a single paragraph, aiming to preserve the integrity of this content through the chunking process. The approach is dynamic; for paragraphs exceeding a predefined character limit, the text is divided into at least two chunks to accommodate its length. Conversely, if a paragraph falls below this limit, it is selected in its entirety. The critical challenge in Recursive Chunking lies in setting an appropriate character limit that aligns with the dataset's structure and content. This limit is crucial for achieving a balance between maintaining semantic coherence within chunks and ensuring they are of a manageable size for subsequent processing stages.
- *Semantic Chunking* Semantic Chunking diverges from traditional methods by leveraging embeddings to understand and group texts based on semantic similarity, rather than imposing a uniform chunk size. This approach, utilizing strategies like hierarchical clustering with positional rewards and detecting breakpoints between sentences, aims to preserve semantic cohesion by ensuring semantically related information stays together. However, this method's iterative process of comparing chunk pairs for semantic similarity, requiring LLM inference steps per pair, makes it both time-consuming and costly. Despite its resource-intensive nature, Semantic Chunking offers a nuanced way to maintain the integrity of the text's meaning, with the potential for employing alternative

semantic similarity metrics to streamline the process.

- *Alternative Data Structure or Filters* This module recognizes that not all data benefits from being organized into chunks. It proposes using alternative data structures, such as curated JSON formats or graph structures, to facilitate easier interpretation and evaluation of the data. Additionally, incorporating filters, either within these structures or alongside metadata in chunks, can effectively segregate semantically similar data into distinct categories within a multi-dimensional database, enhancing the system's organizational efficiency.
- *Relevant-Heatmap-Indexing¹* An innovative approach to managing the distribution of company data, Relevant-Heatmap-Indexing, zeroes in on pinpointing the segments of data most frequently queried or deemed relevant. This is achieved by attaching a counting label to each vector, thereby monitoring the frequency of retrieval. Over time, this labeling technique helps identify the most commonly asked questions. With this insight, there exists the possibility to bypass the traditional, time-consuming retrieval step for these frequently asked queries. By employing cosine similarity, the system compares the query vector with the vectors of the top N most asked questions. If the similarity between the query and any of these top N vectors crosses a predefined threshold, the process can directly proceed to the generation phase, sidestepping the usual retrieval phase. This method not only streamlines the response process for common queries but also ensures efficiency and speed in delivering relevant information.

At this stage, the vector store has already been established through one of the digestion methodologies discussed previously. This vector store serves as the repository from which information will be retrieved based on the user's query. The emphasis now turns to how the system interprets and processes these queries to match them with the relevant vectors in the store. This phase is critical for ensuring that the retrieval process is both efficient and accurate. In the following, we will delve into various modules that play a key role in bridging the Digestion and Retrieval phases, each designed to optimize this transition and enhance the overall performance of the RAG system.

3.2.3.2. Transition: Digestion to Retrieval Phase

- *Hypothetical Document Embeddings (HyDE)* HyDE addresses the challenge where the semantic similarity between a question and its answer is not direct, by generating a hypothetical document that closely aligns with potential answers. This strategy enhances retrieval by focusing on the embedding similarity between the generated hypothetical answer and real documents, rather than relying on direct semantic similarity between the query and potential answers. HyDE emerges as a particularly effective solution in domains where the LLM already possesses substantial knowledge, enabling more accurate document retrieval by leveraging the model's existing understanding. [14]

¹This method is an original creation by the authors.

- *Query2Doc* Query2Doc employs LLMs to create a comprehensive pseudo-document by integrating the original query with additional contextual information. This method aims to close the semantic gap between the query and document content, facilitating a more accurate retrieval process by ensuring that the search mechanism can grasp the full spectrum of the user’s intent. [15]
- *ITER-RETGEN* ITER-RETGEN represents a sophisticated approach in query rewriting, where LLMs are used to iteratively refine the query by generating successive versions that better capture the nuances of the user’s request. This iterative enhancement helps in pinpointing the most relevant documents by gradually aligning the query’s semantics with the available content. [16]
- *Reverse Retrieval and Reading (RRR)* RRR introduces a reverse-engineered framework for query rewriting, flipping the traditional sequence of retrieval and reading. This module prioritizes rewriting the query to better match the document content upfront, thereby streamlining the retrieval process to be more effective and targeted. [12]
- *STEP-BACKPROMPTING* With STEP-BACKPROMPTING, LLMs are guided to perform abstract reasoning and retrieval by focusing on high-level concepts derived from the initial query. This method enables a broader, more conceptual approach to query rewriting, enhancing the model’s ability to address complex queries through a conceptual lens. [17]
- *Search Module* Contrary to the similarity-based retrieval found in Naive/Advanced RAG, the Search Module is specifically designed for particular scenarios, enabling direct searches across additional corpora. This module leverages code generated by LLMs, utilizes query languages such as SQL or Cypher, and employs custom tools for integration. It facilitates searches across diverse data sources, including search engines, textual data, tabular data, and knowledge graphs, broadening the scope of information retrieval beyond the vector store. [18]
- *Fusion RAG*-Fusion aims to transcend the limitations of traditional search systems by adopting a multi-query strategy. This approach enriches user queries into multiple, varied perspectives through an LLM, capturing not only the explicit information sought by users but also unveiling deeper, transformative insights. The fusion process executes parallel vector searches for both the original and the expanded queries, employs intelligent re-ranking for optimal result alignment, and combines the best outcomes with new queries. This method ensures that search results are deeply aligned with the user’s explicit and implicit intentions, facilitating more insightful and relevant information discovery. [19]
- *Task Adapter* The Task Adapter module is designed to tailor the RAG framework to a wide array of downstream tasks. UPRISE, a part of this module, automates the retrieval of prompts for zero-shot task inputs from a pre-constructed data pool, thereby

broadening the universality and applicability of the RAG system across various tasks and models. [20]

- *Query Specification* Influenced by the principles of Retrieval-augmented Language Models (RALMs), Query Specification addresses the challenge of complex user queries by decomposing them into sub-queries. This process allows for a more targeted approach during the Retrieval phase, where each sub-query is individually processed to gather relevant context chunks. In the Generation phase, these retrieved pieces of context are then synthesized to construct a comprehensive response to the original, complex query. This method not only streamlines the retrieval process by focusing on specific aspects of the query but also enhances the generation process's ability to produce accurate and complete answers by effectively piecing together the segmented insights obtained from the sub-queries. [21]

As we progress further into the intricacies of the Retrieval-Augmented Generation (RAG) framework, our focus shifts towards the modules that primarily operate within the retrieval step. This crucial phase is where the system sifts through the pre-processed and indexed data to find information that best matches the user's query. The modules designed for this step are pivotal in enhancing the precision and efficiency of information retrieval, ensuring that the subsequent generation phase is fed with the most relevant and accurate data. In the following, we'll explore some of these key modules, each playing a distinct role in refining the retrieval process:

3.2.3.3. Retrieval Phase

- *Hybrid Search Exploration or Ensemble Retriever* This module underscores the versatility of the RAG system by blending various search techniques such as keyword-based, semantic, and vector searches. Hybrid Search Exploration capitalizes on the distinct advantages of each method to cater to a wide range of query types and information requirements. This multifaceted approach not only ensures the retrieval of highly relevant and contextually rich information but also bolsters the retrieval strategy's robustness, significantly enhancing the RAG pipeline's effectiveness.
- *Recursive Retrieval and Query Engine or Child-Parent-Retriever* The Recursive Retrieval and Query Engine adopts a phased approach to information retrieval. Initially, it focuses on acquiring smaller chunks to grasp key semantic meanings, followed by retrieving larger chunks that offer more contextual details in subsequent stages. This modular retrieval strategy, also known as Child-Parent-Retriever, is designed to balance efficiency with the need to provide the LLM with contextually comprehensive responses, optimizing the retrieval process for depth and relevance.
- *Agentic Retriever* The Agentic Retriever module introduces a strategic dimension to the retrieval phase, acknowledging that some queries might be answered with general knowledge already present within the multi-dimensional embedding space. By identifying such instances, the Agentic Retriever can bypass the retrieval step altogether.

This preemptive action aids in minimizing the chances of generating inaccurate or false responses during the generation phase, especially when faced with irrelevant context, streamlining the process towards more reliable and factual outputs.

- *Low-Dimensionality Retriever*² The Low-Dimensionality Retriever begins with a dimensionality reduction step on the embedding space, revealing underlying clusters within the dataset using mechanisms like K-means clustering. During retrieval, it first assesses the similarity between the input query and identified clusters, focusing subsequent searches on the original vectors within the relevant cluster. This elegant approach leverages clustering to provide an initial high-level overview of potential relevance before delving into detailed vector analysis, streamlining the retrieval process by concentrating efforts on the most promising clusters.

Bridging the retrieval and generation phases in the RAG framework is essential for streamlining the use of retrieved data. This stage focuses on refining and preparing data to ensure the generation phase efficiently produces relevant responses. Without this crucial preprocessing, the LLM might struggle to interpret the data accurately while also generating suitable answers. Effective data flow and preparation enable the LLM to concentrate on crafting high-quality responses, enhancing the RAG system's performance by ensuring responses are both precise and contextually appropriate. The following modules introduce some selected ways to do that:

3.2.3.4. Transition: Retrieval to Generation Phase

- *Information Compression* Information Compression addresses the challenge of managing the extensive volume of data retrieved from the knowledge base. Despite ongoing efforts to expand the context length capabilities of LLMs, limitations persist, making it difficult for models to handle vast amounts of context efficiently. Information compression becomes crucial in such scenarios, aiming to distill the retrieved data to its essence. This process helps in minimizing noise, circumventing context length constraints, and ultimately, improving the generation phase's output by providing a more focused and relevant set of information for the LLM to process.
- *Diversity Ranker* This module emphasizes the importance of document diversity to mitigate redundancy in the retrieved information. By prioritizing a variety of documents, Diversity Ranker ensures that the content fed into the LLM covers a broad spectrum of perspectives, enhancing the richness and comprehensiveness of the generation output.
- *LostInTheMiddleRanker* Addressing the challenge of crucial information getting buried in the middle of retrieved content, LostInTheMiddleRanker strategically places key documents at both the beginning and the end of the context window. This method ensures that significant information is immediately accessible to the LLM, aiding in the generation of more accurate and relevant responses.

²This method is an original creation by the authors.

- *Re-Formatting* Given the variability in user input and the potential for minor changes throughout the RAG pipeline to significantly impact the LLM's output, it's crucial to ensure consistency in the structure of the information fed into the generation phase. Re-Formatting addresses this need by standardizing the output into a JSON format. This approach enables precise extraction and utilization of desired content, ensuring that the generation phase receives information in a predictable and structured form, facilitating more accurate and relevant responses.
- *Semantic Diversity* Unlike a specific module, Semantic Diversity embodies a strategic approach to handling retrieved information. It acknowledges the possibility that the top-ranked chunks from retrieval might contain very similar information, potentially overlooking diverse yet relevant insights related to the user's query. To counteract this, Semantic Diversity involves grouping the retrieved chunks to maintain a broad spectrum of information while ensuring that the content processed in the generation phase encompasses a wide range of semantic perspectives. This strategy prevents the omission of crucial, albeit less similar, information, enriching the generated responses with comprehensive and varied insights.

Within the generation phase of the RAG framework, several challenges can arise that may compromise the quality of the generated response. Despite meticulous efforts in previous phases, issues such as insufficient context provided during retrieval or the model's propensity for hallucination—where it generates content unrelated to the query—can significantly impact outcomes. Hallucinations particularly occur when the model, instead of leveraging the retrieved context, defaults to generating information based on its training, disregarding the user's specific question. Possible modules at this stage are:

3.2.3.5. Generation Phase

- *(Dynamic) In-Context-Learning (ICL)* In the generation phase of the RAG framework, the primary objective for the LLM is to focus on generating a coherent and contextually relevant response. The introduction of ICL plays a crucial role by providing the LLM with examples that align with the expected response format, guiding the model towards producing outputs that closely match the query's intent. Taking this a step further, Dynamic ICL enhances this process by dynamically selecting semantically similar Question-Answer pairs during the generation, ensuring that the guidance is not only relevant but also specifically tailored to the semantic domain of the query. This approach prevents the dilution of the LLM's focus with unrelated information, facilitating the generation of precise and contextually appropriate answers.
- *Stop Token* This module is particularly useful for managing the output of smaller, open-source models, which sometimes generate endlessly without reaching a natural conclusion. Implementing a Stop Token within the generation script provides a mechanism to define a clear endpoint for the prompt generation, ensuring that the model terminates its output at an appropriate juncture. This is crucial for maintaining the

efficiency of the generation process and ensuring that the resulting content is concise and to the point, avoiding unnecessarily prolonged or rambling text.

3.3. Evaluation

In assessing the performance of NLP models, especially within the domain of generated text, it is pivotal to employ robust evaluation metrics. Two notable metrics that have been extensively used are BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation). These metrics, while fundamental in measuring the similarity between machine-generated text and reference texts, exhibit limitations when applied to complex NLP tasks such as RAG for question-answering, due to their focus on surface-level lexical similarities. Recognizing these constraints, this chapter will further introduce and discuss alternative metrics that have emerged specifically for the nuanced evaluation of RAG systems. These advanced metrics aim to provide a more accurate assessment by considering the semantic accuracy and contextual relevance essential for evaluating the efficacy of RAG models.

The BLEU score, essential for machine translation, measures translation quality by comparing machine outputs with human references, focusing on n-gram precision and applying a brevity penalty for short translations to match reference length. However, in RAG for question-answering, BLEU's emphasis on lexical similarity over semantic accuracy limits its usefulness. It overlooks the importance of context and meaning, penalizing answers that are semantically correct but lexically different from references, highlighting the necessity for metrics in RAG systems that assess semantic and contextual understanding beyond just word-for-word matching.

The ROUGE metric, designed for evaluating text summarization by comparing machine-generated summaries to reference texts, assesses content overlap through n-grams and other sequences, focusing on how well the summary captures reference content. However, in the context of RAG for question-answering, ROUGE's emphasis on surface form overlap limits its effectiveness. RAG demands a deep understanding of context and the ability to integrate information from multiple sources, which goes beyond mere content matching. Thus, ROUGE's approach may miss the mark in recognizing semantically accurate or contextually fitting answers that don't exactly replicate reference phrasing, pointing to a need for metrics that evaluate semantic and contextual accuracy in RAG systems.

In our context, it is imperative to distinguish between the quality of retrieval and the quality of generation. This distinction is crucial because the effectiveness of a RAG system hinges not only on generating accurate and contextually relevant responses but also on its ability to retrieve the most pertinent information from a vast corpus. Assessing these two components separately ensures a comprehensive evaluation of a RAG model's overall performance, highlighting areas for improvement in both retrieving relevant documents and generating coherent, semantically rich answers.

3.3.1. RAG Evaluation

The growing importance of RAG in NLP requires a detailed evaluation method that separates Retrieval Quality and Generation Quality. This distinction is crucial because the success of RAG models depends on the accuracy of the retrieved information and the relevance and clarity of the generated text. As RAG technology evolves and finds broader applications, it's vital to improve these areas to enhance performance in different situations. [22, 23, 24]

Retrieval Quality Retrieval Quality focuses on the effectiveness of the RAG model's retriever component in sourcing contextually relevant information. This aspect of evaluation utilizes standard metrics from the domains of search engines, recommendation systems, and information retrieval, such as Hit Rate, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). These metrics offer insights into how well the retrieval mechanism can identify and prioritize pertinent information from a vast corpus, a critical step for ensuring the subsequent generation process is based on accurate and relevant context. [25, 26]

Hit Rate is a straightforward metric that measures the proportion of queries for which the correct answer is found within the top-k retrieved documents. It effectively quantifies the system's success rate in presenting the right answer among the initial set of guesses. The Hit Rate is calculated using the formula:

$$\text{Hit Rate} = \frac{\text{Number of hits in top-k}}{\text{Total queries}}$$

Mean Reciprocal Rank (MRR) offers insights into the average rank position of the first relevant document for a series of queries. It is computed as the average of the reciprocal ranks of the first relevant document across all queries, with the formula:

$$\text{MRR} = \frac{1}{Q} \sum_{i=1}^Q \frac{1}{\text{rank}_i}$$

where Q is the number of queries, and rank_i is the rank position of the first relevant document for the i -th query. MRR values range from 0 to 1, with 1 indicating perfect retrieval performance.

Normalized Discounted Cumulative Gain (NDCG) captures the effectiveness of a retrieval system by comparing the ranked list of documents it produces to an ideal ranking, taking into account the position of relevant documents. The DCG is calculated as:

$$\text{DCG}_k = \sum_{i=1}^k \frac{2^{\text{rel}_i} - 1}{\log_2(i + 1)}$$

where rel_i is the relevance score of the document at position i and k is the rank cut-off. NDCG normalizes DCG by the Ideal DCG (IDCG), which represents the perfect ranking, thus allowing for comparisons across different queries and systems:

$$\text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k}$$

These metrics collectively provide a comprehensive assessment of the retrieval phase, highlighting not just the ability to fetch relevant documents but also the efficiency of ranking them in a manner that aligns with user expectations. Understanding and optimizing these metrics can significantly enhance the overall performance of RAG models, ensuring that the generation phase is supported by the most pertinent and contextually appropriate information.

Generation Quality The evaluation of Generation Quality in RAG models is a critical process that can be conducted using larger models (e.g., GPT 4) or human evaluators. Regardless of the method, the evaluation hinges on three pivotal elements: the Original Question, the True Answer, and the Predicted Answer from the LLM. These components are essential for a thorough and accurate assessment across the four main criteria: Relevance, Coherence, Fluency, and Faithfulness. Each criterion has specific scoring criteria and steps to ensure a comprehensive assessment:

1. **Relevance** Score Criteria: Evaluates on a scale of 1-5 the extent to which the Predicted Answer encompasses the content in the True Answer, relative to the Original Question. Answers deviating from the information mentioned in the True Answer are penalized. Score Steps: Review the Original Question, Predicted Answer, and True Answer. Assess the relevance of the Predicted Answer's content, ensuring it aligns with the key points of the True Answer and the Original Question's intent. Assign a relevance score from 1 to 5, based on coverage and pertinence.
2. **Coherence** Score Criteria: Assesses on a 1-5 scale the structural and organizational quality of the Predicted Answer, ensuring it forms a cohesive narrative relevant to the Original Question and True Answer. Score Steps: Analyze the True Answer for its main topic and key points. Compare the Predicted Answer for structural alignment and logical presentation. Score coherence based on the logical flow and clarity in relation to the Original Question.
3. **Fluency** Score Criteria: Rates from 1 to 5 the linguistic quality of the Predicted Answer, focusing on grammar, spelling, punctuation, word choice, and sentence structure, in response to the Original Question. Score Steps: Evaluate the Predicted Answer for linguistic errors and readability. Assign a fluency score reflecting the ease of reading and understanding in the context of the Original Question and True Answer.
4. **Faithfulness** Score Criteria: Measures factual alignment on a scale of 1-5 between the Predicted and True Answers, penalizing inaccuracies or "hallucinated" facts not present in the True Answer related to the Original Question. Score Steps: Verify the factual accuracy of the Predicted Answer against the True Answer. Assign a score based on the presence and accuracy of factual information, ensuring it faithfully represents the True Answer in response to the Original Question.

This comprehensive evaluation framework, encompassing Relevance, Coherence, Fluency, and Faithfulness, ensures a nuanced understanding of Generation Quality in RAG models.

For a detailed description of the scoring criteria and steps associated with each evaluation metric, refer to the Appendix of this thesis. A

Using a combination of Hit Rate from the Retrieval Quality and the just mentioned four metrics from the Generation Quality, we will see in Chapter Methodology two new approaches at evaluating our RAG frameworks Generation Quality: RAG Confusion Matrix and RAG AVG-Metric Evaluation - both in using GPT 4 and Human Evaluators' in assessing our metrics.

4. Corpora

4.1. TUM Studyprogram Corpora

In developing a Question-Answer RAG framework, it is imperative to establish a connection to an external data source. This requirement stems from the intrinsic nature of RAG systems, which rely on external information to generate responses. As outlined in the "Modular RAG" chapter, various methods exist for constructing such corpora. A common approach involves assembling a vector store that houses all relevant data, enabling retrieval mechanisms.

However, this project presented unique challenges due to the homogeneity observed within the study program descriptions. Preliminary data analysis revealed that the language and content across different study program websites were remarkably similar. This similarity is understandable, given that users can usually navigate these websites easily due to consistent structural layouts, quickly locating the desired information. Consequently, employing a standard corpora would likely result in the retrieval of overlapping information from multiple study programs, diminishing the specificity and relevance of the generated responses. Faced with this dilemma, two potential solutions emerged: either develop a filtering system capable of isolating specific data segments or leverage a structured data format like JSON or a database, which inherently facilitates the selection of discrete data sections. Further investigation into the structure of the study program content, which typically follows a pattern of a heading followed by its associated content, steered the decision towards the latter option. Choosing a JSON format was driven by the desire to reduce preprocessing efforts and enable the LLM to conduct the required contextual interpretation during the answer generation phase.

The chosen approach resulted in the creation of a JSON file that connects data from both the official TUM study program website and individual faculty websites. The rationale and methodology behind the construction of this file, as well as its implications for the RAG framework's effectiveness, will be elaborated upon in subsequent sections. As a preliminary glimpse into this JSON structure, below are the first key-values on the "Mathematics in Data Science Master of Science" program. This segment illustrates the depth and organization of the data compiled, showcasing the comprehensive approach taken in this project to enhance the RAG system's performance through targeted data structuring.

```
1 {
2   "Mathematics in Data Science Master of Science (M.Sc.)": {
3     "level": "Master of Science (M.Sc.)",
4     "studiengang": "Mathematics in Data Science",
5     "description": "The master's degree program Mathematics in Data Science
   ↪ combines a high-profile education in mathematics with an emphasis on
```

```

6     ↪ the burgeoning area of Big Data.",
    "school": "The master's degree program Mathematics in Data Science
7     ↪ combines a high-profile education in mathematics with an emphasis on the
    ↪ burgeoning area of Big Data.",
8     "school_website#1": "https://www.ma.tum.de/en/studies-information/
    ↪ study-programs-mathematics/master-mathematics-in-data-science.html",
9     "Type of Study": "Full Time",
10    "Standard Duration of Studies": "4 (fulltime)",
11    "Credits": "120 ECTS",
12    "Main Locations": "Garching",
13    "Application Period": "Winter semester: 01.01. to 31.05. Summer
    ↪ semester: 01.09. to 30.11.",
14    "Admission Category": "Aptitude Assessment for Master",
15    "Start of Degree Program": "Possible for both winter and summer
    ↪ semester",
16    "Costs": "Student Fees: 85.00 EUR, Tuition fees for international
    ↪ students",
17    "Required Language Proficiency": "English",
18    "Program profile": "The Master's in 'Mathematics in Data Science' is a
    ↪ full-time degree program that usually takes two years to complete.
    ↪ Applicants must have a bachelor's degree in Mathematics or a bachelor's
    ↪ degree in Computer Science (with minor Mathematics) or an equivalent
    ↪ qualification in a similar field of study. The master's program '
    ↪ Mathematics in Data Science' is oriented towards students who want to
    ↪ receive a high profile education in mathematics with an emphasis on the
    ↪ burgeoning area of Big Data. Graduates of this program are qualified to
    ↪ understand in detail complex techniques for data editing and data
    ↪ analysis, how to adapt complex models ..."
19 }

```

Listing 4.1: JSON snippet for Mathematics in Data Science

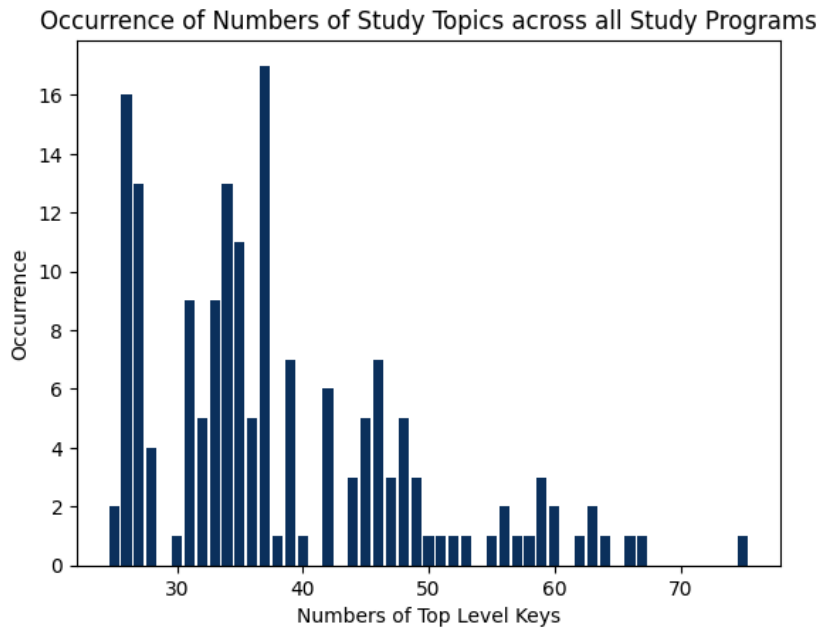


Figure 4.1.: Caption describing the figure.

The JSON snippet in listing 4.1 provides a glimpse into our Corpus, specifically focusing on the Mathematics in Data Science study program. The diversity in the number of topics covered by a study program reflects its complexity, the effort faculty members put into detailing their programs online, and the distribution of information across various headings. Figure 4.1 illustrates the variation in topic numbers across different study programs. It reveals that most programs encompass 30 to 40 topics, although there are outliers with over 70 topics, complicating the retrieval process.

4.1.1. Official TUM website

Before starting this subsection, it's paramount to acknowledge a fundamental distinction between the degree programs as presented on the official TUM website and those detailed on individual faculty pages. This segment aims to portray the nature of information typically encountered on the TUM central website, setting the groundwork for understanding its contribution to the construction of the TUM Studyprogram Corpora.

The TUM website serves as a comprehensive repository, hosting a wide array of information pertaining to its degree programs. A cursory examination of the site structure, particularly through the lens of HTML headings ranging from H1 to H5, reveals a consistent pattern of content organization. These headings include the study program and degree level alongside sections covering program overviews, "Key Data", "Study Information", "Application and Admission", and more. The content covered under these headings range from detailed program descriptions, admission requirements, curriculum specifics, study structure, language of instruction, study fees, to contact information for faculty and study coordination. This

uniformity in website architecture not only facilitates intuitive navigation for human users but also lays a solid foundation for structuring our JSON file with a clear hierarchy and segmentation of data.

This inherent structure is crucial; it enables users to swiftly pinpoint the information they seek, mirroring the objective behind the curated JSON structure aimed at optimizing our RAG framework. The deliberate organization of headings contributes significantly to the pre-processing phase, imbuing the dataset with an intrinsic level of knowledge that is vital for the subsequent generation and retrieval processes.

Venturing beyond the surface-level organization, we embarked on a more granular exploration of the content through the implementation of a Python script. This script meticulously parses through the textual data, segmenting it into chunks of approximately 1000 characters. Each chunk is then analyzed in conjunction with the associated study program and its preceding topic to dynamically refine and update the topic labels. This nuanced approach to topic categorization considered two methodologies: the extraction of key terms akin to a Named-Entity-Recognition (NER) algorithm or generating short summaries of topic contents. Opting for the former, the decision was driven by the strategic goal to later utilize cosine similarity for comparing topics, necessitating a dataset enriched with distinct, highly representative keywords that encapsulate the essence of each topic. This meticulous process of topic refinement underscores the thesis's commitment to enhancing the accuracy and relevance of the RAG system's output, ensuring that the generated responses are not only precise but also contextually aligned with the query at hand.

4.1.2. Individual Faculty website

Building upon the insights from the study program central website, this section ventures into the nuanced realm of individual faculty websites. Unlike the centralized TUM website, the websites associated with specific faculties exhibit a remarkable degree of variability. This diversity manifests in several dimensions: the depth and breadth of content, the organizational structure and number of headings and the presentation format. Some faculties have their details spread over a single webpage, while others span multiple pages; some offer content in both German and English, whereas others limit themselves to a single language. This heterogeneity poses unique challenges for data retrieval, underscoring the necessity for a tailored approach.

The task of extracting information from these diverse sources proved to be anything but straightforward. Standard web parsing tools often fell short, unable to navigate the complex and varied layouts of faculty websites. In response, a custom parser was developed, designed to retrieve div boxes that structure these web pages, extracting the textual content therein. A solution was necessary in general, since their depth of information often surpasses that of the central TUM site, offering insights that are crucial for an accurate and comprehensive response generation.

Acknowledging the significant overlap in information between the official TUM website and individual faculty pages, no further post-processing was deemed necessary upon connecting these data sources. It was observed that information common to both platforms holds

particular relevance, likely to be favored during the retrieval phase. This realization motivated the decision to integrate the data from these two origins without additional refinement, aiming to preserve the integrity and importance of the shared knowledge.

The process of data collection entailed visiting each faculty's website, employing the custom parser to harvest the information contained within. It is important to note an exception in this endeavor: the faculty of management. This faculty's website diverges from the norm not only in its exclusion of div box structuring for content organization but also in dispersing its information across multiple pages. This configuration presented a significant obstacle for the parser, complicating data extraction efforts. Given the scope and resources allocated to this thesis, dedicating additional time to manually gather data from these more complex sites was not feasible. Consequently, queries pertaining to the management faculty might not be answered with the expected depth or accuracy, reflecting the limitations inherent in this research project's methodology.

4.2. Question-Answer Set

Before diving into the intricacies of the Question-Answer Set, it's important to establish a foundation for what our model will infer. To facilitate this, we developed a script that navigates through the previously discussed JSON structure, generating Question-Answer pairs. This process involves randomly selecting a study program (root key) and then delving into its nested directory to pick a topic (top level key). The topic's content (nested value) is then fed along with the study program and topic information into GPT 4, which in turn generates a Question-Answer pair. This procedure is repeated 200 times; producing what we'll refer to as the Question and True Answer for the sake of our evaluations.

```
1 {  
2   "1": {  
3     "question": "What are the tuition fees for international students in  
↪ the Aerospace Master of Science (M.Sc.) program?",  
4     "answer": "The tuition fees for international students in the Aerospace  
↪ Master of Science (M.Sc.) program are 85.00 EUR.",  
5     "study_program": "Aerospace Master of Science (M.Sc.)",  
6     "section": "Tuition fees, International fees",  
7     "source": "Student Fees: 85.00 EUR, Tuition fees for international  
↪ students"  
8   }  
9 }
```

Listing 4.2: JSON snippet for Question-Answer Set

4.3. Evaluation Set

Adjacent to the Question-Answer Set, the Evaluation Set plays a crucial role in discerning the performance nuances of each instantiated RAG framework and their respected module permutation. Evaluating a RAG system's efficacy hinges on two principal dimensions: Retrieval Quality and Generation Quality. Ideally, the evaluation would encompass all 200 inferences for each run to ensure comprehensive coverage. However, such an extensive evaluation would be prohibitively resource-intensive. The pragmatic approach, therefore, involves selecting a representative subset of inferences from each run for assessment. This necessity arises from the specific challenges within the Generation Quality metric, where the performance evaluation must consider scenarios both when the correct source is accurately retrieved and when it is not.

For the construction of the Evaluation Set, each of the 200 Questions is funneled through our RAG framework configurations. We not only decide which modules to incorporate but also experiment with enabling and disabling different modules to explore all possible permutations. Consequently, for a single LLM, this process results in 1600 individual inferences (calculated as 1 LLM * 8 module permutations), considering each permutation of the modules.

Taking into account that we utilize 7 different LLMs—with 5 operating on both German and English datasets, and 2 only on English—the total comes to 19,200 runs (calculated as 1,600 inferences *(5 LLMs * 2 languages + 2 LLMs * 1 language)). As a result, we organize the output into 96 distinct JSON files, each representing a unique run. The naming convention for each file encodes critical information about the run:

<LLM_NAME>_<NUMBER_OF_PARAMS>_<LANGUAGE>_<MODULE_PERMUTATION> providing at a glance the LLM used, the number of parameters, the language, and the specific module permutation for that run.

This strategy acknowledges the ideal of evaluating all 200 inferences per run but recognizes the constraints imposed by practical considerations. As the number of evaluated inferences within a single run expands, we progressively approximate the true performance metric of that specific RAG configuration. Since we have about 96 runs, we decided with respect to our resources to auto evaluate only 20 inferences per run. This context leads to the creation of 96 unique evaluation sets, tailored to each run's outcomes, to accurately reflect each RAG framework's interaction with the data under varied conditions. By focusing on a subset of inferences, we aim to capture a meaningful snapshot of each configuration's performance, accommodating evaluations where the correct source has either been successfully retrieved or missed. Here is again a snippet of one selected local run:

```
1 {
2   "1": {
3     "question": "What are the tuition fees for international students in
↪ the Aerospace Master of Science (M.Sc.) program?",
4     "answer": "The tuition fee for international students in the Aerospace
↪ Master of Science (M.Sc.) program is 85.00 EUR.",
5     "true_answer": "The tuition fees for international students in the
```

```

6     ↪ Aerospace Master of Science (M.Sc.) program are 85.00 EUR.",
7     "studyprogram": "Aerospace Engineering Master of Science (M.Sc.)",
8     "true_studyprogram": "Aerospace Master of Science (M.Sc.)",
9     "section": [
10        "Student Fees, Tuition",
11        "Aerospace Engineering Master",
12        "Aerospace Engineering Careers",
13        "Aerospace Engineering, TUM"
14    ],
15    "true_section": "Tuition fees, International fees",
16    "source": "Student Fees: 85.00 EUR, Tuition Fee\nAwarded by TUM, the
17    ↪ program is conducted in Singapore and serves to provide graduates with
18    ↪ an in-depth knowledge in the field of aerospace engineering, focusing in
19    ↪ the areas of aeronautical design, space design and research.\nGraduates
20    ↪ of the Master's degree course are able to apply, analyse, evaluate and
21    ↪ develop knowledge and methods from the aerospace field whilst
22    ↪ considering the relevant technical, scientific, economic, environmental
23    ↪ and legal aspects. They are also equipped to employ a highly process
24    ↪ driven and verifiable approach to work, a critical requirement in the
25    ↪ aerospace sector. Graduates are well versed in competencies relating to
26    ↪ complete flying systems (fixed wing aircraft, rotary wing aircraft,
27    ↪ helicopters, fuselages, spacecraft and satellites) and, depending on the
28    ↪ other course modules selected, also have thorough knowledge of
29    ↪ transport systems, flight systems for inside and outside the atmosphere
30    ↪ as well as from the disciplines aerodynamics, lightweight design, flight
31    ↪ system dynamics, flight propulsion, control technology, aircraft design
32    ↪ and space travel technology. Furthermore, graduates have specific
33    ↪ knowledge of production methods and materials science (from development
34    ↪ to application) in order to meet the unique and extreme demands of the
35    ↪ aerospace field (e.g. safety, reliability, quality and structural
36    ↪ integrity). Graduates are able to understand transport systems as
37    ↪ complete systems (including their sub-systems) and also to analyse,
38    ↪ evaluate and develop them. The knowledge and skills they acquire in
39    ↪ understanding highly complex dynamic systems with all their
40    ↪ characteristics and operating conditions makes them qualified for other
41    ↪ fields in addition to aerospace. This course's graduates are in fact
42    ↪ well qualified for other sectors such as vehicle construction or
43    ↪ information technology.\n",
44    "true_source": "Student Fees: 85.00 EUR, Tuition fees for international
45    ↪ students",
46    "queries": [],
47    "language": "en",

```

```
19     "query_duration": 3.639385461807251,  
20     "child_parent_retriever": false,  
21     "multi_query": false,  
22     "bm25_retriever_weight": 0.5,  
23     "in_context_learning": false  
24 }  
25 }
```

Listing 4.3: JSON snippet for RAG run Llama 2 13B (bm25)

5. Methodology

This chapter is designed to explicate both the theoretical underpinnings and practical implementations of the RAG framework utilized in this study. It aims to detail why specific decisions were made at various stages of the RAG framework's development, particularly in light of the 96 different runs that form the core of our experimental analysis. These runs are distinguished by the activation and deactivation of modules, the employment of various LLMs, and the choice between the English and German languages. Furthermore, this chapter will explain the process of constructing the RAG framework, including the creation of a comprehensive JSON file that encompasses all study programs in both German and English, ensuring a robust dataset for the RAG system to draw upon.

In subsequent sections, the focus will shift to the LLM models used within the RAG framework, encompassing both open-sourced and closed-sourced models and the specific templates applied across different LLMs and languages. The architecture of the RAG system, detailed in this chapter, is modular, covering the three principal phases of operation: document retrieval, user query processing, and answer generation. Key modules such as the Multi-Query module (Generation Phase), Child Parent Retriever and Ensemble Retriever (Retrieval Phase), and In-context-learning (Generation Phase) will be discussed to highlight their roles in enhancing the framework's performance. Each module's inclusion reflects strategic choices made to optimize the system's efficiency and effectiveness in generating contextually relevant answers.

5.1. Models

In the landscape of LLMs, there's a notable differentiation between general-purpose text generation models and instruct fine-tuned models. General-purpose models are designed to generate a wide range of text outputs without specific instructions. In contrast, instruct-tuned models, as the name suggests, are refined on datasets that include explicit instructions or prompts followed by appropriate responses. This training method enhances the models' ability to understand and execute the instructions provided in prompts, leading to outputs that better match the intended outcome.

These instruct-tuned models signify a shift towards more task-oriented applications of LLMs, offering precision and compliance with user requests across various tasks. This distinction is crucial for our study, which incorporates a diverse array of models to cover a broad spectrum of capabilities. Each model's unique training and potential instruction-based fine-tuning contribute differently to the RAG framework. Although delving into the specific training details of each model is beyond this thesis's scope, key information about

the models will be presented, highlighting their importance and expected impact on the RAG framework's effectiveness

5.1.1. Open-Sourced Models

In our study, we have chosen to incorporate open-source models for two primary reasons. Firstly, to evaluate their performance relative to closed-source models available on the market, providing insights into how open-sourced models fare in various tasks. Secondly, utilizing open-source models presents a significant cost advantage, as expenses are incurred only during the inference step, substantially reducing the overall cost of deploying Large Language Models (LLMs) for extensive testing and development.

- We incorporate two models from the Llama family, llama2:7b-chat and llama2:13b-chat, notable for their instruction-following capabilities which are essential for processing user queries within a structured template. These templates are designed with the flexibility to include or exclude Question-Answer pairs, facilitating our in-context-learning module. The advancement of Llama 2 models is rooted in an enhanced pretraining strategy that emphasizes robust data cleaning, a refreshed data mix, and an increased training volume, among other improvements. This strategy ensures that the models are trained on a carefully selected corpus, excluding sensitive data and focusing on up-sampling reliable sources to improve accuracy and minimize inaccuracies, thus achieving a balance between cost and performance. Both models operate with a context window of 4096 tokens. [27] The template for better understanding:

```
1 <s>[INST] <<SYS>>{sp}
2 {cp}<</SYS>>
3 Human: {q1} [/INST] Response: {a1} </s>
4 <s>[INST] Human: {q2} [/INST] Response: {a2} </s>
5 <s>[INST] Human: {q3} [/INST] Response: {a3} </s>
6 <s>[INST] Human: {{user_input}} [/INST] Response:
```

Note that sp stands for System Prompt, cp for Context Prompt, and q*, a* with * $\in \{1, 2, 3\}$ are used only in the In-Context-Learning step. {user_input} will be included from the user during the Generation Phase.

- Another model is mistral:7b-instruct, which at his time of being published outshines similar open-source models in areas such as commonsense reasoning, world knowledge, reading comprehension, and code. It offers the performance equivalent to a hypothetical Llama 2 model more than three times its size, showcasing superior cost and memory efficiency. The model utilizes a sliding window attention mechanism for a linear compute cost, significantly improving processing speed for lengthy sequences, with a context window of 4096 tokens. Similar to Llama models, we used the custom template for Mistral 7B. [28]

- Alongside Llama 7B/13B and Mistral 7B, we also leverage the capabilities of `vicuna:7b-16k` and `orca-mini:7b-v3` models. Vicuna (16k) is an evolution of the Llama 2, enhanced through supervised instruction fine-tuning and linear RoPE scaling, trained on approximately 125K conversations from ShareGPT.com. Orca Mini is part of the OpenLLaMa-7B model suite, benefits from explain-tuned datasets derived from Instructions and Input from WizardLM, Alpaca, and Dolly-V2 datasets, utilizing construction approaches from the Orca Research Paper; this model is skilled at generating explanations with a default context window of 2.048 tokens.

For practical implementation, we utilized the Ollama wrapper provided by LangChain. This choice was motivated by the simplicity and efficiency of running the models, streamlining the process of integrating these open-source LLMs into our RAG framework.

5.1.2. Closed-Sourced Models

- Now, we examine the use of closed-source models `gpt-3.5-turbo-1106` and `GPT 4-0125-preview` for our RAG framework, due to their advanced generative capabilities and large context windows, 16.385 and 128.000 tokens respectively. While specifics about their training data and internal mechanisms remain undisclosed, their performance in language comprehension, creativity, and handling complex queries is noteworthy. The choice of these models is driven by the diversity in training approaches and the expected variation in response quality across different tasks. Similar to the open-source models here is a representation of the template used:

```
1 ### System:
2 {sp}
3 ### Context:
4 {cp}
5 ### Few-shot-example:
6 Q1: {q1}
7 A1: {a1}
8 Q2: {q2}
9 A2: {a2}
10 Q3: {q3}
11 A3: {a3}
12 ### Human: {{user_input}}
13 ## Response:
```

Note, that we are providing the System Prompt inside the prompt itself. This has been chosen deliberately, due to the fact, that we wanted to have similar starting points for all the models and not alter the configuration of any models in any way.

5.2. Pre-Retrieval Phase

Within the scope of this thesis, having established a TUM study program dataset and selected LLMs, we now transition to the phase that bridges digestion and retrieval, termed as the Pre-Retrieval phase. This segment is pivotal, considering that each language model is adapted to a unique template, designed to dynamically incorporate necessary variables for every inference process. This phase marks the initial interaction with the prepared JSON file, which has been meticulously cleaned and structured to facilitate efficient filtering based on specific study programs and topics. Herein, we delve into the methodologies employed to seamlessly navigate from the digestion of prepared data to the strategic retrieval of contextually relevant information.

As discussed four modules play crucial roles: Multi-Query, Child-Parent-Retriever, Ensemble Retriever, and In-Context-Learning. These modules are designed to enhance the framework's flexibility, allowing for precise adjustments based on the task at hand. In the following we will illuminate what happens within our RAG framework. As mentioned modules can be either be activated or deactivated; at every time there is the possibility to do one of the two, we are going to explain what happens when we skip the module and what happens when we enable the module. Typically for this chapter skipping the module will be the default.

The initial step in the process is aiming to accurately determine the study program and topic. This stage, while akin to data retrieval, is distinctively positioned outside the Retrieval Phase. This delineation underscores our strategic decision to emphasize that the Retrieval Phase exclusively leverages specific modules for extracting the data that will later be used as context for the Generation Phase.

Identifying the appropriate study program and topics acts as a filtering mechanism. The selection of a study program is executed by correlating the user's query with a predefined list of programs, which are read from the JSON file positioned as root keys. Combining the user query and the possible study programs in the following prompt template yields the answer. This process ensures the selection of the most relevant study program based on the query provided.

```

1 ### System:
2 The following will be a query by a student about study programs of the
   ↳ Technische Universitaet Muenchen (TUM). I want you to output the study
   ↳ program the student is having questions about. Only output the study
   ↳ program!
3 ### Context:
4 Please note, that we only need the study program. Do not self-reference,
   ↳ comment or give any notes. Only output the study program. Here are the
   ↳ possible study programs: {listed_root_level_keys}
5 ### Human:
6 How many ECTS points do I need for Mathematics in Data Science?
7 ## Response:

```


To ensure we later accurately retrieve topic values, we match the generated study program name with the exact names in our dataset. We address potential mismatches by comparing the generated name's cosine similarity with all study programs in our vectorstore. The program with the highest similarity score is selected, bypassing the initial LLM output for greater precision in identifying the correct study program.

It's important to note that our framework does not incorporate user journey safeguards or checks at this point, as the primary focus is on achieving accurate and relevant information retrieval. The decision to omit such features stems from the goal of exploring the framework's efficacy in a controlled, research-oriented environment, rather than user interaction optimization. Following the determination of the study program, the next step involves pinpointing the specific topic within that program.

Upon obtaining the study program from the initial query, the next step is to put the user's query in a structured template designed to direct the model's focus towards generating topics relevant to the study program:

```

1 ### System:
2 Classification task: You will get a question from the user, and your task is to
   ↪ classify the question in the most likely class.
3 ### Context:
4 Please note, that we only need the output. Do not self-reference, comment or
   ↪ give any notes. Only output the most likely class. Here are the possible
   ↪ classes: {top_level_keys}
5 ### Few-shot-example:
6 Q1: "I am interested in Mathematics in Data Science"
7 A1: "Program profile"
8 Q2: "How many ECTS points do I need for Informatics Masters?"
9 A2: "Credits"
10 Q3: "How high is the fee in Life Biology Bachelor?"
11 A3: "Fees for the program"
12 ### Human:
13 How many ECTS points do I need for Mathematics in Data Science?
14 ## Response:

```

In this phase, we manually apply in-context learning to direct the model towards generating relevant topics based on the selected study program. This ensures focused and accurate retrieval of information. Similar to previously we create a vectorstore of all the subsequent topics of the study program and match the top 5 values that we have found. This is later on given to the LLM.

5.2.1. Multi Query

The Multi-Query module is utilized to generate variations of the user's original question, aiming to explore different formulations that could prompt the LLM to provide a broader range of topics as context during the Generation Phase. We note that the study program

remains constant; only the user’s question is rephrased. This strategy ensures we can elicit more detailed or varied information relevant to the same study program without altering its core focus. Here’s a template for clarity:

```
1 ### System:
2 Your task is to generate two other different versions of the given user
  ↳ question to retrieve relevant documents from a vector database with
  ↳ respect to the areas of interest. By generating multiple perspectives on
  ↳ the user question, your goal is to help the user overcome some of the
  ↳ limitations of the distance-based similarity search. Provide these
  ↳ alternative questions numbered from 1. to 2. in newlines.
3 ### Context:
4 The Studyprogram is called '{studyprogram}' and the area of interest are '{
  ↳ top_level_keys}'
5 ### Human:
6 How many ECTS points do I need for Mathematics in Data Science?
7 ## Response:
```

To efficiently refine our question and topic selection, we employ a straightforward method: after generating questions and topics from the LLM, we use regex to extract each item based on specific markers (e.g., '1.' to the newline for the first item, and similarly for subsequent items). This process allows us to identify and collect relevant questions and topics directly from the LLM’s output. For topic refinement, we insert the question into our topic generator, compare the generated topic with existing ones, and select the five most similar topics to update our list—using the property of a set to avoid duplicates.

At this juncture, we have pinpointed the study program and relevant topics. Subsequently, we construct our document database by reading values from the JSON. This pre-retrieval step ensures our database is focused on selected areas, sidestepping the common issue of retrieving overlapping information from various study programs. Unlike many RAG frameworks that commence without pre-filtering, our method concentrates the database on a targeted subset, setting a foundation for text chunk retrieval from this refined data space. Thus, we’re ready to embark on the Retrieval Phase, focusing solely on this streamlined subset.

5.3. Retrieval Phase

In the retrieval phase, highlighted in the Background chapter, we explore various modules to enhance question-answering capabilities within our RAG framework. Specifically, we employ two distinct retrieval strategies: the Child-Parent Retriever, known in literature as the Parent Retriever or Small2Big Retriever, and the Ensemble Retriever, often referred to as the Hybrid Retriever. The Child-Parent Retriever focuses on initially retrieving smaller, more focused chunks of information, which are then used to guide the retrieval of broader, contextually rich content. Conversely, the Ensemble Retriever integrates at least two different retrieval methodologies, allowing for a weighted approach in selecting the most relevant text chunks.

5.3.1. Child Parent Retriever

In the methodology of the Child-Parent Retriever, we distinguish between two types of vector stores: the "memory store" for child chunks and the "vectorstore" for parent chunks, each tailored for different granularities of data. The memory store ingests data segmented into 300-character chunks, while the vectorstore processes larger 1500-character chunks. All embeddings for these vector stores are generated using the all-MiniLM-L6-v2 model, ensuring consistency across the embedding process. This choice is driven by a desire to maintain uniformity in our embedding strategy, mirroring our approach in template usage where we aim to avoid model-specific prompts influencing the retrieval process outside of designated areas. This uniform embedding approach facilitates the comparative analysis of results across different models within our RAG framework.

The challenge is to reconcile the need for small document chunks, which allow for more accurate embeddings, with the requirement for maintaining sufficient document size to ensure context is preserved. Each child chunk is assigned a unique ID, and corresponding parent chunks maintain a list of these IDs. During retrieval, the process identifies a relevant child chunk in the memory store, retrieves its unique ID, and then locates the parent chunk containing that ID in its list. This approach ensures that while the embeddings of small chunks accurately reflect their content, the broader context is not lost, as parent chunks provide a comprehensive view necessary for subsequent processing steps.

For the retrieval process we first examine each question individually. If the Multi-Query module is active, generating multiple versions of the same question, we then focus on identifying the most relevant small chunk from the memory store for each question variation, based on cosine similarity between the question and potential answers. This step pinpoints the specific chunks to be utilized as context for our model during the Generation Phase.

The Retrieval Phase also explores an alternative approach through the ensemble retriever, elaborated further in the subsequent section.

5.3.2. Ensemble Retriever

The Ensemble Retriever integrates two retrieval methods: BM25, which prioritizes documents based on direct term matches, and cosine similarity, which focuses on the contextual relevance between texts. This combination offers a versatile approach to document retrieval, balancing the precision of exact term matching with the depth of semantic understanding.

BM25 BM25 is a retrieval function utilized in information retrieval to rank documents by their relevance to a query. It operates under the bag-of-words model, disregarding the sequence and proximity of query terms within documents. The effectiveness of BM25 stems from its consideration of term frequency and inverse document frequency. Term frequency refers to the number of times a query term appears in a document, indicating relevance. Conversely, inverse document frequency assesses the rarity of a term across all documents, assigning higher value to rarer terms, thereby balancing the relevance of frequently occurring terms.

The function is calibrated using parameters that adjust its sensitivity to document length

and term frequency, ensuring a fair assessment of documents of varying lengths. BM25's methodology focuses on identifying documents that contain query terms not just frequently, but also those terms that are uncommon across the document corpus, enhancing the retrieval of pertinent documents. This approach allows BM25 to effectively prioritize documents that are likely to be more relevant to the query, based on the presence and distribution of terms, without the influence of their arrangement within the documents. [29]

Cosine Similarity: Contrasting with BM25, cosine similarity measures the cosine of the angle between two non-zero vectors in a multi-dimensional space, in this case, the vectors representing the text of the question and documents. This approach evaluates semantic similarity rather than exact term match, making it advantageous for identifying documents that are contextually related to the query even if they don't share specific query terms.

The Ensemble Retriever combines the BM25 retrieval function with cosine similarity, offering adjustable weighting between the two methods to optimize search outcomes. This flexibility allows for the tuning of the retriever's focus: with a lower weight on BM25, cosine similarity predominates, emphasizing semantic relationships within the vector space. This is particularly useful for recognizing synonyms and navigating semantically diverse domains. Conversely, increasing the BM25 weight shifts the focus towards specific, non-common terms, leveraging the retriever in scenarios where domain-specific language is prevalent, and the LLM has the capacity to generate highly relevant terms. This adjustment capability makes the Ensemble Retriever adept at handling larger text segments, where semantic significance might be obscured by extraneous information, ensuring efficient retrieval by balancing semantic similarity and term specificity.

Incorporating a weight of 0.5 for BM25 within an ensemble retriever framework can be clearly described as assigning a moderate level of importance to the BM25 retrieval method relative to other retrieval methods within the ensemble. This weighting mechanism is instrumental in harmonizing the retrieval capabilities of the ensemble, ensuring that no single retriever disproportionately influences the overall ranking of documents. When BM25 is assigned a weight of 0.5, it indicates that the rankings produced by BM25 are considered equally alongside the results from other retrievers, under the assumption that other retrievers are also assigned similar weights.

The weighted Reciprocal Rank Fusion (RRF) method utilized by the ensemble retriever operates on a straightforward principle: each document's final score is a sum of reciprocal ranks from each retriever, adjusted by the retriever's weight. Specifically, for BM25, the weight of 0.5 modifies the RRF formula such that the contribution of each rank from BM25's results to a document's overall score is halved. Consequently, the influence of BM25 on the final document ranking is balanced against that of other retrievers in the ensemble.

Activating the Ensemble Retriever typically sets the BM25 weight in this thesis to 0.5, achieving an equilibrium between BM25 retrieval and cosine similarity for document ranking. The primary objective of this thesis is to evaluate the impact of incorporating BM25 into the RAG framework, seeking any positive influence on retrieval outcomes. Demonstrating a beneficial effect of BM25 integration lays the groundwork for further enhancements to the RAG framework outside of this thesis, including fine-tuning the BM25 weighting as a next

step in optimization.

Regardless of whether the Child-Parent-Retriever or the Ensemble Retriever is employed, at this step in the thesis, we have retrieved the relevant data. This completion of the retrieval stage allows us to proceed to the next phase: Generation

5.4. Generation Phase

In the Generation Phase of our RAG framework, we directly leverage the data retrieved in the preceding steps without subjecting it to further post-retrieval processing. Integrating additional post-retrieval processing could have significantly increased the complexity of our experiments, necessitating a doubling of iterations and, consequently, a substantial increase in the number of runs required for comprehensive analysis. This section, therefore, centers predominantly on the aspect of In-Context-Learning, examining how the RAG framework utilizes the unaltered retrieved data chunks to generate responses that are contextually relevant and informed by the input query and the associated retrieved information.

In the case of the not enabling the In-Context-Learning module we use the following template:

```
1 ### System:
2 Answer the question in one to maximally two sentences based only on the
   ↪ following context. Keep it short.
3 ### Context:
4 \"{context}\"
5 ### Human:
6 How many ECTS points do I need for Mathematics in Data Science?
7 ## Response:
```

Here the context variable is filled up with the text chunks that we have retrieved during the Retrieval Phase.

5.4.1. In-Context-Learning

This phase begins after the retrieval of data chunks, a process thoroughly examined in the Background chapter, where it was decided to refrain from altering the retrieved data to maintain focus on the core components of the RAG framework. Opting not to implement additional post-retrieval processing was strategic, aiming to avoid the potential doubling of iterations and the subsequent need for increased computational runs. The crux of this chapter, therefore, is to explore the efficacy of In-Context Learning, particularly through a three-shot approach.

The literature on In-Context Learning differentiates among one-shot, two-shot, and five-shot approaches, each offering varying degrees of learning complexity and adaptability. Similar to adjustments in the BM25 weighting, our primary goal is to get whether a significant improvement in response generation can be observed with a three-shot approach, independent of the number of examples we parse in the context as learning. This would potentially justify

further exploration into varying the lengths of In-Context Learning. A considerable challenge was selecting three Question-Answer pairs that exhibit a broad semantic diversity to prevent model bias towards the domains of the examples provided. Such bias could impair the model's performance on user queries from unrelated domains. In the forthcoming sections, we will detail the template used for this In-Context Learning approach, emphasizing its design to maximize the model's ability to generate contextually rich and accurate responses across diverse query domains.

```
1 System:
2 Answer the question in one to maximally two sentences based only on the
   ↳ following context. Keep it short.
3 ### Context:
4 \"{context}\"
5 ### Few-shot-example:
6 Q1: How do I apply for the Master's program in Management at TUM if I have an
   ↳ undergraduate degree from outside the EU/EEA?
7 A1: You must apply through the TUMonline portal and provide Preliminary
   ↳ Documentation (VPD), with your documents pre-evaluated through uni-
   ↳ assist for the Management program.
8 Q2: Where are the main locations for the Teaching at Academic Secondary Schools
   ↳ â Scientific Education Master of Education (M.Ed.) program offered by
   ↳ TUM?
9 A2: The main locations for this program are Munich, Garching, and Weihenstephan
   ↳ (Freising).
10 Q3: What types of specialized modules and hands-on experiences can students
   ↳ expect in the Biology Master of Science (M.Sc.) program at TUM?
11 A3: Students can engage in a variety of modules like mammalian cell culture,
   ↳ plant physiology, ecological microbiology, and oncology, along with
   ↳ research internships offering 1:1 supervision, focusing on current
   ↳ research tools in small or very small groups.
12 ### Human:
13 How many ECTS points do I need for Mathematics in Data Science?
14 ### Response:
```

5.4.2. Parsing Meta-Data

For every inference step within our RAG framework, multiple layers of information were incorporated to facilitate a deeper analysis and understanding of the framework's operations. This enriched dataset enables a granular inside view of the RAG framework, beginning with the identification of specific study programs and topics (also referred to as sections) parsed during the filtering step into the retrieval process. Additionally, this approach allows for an examination of the context seen by the LLM during the Generation Phase, offering insights into the data the LLM was exposed to and how it influenced the response generation. To further

assess and categorize the outputs, each includes a depiction of the module state—indicating whether specific modules were enabled or disabled. This data provides a clear view of how the internal mechanisms collectively contributed to constructing the entire pipeline.

Moreover, the inclusion of the duration metric for each inference step offers an understanding of the time efficiency of different processes within the framework. Most open-source model inferences were performed using either an Nvidia Tesla V100 or Nvidia Tesla A100 GPU on Google Colab, with a preference for the Tesla A100, although resource availability sometimes necessitated the use of the Tesla V100. Conversely, all closed-source models were run through their respective APIs, utilizing a 100 MB WiFi connection, delineating a distinct operational environment for these models.

To get a better understand, here is the result of one inference step, using Llama 13B on our english dataset:

```

1 {"95": {
2   "question": "What are the tuition fees for international students from
   ↪ third countries in the Mathematics in Data Science Master of Science (M.
   ↪ Sc.) program?",
3   "answer": "The tuition fees for international students from third
   ↪ countries in the Mathematics in Data Science Master of Science (M.Sc.)
   ↪ program at TUM are 6,000 EUR per semester.",
4   "true_answer": "The tuition fees for international students from third
   ↪ countries for this degree program are 6,000 euros per semester.",
5   "studyprogram": "Mathematics in Data Science Master of Science (M.Sc.)"
   ↪ ,
6   "true_studyprogram": "Mathematics in Data Science Master of Science (M.
   ↪ Sc.)",
7   "section": [
8     "Tuition Waivers Scholarships",
9     "Tuition fees",
10    "Exams, studying abroad",
11    "Study Program Requirements",
12    "Mathematics, Data Science"
13  ],
14  "true_section": "Tuition Waivers Scholarships",
15  "source": "The tuition fees for international students from third
   ↪ countries for this degree program are 6,000 euros per semester. Many
   ↪ international students can have their fees waived or receive
   ↪ scholarships to finance them. You can find all information on waivers
   ↪ and scholarships here. Please note: The semester fee as a contribution
   ↪ to the student union must be paid additionally. It varies depending on
   ↪ where you are studying. You can find all information on the semester fee
   ↪ here.\nStudent Fees: 85.00 EUR, Tuition fees for international students
   ↪ \nTUM Center for Study and Teaching Email:studium@tum.de\nhttps://www.ma.
```

```

15 ↪ tum.de/de/studium/studiengaenge-mathematik/master-mathematics-in-data-
16 ↪ science.html\nThe tuition fees for international students from third
17 ↪ countries for this degree program are 6,000 euros per semester. Many
18 ↪ international students can have their fees waived or receive
19 ↪ scholarships to finance them. You can find all information on waivers
20 ↪ and scholarships here. Please note: The semester fee as a contribution
21 ↪ to the student union must be paid additionally. It varies depending on
22 ↪ where you are studying. You can find all information on the semester fee
23 ↪ here.\nStudent Fees: 85.00 EUR, Tuition fees for international students
24 ↪ \nTUM Center for Study and Teaching Email:studium@tum.de\nhttps://www.ma-
25 ↪ tum.de/de/studium/studiengaenge-mathematik/master-mathematics-in-data-
26 ↪ science.html\n",
27     "true_source": "The tuition fees for international students from third
28     ↪ countries for this degree program are 6,000 euros per semester. Many
29     ↪ international students can have their fees waived or receive
30     ↪ scholarships to finance them. You can find all information on waivers
31     ↪ and scholarships here. Please note: The semester fee as a contribution
32     ↪ to the student union must be paid additionally. It varies depending on
33     ↪ where you are studying. You can find all information on the semester fee
34     ↪ here.",
35     "queries": [
36         "1. What are the total costs, including tuition fees and other
37         ↪ expenses, for international students from third countries enrolled in
38         ↪ the Mathematics in Data Science Master of Science (M.Sc.) program?",
39         "2. How do the tuition fees for international students from third
40         ↪ countries in the Mathematics in Data Science Master of Science (M.Sc.)
41         ↪ program compare to those of domestic students or students from other
42         ↪ countries?"
43     ],
44     "language": "en",
45     "query_duration": 14.749154806137085,
46     "child_parent_retriever": false,
47     "multi_query": true,
48     "bm25_retriever_weight": 0.5,
49     "in_context_learning": true
50 }

```

Listing 5.1: JSON snippet for Llama 13B inference step

5.5. User Interface

In our exploration of optimizing the RAG framework, a Streamlit user interface has been integrated to facilitate interactive demonstrations and evaluations. Streamlit is an open-source app framework designed for machine learning and data science teams, enabling the rapid creation of custom web apps for data analysis, machine learning, and AI-driven projects. Its simplicity and efficiency make it an excellent tool for showcasing models and frameworks like ours, allowing users to interact with the technology directly without navigating complex programming environments.

The Streamlit app developed for our RAG framework offers flexibility in deployment, capable of running either online for broader accessibility or locally on a device for more controlled, personal use. We differentiate between two configurations for initializing the app: one for leveraging open-source models and another for utilizing closed-source models. In the case of running the app locally with closed-source models, the process is straightforward. We begin by specifying the module iteration and the language, which dictates the dataset to be used. Once initialized, the app operates similarly to a chatbot, ready for interaction.

Conversely, to run the app locally with an open-source model, an additional step is required: initiating the server (in our case this is Ollama) that hosts the LLMs. This server is directly linked to our repository, ensuring seamless integration and operation of the models within the Streamlit app. After launching the server, the subsequent steps mirror those of the closed-source configuration. During initialization, the user selects the module iteration and language, prompting the user interface to activate and become ready for interactive sessions.

5.6. Evaluation

5.6.1. RAG Confusion Matrix

The advent of RAG systems has introduced complex dynamics in NLP tasks, necessitating refined evaluation techniques. Building on this need, the RAG Confusion Matrix emerges as a novel evaluation framework that synergizes existing evaluation metrics that we saw in the Background Chapter with the unique structural properties of RAG systems. This innovative approach is designed to offer a comprehensive assessment strategy, capturing the multifaceted performance aspects of RAG models in handling real-world tasks.

This chapter will first outline the foundational concept behind combining traditional evaluation metrics with the distinct characteristics of RAG systems to formulate the RAG Confusion Matrix. This novel evaluation paradigm is structured to address the intricate challenges posed by RAG systems, especially in terms of accurately sourcing relevant information (context) and synthesizing this information into an accurate, coherent, and contextually relevant Predicted Answer. Through this comprehensive evaluation framework, we endeavor to illuminate the strengths and pinpoint areas for improvement within RAG systems, thereby contributing to the advancement of NLP technologies and their applications.

The traditional Confusion Matrix is a fundamental tool in machine learning for evaluating the performance of classification models. It provides a visual and quantitative representation

of the true and predicted classifications, divided into four categories: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). This matrix allows for a nuanced analysis of a model's accuracy by not only highlighting correct predictions but also categorizing errors, offering insights into the model's precision and recall.

In the context of RAG systems, the Confusion Matrix concept is adapted to assess both the retrieval and generation phases of these models. This adaptation recognizes the unique operational dynamics of RAG systems, which involve sourcing contextually relevant information before generating a response. Thus, the RAG Confusion Matrix evaluates:

1. **True Positive (TP):** The model's success in accurately retrieving relevant information and generating a response that aligns with the reference output, indicating the RAG system's effectiveness in both identifying the correct source and utilizing it to answer the query accurately.
2. **True Negative (TN):** Occurs when the RAG model identifies an incorrect source and produces an irrelevant or incorrect response. This reflects a correct understanding that the question and the provided source do not align, leading to a deliberate negative outcome.
3. **False Positive (FP):** Arises when the RAG system mistakenly assesses an irrelevant source as accurate but still produces a response that seems correct or is aligned with the reference output, showcasing a discrepancy between source relevance and response accuracy.
4. **False Negative (FN):** Happens when the RAG system overlooks a correct and relevant source, leading to an inaccurate or irrelevant response where a precise answer was possible, highlighting a missed opportunity for accurate retrieval and generation.

By adapting the traditional Confusion Matrix to the RAG framework, this approach provides a comprehensive method to dissect the complex process of information retrieval and response generation inherent in RAG systems. This novel RAG Confusion Matrix not only captures the accuracy of the final generated responses but also delves into the intricacies of how effectively RAG models source and utilize information to craft these responses.

To better understand how our RAG framework performs, we examine an example of evaluated inference steps. In our evaluation table, there's a "Match" column that shows if we correctly captured the intended topic for the Generation Phase, based on the Meta Data. Other columns include Relevance Score, Coherence Score, Fluency Score, and Faithfulness Score, with scores ranging from 1 (lowest) to 5 (highest), as previously discussed.

Our focus here is on the Relevance Score because it tells us how closely the answer from the LLM matches the true answer, indicating the accuracy of the generated response. When we refer in the following to the RAG Confusion Matrix, then we implicitly use the Relevance Score column; only when we also indicate one of the other three columns, then we delineate from the default choice. By concentrating on this score, we can assess the framework's ability to provide relevant answers. Next, we will look at an example to see RAG Confusion Matrix

Table 5.1.: Model Evaluation Summary

Match	Question	True Answer	Predicted Answer	Rel.	Coh.	Flu.	Faith.
1	Question 1	True Answer 1	Answer 1	5	5	5	5
1	Question 2	True Answer 2	Answer 2	3	4	4	2
1	Question 3	True Answer 3	Answer 3	5	5	5	4
<i>(Entries 4-10 omitted)</i>							
0	Question 11	True Answer 11	Answer 11	1	1	5	1
0	Question 12	True Answer 12	Answer 12	3	1	3	2
0	Question 13	True Answer 13	Answer 13	5	5	5	2
<i>(Entries 14-20 omitted)</i>							

in terms of relevance, giving us a clear picture of its performance in generating accurate and pertinent responses.

Table 5.2.: RAG Confusion Matrix, Metric: Relevance, Threshold: 5

	Correct	Incorrect
Match	7	3
No Match	4	6

This matrix demonstrates the distribution of correct and incorrect predicted answers, categorized by whether there is a match or no match with the source.

When developing the concept of a RAG Confusion Matrix, we made several deliberate decisions to shape our evaluation process. One key choice was how we generated scores for Relevance, Coherence, Fluency, and Faithfulness. We approached this using two judges: one being GPT 4 and the other, human evaluation. To ensure objectivity, GPT 4 evaluations were conducted in a new session to prevent any self-influence on its outputs. Human evaluations were carried out by four individuals, focusing on specific model runs selected for this purpose.

Another significant consideration was the volume of samples subjected to evaluation. As discussed our study encompassed 96 different RAG iterations, each tested with 200 questions. Directly applying GPT 4 evaluation across all samples for each of the four scoring categories would have required 76,800 API calls—a demand too high in terms of both time and financial resources. To manage this, we devised a script to randomly select 10 matches and 10 non-matches from each run, thereby constructing a concise Evaluation Summary. In cases where a run did not yield 10 matches, the number of non-matches was similarly limited, ensuring a balanced and manageable evaluation dataset.

The evaluation results played a crucial role in further refining our RAG framework, as they

provided concrete data on our model's performance. In the following chapter, we will explore the specific adjustments made to enhance the RAG system. Despite these optimizations, we were unable to conduct human evaluations on the improved versions directly. Therefore, we selected three specific model iterations and optimized the RAG framework even more that showed superior performance in Hit Rate compared to others, including both open-source and closed-source models. For these, we generated a total of 60 inference entries, evenly split between 30 matches and 30 non-matches. This set of inferences was subjected to human evaluation, and the findings from this assessment will be discussed in detail in the next chapter.

6. Evaluation Results

In this chapter, we'll tackle the main research question introduced at the start: how do specific modules influence the performance of our RAG framework? Our focus will be on evaluating the roles of Multi-Query, Child-Parent Retriever, Ensemble Retriever, and In-Context Learning. We'll also compare how open-source models perform against closed-source models within our framework.

To make sure we're all on the same page, we'll quickly review how our RAG system operates. We began by collecting data from TUM study programs, organized this information into JSON files, and then generated questions from this data to test the framework's ability to retrieve relevant information and provide accurate answers.

Our analysis involved 96 different runs, using a variety of LLMs, across both German and English study program data. After collecting all this data, we applied several evaluation metrics to a subset of inferences from each run, including some metrics we developed specifically for this thesis.

Following this analysis, we'll discuss how we optimized the RAG framework based on our findings and what changes resulted from these optimizations. This examination not only addresses our initial research question but also opens up avenues for future work in improving RAG systems. Through this detailed exploration, we aim to shed light on the effective components of RAG frameworks and the adjustments that can enhance their performance.

6.1. Retrieval Quality

In this section, we focus on assessing the Retrieval Quality of our model, which essentially measures how effectively the model can pull relevant textual information from our dataset. Thanks to our structured dataset, we have a clear advantage in determining whether the correct data was retrieved or if we managed to "hit" the targeted information. As previously mentioned in the Background chapter, there are several methods to evaluate the retrieval effectiveness, including examining how meaningfully the data is ordered and how significantly different data chunks contribute to answering a given question. We will now primarily focus on the Hit Rate as our main metric of interest.

Our framework is broken down into different parts, like building blocks, each handling a phase of the process. This approach lets us identify which parts can be fine-tuned to improve how well our system finds and retrieves the right information, aiming to enhance our Retrieval Quality.

Here's a look at the parts that impact our ability to accurately find information, directly or

indirectly:

1. We started with gathering a lot of disorganized data from many websites, then organized this data into one big JSON file.
2. Even though we tried to make sure the topics for each study program were well-defined to help with filtering, sometimes the information within a topic is too broad or complex to be neatly summed up in a few words that we call as topic.
3. As we have seen when it comes to turning a question into a search for specific information, we use two prompts that help guide the search towards the right study programs and topics.

Understanding these parts helps us see where adjustments can be made to get better at finding the right information, thus boosting our system’s effectiveness.

6.1.1. Hit Rate

Considering everything we talked so far we present the Hit Rate Table. This table will be analysed in the following.

Table 6.1.: Hit Rate over all possible RAG frameworks

Model	#P		er	cpr	icl	icl- er	mq- er	mq- cpr	mq- cpr-icl	mq- icl- er
Llama 2	7B	de	8.64	8.64	8.64	7.41	13.58	13.58	13.58	13.58
		en	43.21	43.21	43.21	43.21	53.09	53.09	53.09	53.09
Llama 2	13B	de	28.40	28.40	28.40	28.40	34.57	34.57	34.57	34.57
		en	50.62	50.62	50.62	50.62	55.56	55.56	55.56	55.56
GPT 3.5		de	56.79	56.79	58.02	55.56	66.67	64.20	70.37	65.43
		en	44.44	41.98	41.98	43.21	46.91	44.44	45.68	45.68
GPT 4		de	61.73	61.73	61.73	61.73	69.14	66.67	65.43	66.67
		en	65.43	66.67	66.67	66.67	75.31	72.84	72.84	72.84
Mistral	7B	de	39.51	39.51	39.51	39.51	48.15	51.85	51.85	49.38
		en	51.85	51.85	53.09	51.85	56.79	56.79	56.79	56.79
Orca Mini	7B	en	44.44	44.44	44.44	44.44	50.62	49.38	51.85	51.85
Vicuna	7B	en	1.23	1.23	1.23	1.23	3.70	3.70	3.70	6.17

er = Ensemble Retriever, cpr = Child-Parent-Retriever, icl = In-Context-Learning, mq = Multi-Query.

In our findings, we’ve adopted a top-down approach to evaluation, starting from a broad perspective and gradually focusing on the performance of specific module iterations. Two main observations emerge from this analysis. Firstly, it’s clear that any module iteration excluding the Multi-Query feature performs significantly worse than those that incorporate it,

highlighting the critical role of Multi-Query in enhancing retrieval effectiveness. Secondly, while open-source and closed-source LLMs generally show consistent performance within their categories—regardless of whether Multi-Query is used or not—the performance across closed-source models varies notably for identical runs. This outcome is particularly unexpected since the temperature setting for all runs was maintained at zero, suggesting that factors beyond temperature control are influencing the performance disparities observed in closed-source models.

- For the Llama 2 model with 7 billion parameters, comparing results from both German and English datasets shows clear patterns. The model performs the same across ER, CPR, and ICL strategies in each language—8.64% for German and 43.21% for English. This shows the model’s consistent approach regardless of the retrieval method but also highlights its better understanding of English, likely due to more training in this language. When we add the Multi-Query (mq) technique, performance jumps to 13.58% for German and 53.09% for English. This improvement shows that using different ways to ask a question helps the model find and use more relevant information, especially in English.
- For the Llama 2 model with 13 billion parameters, analyzing both the German and English datasets shows that performance increases in comparison to the 7 billion parameter model. Both languages exhibit identical performance across ER, CPR, and ICL strategies at 28.40% for German and 50.62% for English. This indicates improved capabilities due to the larger model size, especially in English where the model is already more adept. Introducing Multi-Query techniques further boosts performance to 34.57% for German and 55.56% for English. These enhancements, similar to the 7 billion parameter model, confirm the effectiveness of mq in expanding the model’s retrieval and response accuracy. The performance jump with the 13 billion parameter model, compared to the 7 billion parameter version, suggests that increased model size enhances the benefits of mq techniques, particularly in English, reflecting the model’s stronger base in this language.
- Moving to the GPT 3.5 model, the results for both German and English datasets provide insights into its performance relative to the Llama 2 model. For German, the performance stands at 56.79% with ER, CPR, and 58.02% with ICL, showing a notable increase in effectiveness compared to both configurations of the Llama 2 model. The English dataset shows a base performance of 44.44% with ER, 41.98% with CPR, and an identical 41.98% with ICL, indicating a slightly different trend where the model does not significantly outperform the Llama 2’s 7B and 13B configurations in English. The introduction of Multi-Query techniques enhances the GPT 3.5 model’s performance to 66.67% for German and up to 46.91% for English. This enhancement is particularly striking in the German dataset, suggesting that GPT 3.5 is exceptionally responsive to mq techniques in this language, outperforming the Llama 2 model’s mq-enhanced configurations significantly. In English, while the improvement is present, it aligns more closely with the increments seen in Llama 2, indicating a consistent benefit from mq

across models but with language-specific efficacy. These findings suggest that while the GPT 3.5 model shows a pronounced improvement in handling German queries, particularly with mq enhancements, its advantage in English is less pronounced when compared to the Llama 2 model's performance.

- For the GPT 4 model, the consolidated performance data across the German and English datasets demonstrates significant advancements over both the Llama 2 and GPT 3.5 models. The performance metrics indicate a uniform rate across ER, CPR, and ICL strategies, with 61.73% for German and an impressive 66.67% for English, showcasing GPT 4's superior understanding and processing capabilities in both languages. The uniformity across different strategies highlights the model's robustness and its ability to maintain high performance regardless of the retrieval or learning method applied. With the implementation of Multi-Query techniques, GPT 4's performance further escalates to 69.14% for German and reaches 75.31% for English. This substantial enhancement underscores the significant impact of mq techniques on GPT 4, particularly in English, where the model achieves the highest performance increment observed among the discussed models. The mq technique's ability to amplify GPT 4's performance beyond its base capabilities suggests a highly effective synergy between the model's architecture and advanced query reformulation strategies. Comparatively, GPT 4 demonstrates a remarkable capacity for leveraging mq enhancements to outperform previous models in both German and English, with the greatest relative gain observed in English. Comparatively, GPT 4 demonstrates a remarkable capacity for leveraging mq enhancements to outperform previous models in both German and English, with the greatest relative gain observed in English.
- The Mistral model, equipped with 7 billion parameters, presents a distinct performance profile when evaluated across the German and English datasets. In both languages, the model's performance with the base strategies of ER, CPR, and ICL is consistent—39.51% for German and slightly higher at 51.85% to 53.09% for English. This performance is indicative of Mistral's competent handling of both languages, with a notably better baseline in English, aligning with the observed trend of models generally performing better in English. Upon the application of Multi-Query techniques, Mistral's performance improves to 48.15% for German and to 56.79% for English. These improvements demonstrate the beneficial impact of mq strategies on Mistral's ability to process and respond to queries more effectively, enhancing its retrieval and generation capabilities. The increase is significant yet somewhat less pronounced when compared to the GPT 4 model's leap in performance with mq techniques, suggesting differences in how each model architecture capitalizes on the expanded retrieval scope provided by mq. Mistral's performance, while robust, suggests that it offers a balance between the capabilities seen in earlier models like Llama 2 and the more advanced GPT 4, with its responsiveness to mq techniques indicating a solid, though not unparalleled, ability to leverage advanced query reformulations.
- Orca Mini and Vicuna, both with 7 billion parameters and operating on the English

dataset, show contrasting responses to retrieval strategies and Multi-Query enhancements. Orca Mini delivers a steady performance of 44.44% across all base strategies, with mq techniques nudging its effectiveness up to between 49.38% and 51.85%. This indicates a moderate but clear benefit from the mq approach. Vicuna, however, starts from a much lower baseline of 1.23% and sees only slight improvements with mq, reaching up to 6.17%. This minimal increase suggests significant challenges in utilizing mq enhancements effectively, hinting at potential limitations in Vicuna’s design or training that mq strategies cannot overcome.

Hit Rate Summary In evaluating the performance of different LLMs with a focus on enhancing module configurations, GPT 4, when equipped with the Multi-Query enhancement in the English dataset (mq-icl-er configuration), emerges as the standout performer among closed-source models. On the open-source front, Llama 2 with 13 billion parameters and Mistral with 7 billion parameters, both utilizing the mq enhancement in the English dataset (mq-icl-er configuration), present themselves as formidable contenders. While their performance metrics are commendable, the Hit Rate for Llama 2 stands at 55.56%, and for Mistral, it is 56.79%, both significant improvements that underscore the effectiveness of the mq approach in enhancing model responsiveness and accuracy.

In the refined analysis with correct figures, the Vicuna model, despite its innovative approach of being fine-tuned on user-shared conversations via ShareGPT, remains the poorest performer. Its performance underscores the challenge that even advanced fine-tuning techniques may not suffice to overcome inherent limitations within certain model architectures or training datasets.

Furthermore, the recalculated data highlight a universal trend of all models performing better on the English dataset. This observation is consistent with the initial analysis, reinforcing the notion of a linguistic bias or a stronger training foundation in English within these models.

The introduction of mq techniques brings to light an interesting dynamic in performance enhancement. For the English dataset, the average performance increase stands at approximately 5.82%, while for the German dataset, it is slightly higher at around 6.67%. This overall average performance increase of about 6.17% when switching from strategies without mq ("icl") to those with mq ("mq-icl-er") across languages highlights the effectiveness of mq in improving model performance. Notably, the slightly higher improvement in the German dataset suggests that mq techniques might be particularly beneficial in addressing or mitigating challenges associated with language complexity or dataset limitations.

6.2. Generation Quality

In this chapter, we’re looking at how well our models generate answers, but we’re doing something a bit different. Usually, methods like Child-Parent-Retriever (cpr) and Ensemble Retriever (er) are used in the first step of finding information (retrieval phase). But here, we’ve decided to include them in the generation phase, where the model creates answers.

We made this choice because our dataset is quite specific, and we wanted to exactly check how accurately the model can pick out the right study program and topics for more detailed answers. This is a bit unusual, as these methods are normally included during the Retrieval Phase and are therefore subject to calculating the Hit Rate. Because of our dataset's unique needs, we're mixing these retrieval methods with In-Context-Learning in the part where we evaluate how good the generated answers are.

To evaluate the quality of the answers generated, we use two main tools: the RAG Confusion Matrix and the Average RAG Evaluation. The RAG Confusion Matrix helps us see how often the model sticks to the topic and gives high-quality answers. The Average RAG Evaluation looks at the model's performance more broadly, not just at the best answers but at how well it does on average. Together, these methods help us measure how well the model combines retrieving and generating information to answer questions.

6.2.1. RAG Confusion Matrix

As we delve into the evaluation of our model's performance, the example table 5.1 from the Methodology chapter becomes crucial. We have a total of 96 such tables, and for each, we apply the RAG Confusion Matrix methodology discussed earlier. An essential part of this process involves focusing on specific metrics to assess the quality of the generated answers. These metrics are:

- **Relevance:** This measures the degree to which the generated answer matches the content in the reference answer. An ideal answer is one that covers all relevant points without including extraneous information. It's rated on a scale from 1 to 5, with 1 indicating poor relevance and 5 indicating perfect relevance.
- **Coherence:** This evaluates how well the generated answer flows. It should be structured and organized, presenting information in a logical sequence rather than as a disjointed collection of facts. Coherence is also rated from 1 to 5, based on the clarity and organization of the answer.
- **Fluency:** This metric assesses the grammatical, spelling, punctuation, word choice, and sentence structure quality of the generated answer. Scores range from 1 (poor fluency) to 5 (excellent fluency), with higher scores indicating fewer errors and smoother readability.
- **Faithfulness:** This evaluates the factual accuracy of the generated answer in comparison to the reference answer. It checks for the presence of incorrect or "hallucinated" facts that aren't supported by the reference. Like the other metrics, faithfulness is scored from 1 to 5, where 1 indicates a lack of factual alignment and 5 indicates complete factual accuracy.

To keep our analysis practical and cost-effective, we included a sample of 20 answers, half with matches to the reference answers and half without. For the GPT 4 model, we asked

for scores between 1 and 5 to gauge answer quality, setting a high standard for inclusion in our analysis. Only responses scoring a perfect 5 were considered, ensuring we focused on the highest quality, most relevant answers. This decision helps us maintain a rigorous standard of evaluation, consistent with the demands of a thesis, and allows us to pinpoint the most effective aspects of the RAG framework in generating relevant answers as seen in the following table 6.2

Table 6.2.: Metric: Relevance, Threshold: 5; Top 3 TP: Correct Match & Correct Response

Match	Llama2 7B (en-mq-er)		Mistral 7B (en-mq-er)		Mistral 7B (de-icl-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	9(TP)	1(FN)	9(TP)	1(FN)	9(TP)	1(FN)
0	2(FP)	8(TN)	5(FP)	5(TN)	4(FP)	6(TN)

The table presented, with a focus on the metric of Relevance and a threshold of 5, highlights the performance of open-source models Llama2 7B and Mistral 7B in both English and German configurations under the conditions of correct context matching and high-quality responses as evaluated by GPT 4. A notable finding across these evaluations is the consistent achievement of 9 correct answers (True Positives) for each model configuration regarding relevance, underlining the effectiveness of these models in processing relevant content when given the correct context.

An important consideration in interpreting these results is the methodology behind the evaluation. Specifically, the analysis was conducted on a set of 20 question-answer pairs for each model configuration, with a mix of matches and no matches to the reference answers. This approach inherently introduces variability, as the selection of question-answer pairs was random for each model run, potentially affecting the comparability of results across different runs. The sample size of 20, while practical for managing API calls and costs, may not fully represent the models' capabilities. Thus, there's an inherent challenge in drawing broad conclusions about model performance due to the possibility that some runs might have encountered either more straightforward or more complex questions than others.

Table 6.3.: Metric: Faithfulness, Threshold: 5

Match	Llama2 7B (en-mq-er)		Mistral 7B (en-mq-er)		Mistral 7B (de-icl-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	8(TP)	2(FN)	9(TP)	1(FN)	8(TP)	2(FN)
0	2(FP)	8(TN)	5(FP)	5(TN)	4(FP)	6(TN)

While relevance provides a critical view of how well the model’s answers align with the expected content, examining only this metric doesn’t paint the full picture of answer quality. To thoroughly assess the accuracy of the top 3 models and ensure they don’t fabricate information, we turn to the faithfulness metric. Upon reviewing the faithfulness scores, we notice a distinct pattern, that two of our models performed worse in the metric faithfulness, then in relevance. Although its responses were relevant, meaning they related to the posed question and aimed towards the correct answer, they diverged from the desired True Answer, introducing unfounded elements or "hallucinations."

It’s important to remember that when we score answers for things like how relevant or faithful they are, we’re only looking at the question, the correct answer, and the predicted answer. We’re not considering what context the model was looking at when it created the predicted answer. So, if a model gives a lot of details and goes beyond the exact answer needed, it might get a lower score for faithfulness. This is because the scoring system might see the extra details as mistakes, even if the model was just trying to give more information.

Observing the metrics, it’s clear that Faithfulness carries slightly more weight than Relevance. This distinction makes sense intuitively: for an answer to be considered faithful, it must inherently be relevant to the question. However, the reverse isn’t always true; an answer can be relevant—meaning it pertains to the topic at hand—without necessarily being faithful, which requires the answer to not only be on topic but also factually accurate and free of fabricated information. Given this nuanced relationship between the two metrics, our next step will be to delve deeper into the models that exhibit the highest levels of faithfulness, exploring how this attribute enhances the overall quality of the generated responses.

Table 6.4.: Metric: Faithfulness, Threshold: 5; Top 3 TP: Correct Match & Correct Response

Match	ChatGPT4 (en-mq-er)		Llama2 7B (en-er)		Mistral 7B (en-mq-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	10(TP)	0(FN)	9(TP)	1(FN)	9(TP)	1(FN)
0	5(FP)	5(TN)	4(FP)	6(TN)	5(FP)	5(TN)

When we focus on the top three models that achieved a perfect score of 5 in Faithfulness in table 6.4, we observe a shift in the leading RAG frameworks compared to those excelling in Relevance. It’s crucial to understand that the terms "Correct" and "Incorrect" in the RAG Confusion Matrices signify different things across metrics. For Relevance, "Correct" implies that the model’s response was appropriately related to the question, showcasing its ability to generate pertinent information. However, when applied to Faithfulness, "Correct" carries a more substantial implication. It indicates that the model not only matched the expected answer but also accurately adhered to the factual content without introducing any unwarranted or fabricated details. This distinction highlights the nuanced criteria used to

evaluate model performance across different metrics, emphasizing the higher standard of accuracy and reliability set by the Faithfulness metric.

When analyzing the top 3 tables that achieved a perfect faithfulness score of 5, an interesting observation emerges regarding False Positive (FP) values. These FPs represent instances where the RAG framework might have included irrelevant information in its responses. Despite this, the models were able to produce answers that remained true to the expected answers, indicating a sophisticated understanding of the question and context. This outcome suggests two key points: first, the models could discern that the question didn't directly relate to the provided context, and second, they managed to generate faithful answers based on their training, rather than introducing hallucinated content.

This finding points to an area for potential improvement: models should prioritize context over their pre-existing knowledge or "model weights" to enhance answer accuracy. A critical realization, however, is that our evaluation approach may have inadvertently penalized correct behaviors. For example, in cases of "No Matches" where the context doesn't support an answer, the ideal response from a model would indicate the lack of relevant information rather than attempting to generate a forced answer. Our current methodology, which extends the same faithfulness scoring approach to such scenarios, fails to adequately reward responses that accurately acknowledge the absence of contextually supported answers. This oversight highlights the importance of refining our evaluation templates to better recognize and reward model responses that accurately reflect the available context, especially in "No Match" situations.

To better understand how the top three RAG frameworks from table 6.4 perform, we should also look at their scores in the other three metrics beyond faithfulness. This way, we get a full picture of each model's abilities, including how relevant, coherent, and fluently they can answer questions, alongside their accuracy. By checking all these aspects, we can see which models do the best overall at generating good answers.

Table 6.5.: Metric: Relevance, Threshold: 5

Match	ChatGPT4 (en-mq-er)		Llama2 7B (en-er)		Mistral 7B (en-mq-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	7(TP)	3(FN)	7(TP)	3(FN)	9(TP)	1(FN)
0	4(FP)	6(TN)	4(FP)	6(TN)	5(FP)	5(TN)

Table 6.6.: Metric: Coherence, Threshold: 5

Match	ChatGPT4 (en-mq-er)		Llama2 7B (en-er)		Mistral 7B (en-mq-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	10(TP)	0(FN)	8(TP)	2(FN)	9(TP)	1(FN)
0	3(FP)	7(TN)	5(FP)	5(TN)	3(FP)	7(TN)

Table 6.7.: Metric: Fluency, Threshold: 5

Match	ChatGPT4 (en-mq-er)		Llama2 7B (en-er)		Mistral 7B (en-mq-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	10(TP)	0(FN)	8(TP)	2(FN)	10(TP)	0(FN)
0	7(FP)	3(TN)	9(FP)	1(TN)	6(FP)	4(TN)

When we look at how well the models did on Relevance, Coherence, and Fluency, two main points stand out. First, while the models scored high for Faithfulness, showing they could stick closely to the true answer, some, like ChatGPT4 (en-mq-er) and Llama2 7B (en-er), lost slightly on Relevance. This was because they gave more information than what was asked, getting penalized for adding unnecessary details.

Second, all the models did well on Coherence and Fluency, showing they can create answers that sound good and make sense. However, this brings up a challenge for evaluating large language models. Just because an answer is smooth and logical doesn't mean it's accurate. This situation shows we might need new ways to judge these models, especially to catch when they sound confident but are actually giving wrong information. It's a reminder that looking good isn't the same as being right, highlighting the need for better evaluation methods in the field.

The novel RAG Confusion Matrix we've introduced is effective at showcasing the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for a dataset. It helps us understand a model's ability to stay relevant to the context and avoid relying too heavily on its pre-trained knowledge or "weights." By balancing the number of matches and non-matches, we gain clearer insights into the model's performance.

This confusion matrix is adaptable, allowing us to focus on different metrics like Relevance, Coherence, Fluency, or Faithfulness. However, setting a threshold for these metrics is a crucial part of the process. For instance, lowering the Faithfulness threshold to 4 shows us that answers deemed "good enough" by the evaluating model can still be useful for our purposes and offer potential for optimization (see table 6.8).

Table 6.8.: Metric: Faithfulness, Threshold: 4

Match	ChatGPT4 (en-mq-er)		ChatGPT4 (de-mq-er)		Llama2 7B (en-er)	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	10(TP)	0(FN)	10(TP)	0(FN)	9(TP)	1(FN)
0	5(FP)	5(TN)	6(FP)	4(TN)	7(FP)	3(TN)

In Table 6.8, we observe that GPT 4 consistently scored at least a 4 across all matches, showcasing its robust performance. Notably, this is the first instance where the German dataset ranks within the top 3 positions, suggesting that larger models like GPT 4 are better at adhering to facts and minimizing hallucinations compared to smaller models. This aligns with the intuitive expectation that bigger models would perform more accurately due to their extensive training data and advanced capabilities, not to mention the cut-off date of GPT 4 in this iteration being April 2023.

Additionally, all three models managed to provide somewhat accurate answers even when presented with incorrect context. This raises an important consideration about whether such a capability is desirable. In our current framework, answering correctly in a wrong context might indicate that the model's pre-trained knowledge (weights) is prioritized over the given context. This observation prompts us to question the balance between relying on the model's internal knowledge versus the specific context provided, especially in scenarios where the context might lead to inaccurate or irrelevant responses.

Yet, deciding on a threshold in general also presents challenges. It creates a binary classification of entries—those meeting or exceeding the threshold and those that do not. This approach limits our understanding of the model's performance on entries that fall short of the threshold. For example, it matters whether false negatives scored a 1 or just missed the mark with a 4, as it significantly impacts our evaluation of the model's performance, as we just saw.

Despite these limitations, the RAG Confusion Matrix is a valuable tool for situations where only the highest quality answers are acceptable; the RAG Confusion Matrix also gives us by using appropriate templates a view in the ability of the LLM to be self-critical and express the fact that the given information is not provided in the context; mitigating hallucination. In the next chapter, we'll explore a different evaluative approach aimed at gaining a more nuanced understanding of how well the model performs across all its responses, not just those that meet a certain threshold.

6.2.2. RAG AVG-Metric Evaluation

In the chapter on RAG AVG-Metric Evaluation, we shift our focus from classifying predictions as "Correct" or "Incorrect" based on a threshold within the RAG Confusion Matrix. Instead, we'll analyze average evaluations, offering a broader view of model performance. This

approach is encapsulated in Tables 6.9 and 6.10, designed for ease of analysis and maximum clarity on the models' overall effectiveness.

Our analysis will concentrate on the top three tables related to the metrics of Relevance and Faithfulness, as detailed in Tables 6.2 and 6.4. The goal is to understand the overall performance of these models without the binary constraints of thresholds. This method provides another view of how well models perform across all given data.

Notably, "No Matches" scenarios will be de-emphasized in this chapter. Our investigation into the raw data revealed that responses to "No Matches" varied significantly among RAG frameworks, with some recognizing the absence of relevant context and others providing answers closely aligned with the True Answer. Since these outcomes can greatly vary even within the same model run, focusing on average scores for "No Matches" does not give us enough interpretable confidence and therefore we are mainly focusing on the average values of "Match".

Table 6.9.: Metric: Faithfulness, Average Values

		In-Context-Learning						
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	2.4	3.5	2.9	3.3	3.2	<i>x</i>	<i>x</i>
	No Match	2.1	2.4	4.0	3.8	3.4	<i>x</i>	<i>x</i>
en	Match	3.2	2.4	2.5	3.3	3.5	1.9	0.5
	No Match	2.3	1.4	3.5	2.7	2.2	3.0	0.5

		Child-Parent-Retriever						
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	2.6	3.1	3.2	2.6	2.9	<i>x</i>	<i>x</i>
	No Match	1.6	3.1	3.2	4.0	2.7	<i>x</i>	<i>x</i>
en	Match	3.2	3.2	2.8	3.0	2.4	2.3	0.5
	No Match	2.5	2.6	1.9	2.6	2.5	2.2	0.1

		Ensemble Retriever						
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	2.7	4.4	4.2	4.4	3.7	<i>x</i>	<i>x</i>
	No Match	1.5	3.3	2.9	3.2	2.9	<i>x</i>	<i>x</i>
en	Match	4.7	3.9	4.3	3.4	3.7	3.9	0.5
	No Match	3.5	2.6	2.1	2.3	2.5	2.7	0.2

		In-Context-Learning & Ensemble Retriever						
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	2.6	4.2	3.8	3.9	4.5	<i>x</i>	<i>x</i>
	No Match	2.0	3.6	3.1	3.0	3.9	<i>x</i>	<i>x</i>
en	Match	2.1	3.4	3.1	4.3	4.4	2.1	0.5
	No Match	2.8	3.0	2.7	2.7	3.3	2.8	0.1

Table 6.10.: Continuation of table 6.9; Metric: Faithfulness, Average Values

Mutli-Query & Ensemble Retriever								
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	4.0	3.0	3.9	4.8	4.4	<i>x</i>	<i>x</i>
	No Match	1.9	2.0	2.7	3.4	3.3	<i>x</i>	<i>x</i>
en	Match	4.5	3.8	3.6	5.0	4.6	4.0	1.4
	No Match	2.6	2.7	1.8	3.3	3.5	2.9	0.3

Multi-Query & Child-Parent-Retriever								
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	3.6	2.8	3.2	2.5	4.0	<i>x</i>	<i>x</i>
	No Match	2.0	3.3	2.6	3.1	3.8	<i>x</i>	<i>x</i>
en	Match	3.2	2.7	3.4	3.6	2.3	3.4	0.7
	No Match	2.4	2.1	1.5	2.2	2.7	3.2	0.5

Multi-Query & Child-Parent-Retriever & In-Context-Learning								
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	2.9	2.7	3.6	3.3	2.6	<i>x</i>	<i>x</i>
	No Match	3.1	2.8	2.9	3.1	4.5	<i>x</i>	<i>x</i>
en	Match	3.6	3.2	3.1	3.5	3.2	2.6	0.3
	No Match	3.2	2.3	2.5	2.7	2.2	3.2	1.0

Multi-Query & In-Context-Learning & Ensemble Retriever								
		Llama 2 7B	Llama 2 13B	GPT 3.5	GPT 4	Mistral 7B	Orca Mini 7B	Vicuna 7B
de	Match	3.6	3.3	3.9	4.0	4.1	<i>x</i>	<i>x</i>
	No Match	3.1	2.9	2.1	3.2	4.3	<i>x</i>	<i>x</i>
en	Match	3.1	4.6	4.0	3.8	4.2	3.2	2.1
	No Match	2.5	2.7	1.9	2.9	2.3	2.9	0.5

In our evaluation, we've highlighted all average scores above 4 in bold for clearer visibility. Before delving into the detailed tables from the RAG Confusion Matrix, let's outline the key findings:

- Runs using only In-Context-Learning or the Child-Parent-Retriever didn't reach an average score of 4 in any scenario.
- Conversely, configurations incorporating the Ensemble Retriever demonstrated superior

performance, with several runs achieving average scores above 4, across both English and German datasets.

- Notably, one particular run achieved a perfect average score of 5.0 in the English dataset and an impressive 4.8 in the German dataset, utilizing a combination of Multi-Query and Ensemble Retriever. This standout performance aligns precisely with our analytical goals.
- The Child-Parent-Retriever, whether used alone or in combination, did not perform as well, contradicting its potential viability suggested by some research. This discrepancy might stem from the method's reliance on semantic similarity between questions and "Child-Chunks," suggesting that a hypothetical answer comparison might have been more effective.
- The most consistently high-performing combinations involve the Multi-Query and Ensemble Retriever, with or without the addition of In-Context-Learning, indicating these are the most effective strategies in our analysis.

Llama2 7B (en-mq-er), Mistral 7B (en-mq-er), Mistral 7B (de-icl-er)

In Table 6.2, we explore the top three runs based on the highest occurrences of scores of 5 in the metric of Relevance. Remarkably, the Llama2 7B configuration (en-mq-er) showcased strong performance with an average generation rate of 4.5 when a match was provided, an impressive feat for a model with only 7 billion parameters. The other configurations, Mistral 7B (en-mq-er) and Mistral 7B (de-icl-er), also demonstrated commendable performance with average generation rates of 4.6 and 4.4, respectively. These results indicate that our top three RAG configurations, in terms of Relevance, consistently scored between 4 and 5 on average, signifying a high level of performance.

ChatGPT4 (en-mq-er) Llama2 7B (en-er) Mistral 7B (en-mq-er)

Table 5 shifts our focus to the top three runs based on the highest occurrence of perfect scores in the metric of Faithfulness. Analyzing the average generation rates in this context offers deeper insights into the effectiveness of the inference processes. Notably, the ChatGPT4 configuration (en-mq-er) stands out with 10 True Positives (TPs) and achieves the highest average generation rate of 5.0, marking it as the best performer in our study, especially when considering its impressive Hit Rate of 75.31%. This suggests that, at least within our limited sample, this configuration most accurately adheres to the given facts.

The other two configurations, Llama2 7B (en-er) and Mistral 7B (en-mq-er), posted average scores of 4.7 and 4.6, respectively. These scores offer insights into how well the True Negatives (TNs) were handled in these runs.

6.3. RAG Framework Enhancement

In this chapter, we transition from exploring different RAG Evaluation methods to focusing on enhancing the performance of the top three RAG frameworks, specifically those with the highest Hit Rates. We will revisit their performance numbers with the goal of improving the

Hit Rate through innovative approaches to retrieving study program and topic information. This effort involves a comparative analysis of their respective Hit Rates and Generation Quality across various metrics. Additionally, we'll incorporate widely recognized metrics such as Rouge and Human Evaluation to deepen our understanding of the frameworks' effectiveness. By combining these evaluation tools, we aim to gain a comprehensive view of how well each RAG framework performs in an optimized setting.

Table 6.11.: Hit Rate over all possible RAG frameworks

Model	#P		mq-icl-er	mq-icl-er-optimized
Llama 2	13B	en	55.56	74.07
GPT 4		en	72.84	81.48
Mistral	7B	en	56.79	67.90

Hit Rate Focusing intensively on three specific runs allowed us to target optimizations for achieving the best possible outcomes for those scenarios. Our optimization efforts spanned various aspects of the filtering steps, where we discovered that the template used previously might have been more of a hindrance than a help. Originally, the template presented the LLM with numerous options for different study programs and topics, potentially overwhelming it with choices. Instead, we leveraged the LLM's inherent "smarts" to deduce the appropriate study program directly from the question posed, simplifying the process.

Theoretically, incorporating additional modules into the filtering step could further enhance the Hit Rate. For the time being, our focus remains on analyzing these three runs across different metrics. This targeted approach not only simplifies the optimization process but also allows for a more manageable evaluation of changes and their impacts on performance.

Model Name	Match	Rouge-1	Rouge-2	Rouge-L	BERT-score
Llama 13B	1	0.321	0.152	0.281	0.881
Llama 13B	0	0.285	0.102	0.215	0.858
Llama 13B (optimized)	1	0.406	0.213	0.332	0.883
Llama 13B (optimized)	0	0.346	0.203	0.308	0.873
GPT 4	1	0.611	0.451	0.568	0.930
GPT 4	0	0.418	0.251	0.364	0.880
GPT 4 (optimized)	1	0.548	0.407	0.511	0.910
GPT 4 (optimized)	0	0.521	0.349	0.450	0.904
Mistral 7B	1	0.402	0.230	0.335	0.887
Mistral 7B	0	0.334	0.197	0.256	0.865
Mistral 7B (optimized)	1	0.367	0.215	0.303	0.893
Mistral 7B (optimized)	0	0.283	0.121	0.222	0.863

Table 6.12.: Average Rouge-1, Rouge-2, Rouge-L, and BERT-score by Model and Match

Rouge and BERT-Score In the context of data where Match = 1, the GPT 4 model leads in performance with the highest Rouge scores (Rouge-1: 0.611, Rouge-2: 0.450, Rouge-L: 0.568) and the highest BERT-score (0.930). This model’s scores reflect its strong ability to generate responses that closely align with the reference texts, both in detail and semantic relevance, indicating its proficiency in producing contextually accurate and engaging content.

The Mistral 7B model, while not leading, shows a balanced performance with Rouge-1 at 0.402, Rouge-2 at 0.230, Rouge-L at 0.335, and a BERT-score of 0.887. These scores suggest that Mistral 7B is capable of generating relevant and coherent responses, albeit with a slight compromise in the precision and depth captured by the GPT 4 model.

The Llama 13B model, with the lowest performance among the discussed models for Match = 1 (Rouge-1: 0.321, Rouge-2: 0.152, Rouge-L: 0.281, and BERT-score: 0.881), indicates challenges in matching the detail and semantic richness of higher-scoring models. This suggests a need for improvement in how it mirrors the nuances and semantic content of the reference texts.

Considering these observations, the GPT 4’s superior metrics suggest a strong alignment with human-like text generation capabilities, making it highly effective for tasks requiring detailed and semantically rich responses. The Mistral 7B’s scores, while not at the top, indicate its adequacy in generating coherent responses, positioning it as a competent, though not outstanding, model within the retrieval-augmented generation framework. This performance gradient highlights the importance of both detail-oriented precision and semantic understanding in generating high-quality, relevant text responses.

6.3.1. RAG Confusion Matrix

Table 6.13.: Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er

	Original Llama2 13B		Optimized Llama2 13B		Human Evaluation Llama2 13B	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	3(TP)	7(FN)	4(TP)	6(FN)	1(TP)	9(FN)
0	1(FP)	9(TN)	2(FP)	8(TN)	0(FP)	10(TN)

Analyzing the RAG Confusion Matrix reveals significant advancements not only in the Hit Rate of the best-performing RAG frameworks but also in the overall Generation Quality across all runs. Delving into specifics, the Llama 2 7B model from table 6.13, which initially showed modest performance, saw an improvement with an increase of 3 additional occurrences, totaling 4 perfect occurrences in the GPT 4 judged Faithfulness Score. However, Human Evaluation of this particular RAG framework indicates a disparity between the high Faithfulness judged by GPT 4 and the evaluators’ perception of performance, suggesting the model’s answers, despite being contextually faithful, did not meet human expectations.

Upon consulting with the four Human Evaluators, it was clarified that their assessment of Faithfulness primarily focused on the accuracy of information in relation to the True Answer provided. While the evaluators emphasized a direct comparison between the True Answer and the Predicted Answer, there’s a possibility that the RAG framework, during its Generation Phase, accessed different contextual information unseen by the evaluators. This discrepancy could explain why the outputs did not align with human evaluations, underscoring the complexity of evaluating model-generated content and the importance of considering various perspectives in assessing model performance.

This finding gains further support when examining the Human Relevance Score, which, with a threshold of 5, resulted in only 1 True Positive (TP). However, adjusting the threshold to 3—interpreted by humans as producing somewhat sufficient answers—yields 5 TPs for Faithfulness and 8 TPs for Relevance. This variance underscores a divergence in expectations between the GPT 4 evaluator and Human Evaluators regarding how questions should be answered, highlighting the nuanced challenges in aligning machine and human judgments of answer quality.

Table 6.14.: Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er

	Original Mistral 7B		Optimized Mistral 7B		Human Evaluation Mistral 7B	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	9(TP)	1(FN)	7(TP)	3(FN)	2(TP)	8(FN)
0	1(FP)	9(TN)	5(FP)	5(TN)	0(FP)	10(TN)

Examining the Mistral 7B model from table 6.14 reveals intriguing outcomes. The evaluation by GPT 4 worsened by 2 points, yet the model managed to retrieve 13.11% more study programs from the overall questions posed. This indicates a potential trade-off where incorporating additional data into the data pool, and thereby expanding the context available to the RAG framework, might introduce more noise, negatively impacting Generation Quality.

However, Human Evaluation presents a contrasting viewpoint, hinting at a mismatch in expectations. Lowering the threshold to 3 unveils a different scenario; with the Human Faithfulness Metric, we observe 8 True Positives (TP) and 8 False Positives (FP), suggesting that Mistral has a considerable capacity to meet moderate human expectations, even when the context is inaccurately provided. Similarly, adjusting the Human Evaluators' Relevance Score to a threshold of 3 results in the RAG framework achieving 6 TPs. This disparity underlines the difference in evaluation standards between GPT 4 and human evaluators and suggests that Mistral can satisfactorily navigate medium-level expectations in terms of faithfulness and relevance, despite potential contextual inaccuracies.

Table 6.15.: Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er

	Original GPT 4		Optimized GPT 4		Human Evaluation GPT 4	
	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
1	6(TP)	4(FN)	8(TP)	2(FN)	7(TP)	3(FN)
0	3(FP)	7(TN)	5(FP)	5(TN)	4(FP)	6(TN)

Examining the performance of the final model, GPT 4, across its original RAG framework, an optimized version, and through Human Evaluation concerning the Metric Faithfulness with a threshold of 5, we observe consistent performance across all generations. Additionally, GPT 4 manages to increase its Hit Rate to 81.48%, marking an improvement of 8.64%. This significant improvement positions GPT 4 at the forefront within the English dataset. Adjusting the threshold down to 3 for Human Evaluation, we see an enhancement in results: the Human

Faithfulness Score reaches 9 True Positives (TP), and for the Relevance Score, we achieve 8 TPs, nearing perfect generation quality.

For completeness, it's worth noting that both Fluency and Coherence metrics, when evaluated by Human Evaluators at a threshold of 3 for all three RAG frameworks mentioned, almost consistently scored full points. This uniform high performance led to their exclusion from this chapter's detailed discussion, as there were no distinct insights to be drawn from these metrics beyond their universally high scores.

6.3.2. RAG AVG-Metric Evaluation

Table 6.16.: Average Values; Module Iteration: mq-icl-er

		Relevance					
		Original Llama 2 13B	Optimized Llama 2 13B	Original GPT 4	Optimized GPT 4	Original Mistral 7B	Optimized Mistral 7B
en	Match	3.5	3.3/3.0	3.4	4.4/4.1	4.6	3.9/2.9
	No Match	2.7	2.7/2.3	2.6	3.3/3.7	2.5	3.4/2.1
		Fluency					
		Original Llama 2 13B	Optimized Llama 2 13B	Original GPT 4	Optimized GPT 4	Original Mistral 7B	Optimized Mistral 7B
en	Match	4.2	4.5/4.8	4.5	4.9/4.8	4.9	4.9/4.9
	No Match	4.4	4.4/4.6	4.1	4.8/5.0	4.4	4.7/4.7
		Coherence					
		Original Llama 2 13B	Optimized Llama 2 13B	Original GPT 4	Optimized GPT 4	Original Mistral 7B	Optimized Mistral 7B
en	Match	3.8	4.5/3.9	3.4	4.4/3.8	4.5	4.6/4.0
	No Match	2.5	2.6/3.4	2.4	3.2/ 4.4	2.9	3.2/3.0
		Faithfulness					
		Original Llama 2 13B	Optimized Llama 2 13B	Original GPT 4	Optimized GPT 4	Original Mistral 7B	Optimized Mistral 7B
en	Match	3.2	3.2/2.7	3.4	4.4/4.3	4.5	4.2/3.4
	No Match	2.0	2.1/2.2	2.9	3.5/3.3	2.5	3.3/2.6

Examining table 6.16 offers us another perspective on our dataset. Before diving into the details, let's clarify how to interpret this table. The scores range from 1 to 5 for both the

original RAG framework and the optimized version. In the optimized RAG framework, scores are presented with a backslash separating two numbers: the first represents the score given by GPT 4, and the second denotes the average score from Human Evaluators.

From the matrix, it's immediately noticeable that all model runs score at least a 4.0 in fluency, as highlighted by the bold numbers. For coherence, scores hover around the 4.0 mark across the board, with the exception of the original GPT 4 run. This outlier could likely be attributed to the limited sample size used in our analysis. With this overview, we return our focus to the crucial metrics of Relevance and Faithfulness, aiming to understand how both the original and optimized RAG frameworks perform in these key areas.

The numbers provided give a clear indication of performance across different RAG frameworks. Llama 7B, despite achieving one of the highest Hit Rates, averaged around 3.2 in both Relevance and Faithfulness during the generation task, showing room for improvement. Mistral, in contrast, performed worse in the optimized version compared to its original setup according to GPT 4's evaluation, experiencing a slight increase in Hit Rate but a decrease of about 0.5 points in both metrics. This suggests a potential trade-off between achieving more information hits and a decline in generation quality, warranting further investigation that goes beyond the scope of this thesis. Due to the limited data sample, definitive conclusions are challenging to draw.

GPT 4, however, maintained similar performance levels in both metrics and was close to achieving the perfect score, with only a few outliers. This consistency is noteworthy, especially when considering the divergence in evaluations between human evaluators and GPT 4 for the optimized RAG frameworks. Llama 2 and Mistral didn't align with human preferences as well as GPT 4 did. This could indicate that GPT 4, having processed vast amounts of data, can present information in a manner more appealing to humans. Alternatively, it suggests that among all models evaluated, GPT 4 was best at discerning relevant from irrelevant information, enabling it to generate answers that closely matched the True Answer and were favorably viewed by human evaluators. This final interpretation also goes along with the findings that the performance of Llama 2 and Mistral dropped compared to the original RAG framework.

7. Conclusion and Future Work

In this concluding chapter, we summarize our findings and address the research questions posed at the outset of this thesis. Our investigation was guided by four key questions, each examining different aspects of the RAG framework, including the effectiveness of the Multi-Query in the retrieval phase and the roles of the Child-Parent-Retriever and Ensemble Retriever in the generation phase. Additionally, we explored the overarching question of how the use of different LLMs impacts the overall performance of the RAG framework.

Firstly, evidence from Table 6.11 demonstrates a significant improvement in Hit Rate with the incorporation of Multi-Query techniques, with some instances showing a doubling of rates. This underscores the Multi-Query’s effectiveness in enhancing the retrieval process.

Secondly, the Child-Parent-Retriever did not produce the anticipated benefits. A plausible explanation is the insufficient semantic similarity between the question vectorization and the short Child-Chunks, which failed to be effectively selected during retrieval. Conversely, the Ensemble Retriever showed remarkable effectiveness, particularly when paired with the Multi-Query, achieving perfect scores in table 6.10 using the GPT 4 model.

Regarding the use of different LLMs, the results warrant a nuanced discussion. GPT 4, particularly when utilizing a combination of Multi-Query and Ensemble Retriever, along with In-Context-Learning, achieved the highest Retrieval (Hit Rate: 81.48%) and Generation Quality (AVG: 5.0 at Match). However, there are notable competitors. An optimized blend involving Llama 2 13B for the Retrieval Phase, leveraging Multi-Query, In-Context-Learning, and Ensemble Retriever, yielded an impressive 74.07% Hit Rate. For the Generation Phase, the Original Mistral 7B with the same module iteration is recommended.

For future work in the realm of Retrieval Augmented Generation (RAG) frameworks, there are several avenues ripe for exploration. Initially, our investigation highlighted the impact of model size on generation capabilities, revealing that larger models tend to exhibit enhanced performance. This observation suggests the potential value of incorporating an even wider array of models, particularly those with more substantial parameter sizes, to further delineate the relationship between model size and generation effectiveness.

Another area for advancement involves the customization of prompt templates based on whether there is a match between the query and the context provided. Our metrics—Accuracy, Fluency, Coherence, and Faithfulness—focus on evaluating the connection between the Question, True Answer, and Predicted Answer. By creating separate prompt templates for scenarios with and without matches, we could gain deeper insights into each model’s generation quality and its ability to identify when relevant information is not present in the provided context. This distinction would allow for a more nuanced examination of how models handle hallucination and inaccuracies.

Moreover, the study revealed that GPT 4, despite its superior performance in terms of

Faithfulness when provided with accurate context, still shows room for improvement, as evidenced by its Hit Rate of 81.48%. This finding indicates that optimizing the query augmentation process—perhaps by refining how student queries are integrated with specific study program information—could further enhance retrieval accuracy and model performance. Our initial efforts to separate study program information from user queries represent a promising step in this direction, suggesting that more sophisticated augmentation techniques could yield even better results.

Throughout this thesis, we've not only addressed key research questions but also laid a foundation for future research in RAG frameworks. The insights gained and the methodologies developed provide a solid groundwork for further exploration and innovation in this rapidly evolving field.

A. Appendix: Evaluation Metrics for Generation Quality

Relevance

Score Criteria: Relevance is rated on a scale from 1 to 5, assessing the degree to which the generated answer encompasses the content present in the reference answer. Only information pertinent to the original question should be included, with extraneous details leading to penalties.

Score Steps:

1. Read the question, generated answer, and the reference answer carefully.
2. Compare the information in the generated answer to the reference answers and check if all points are relevant to the question.
3. Assess how well the answer covers the main query of the question, and the presence of irrelevant or redundant information.
4. Assign a relevance score from 1 to 5.

Coherence

Score Criteria: Coherence is evaluated on a scale from 1 to 5, focusing on the overall structural and organizational quality of the answer. The generated answer should be coherent, well-structured, and well-organized.

Score Steps:

1. Read the reference answer carefully and identify the main topic and key points.
2. Read the generated answer and compare it to the reference answer, checking for logical order and coverage of key points.
3. Assign a score for coherence on a scale of 1 to 5.

Fluency

Score Criteria: Fluency is assessed from 1 to 5, based on the linguistic quality of the generated answer, including grammar, spelling, punctuation, word choice, and sentence structure.

Score Steps:

1. Read the generated answer and evaluate its fluency based on grammar, spelling, punctuation, word choice, and sentence structure.
2. Assign a fluency score from 1 to 5.

Faithfulness

Score Criteria: Faithfulness evaluates the factual alignment between the generated and reference answers on a scale from 1 to 5. The generated answer should not contain hallucinated facts and must be factually consistent with the reference answer.

Score Steps:

1. Compare the generated answer to the reference answer to ensure all mentioned facts are correct.
2. Evaluate the factual accuracy and assign a score from 1 to 5, where 1 indicates all false information, and 5 indicates complete correctness.

List of Tables

5.1. Model Evaluation Summary	45
5.2. RAG Confusion Matrix, Metric: Relevance, Threshold: 5	45
6.1. Hit Rate over all possible RAG frameworks	48
6.2. Metric: Relevance, Threshold: 5; Top 3 TP: Correct Match & Correct Response	53
6.3. Metric: Faithfulness, Threshold: 5	53
6.4. Metric: Faithfulness, Threshold: 5; Top 3 TP: Correct Match & Correct Response	54
6.5. Metric: Relevance, Threshold: 5	55
6.6. Metric: Coherence, Threshold: 5	56
6.7. Metric: Fluency, Threshold: 5	56
6.8. Metric: Faithfulness, Threshold: 4	57
6.9. Metric: Faithfulness, Average Values	59
6.10. Continuation of table 6.9; Metric: Faithfulness, Average Values	60
6.11. Hit Rate over all possible RAG frameworks	62
6.12. Average Rouge-1, Rouge-2, Rouge-L, and BERT-score by Model and Match . .	63
6.13. Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er	64
6.14. Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er	65
6.15. Metric: Faithfulness, Threshold: 5, Module Iteration: mq-icl-er	65
6.16. Average Values; Module Iteration: mq-icl-er	66

Bibliography

- [1] M. Zaharia, O. Khattab, L. Chen, J. Q. Davis, H. Miller, C. Potts, J. Zou, M. Carbin, J. Frankle, N. Rao, and A. Ghodsi. *The Shift from Models to Compound AI Systems*. <https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/>. 2024.
- [2] I. Beltagy, M. E. Peters, and A. Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL].
- [3] B. Dhingra, J. R. Cole, J. M. Eisenschlos, D. Gillick, J. Eisenstein, and W. W. Cohen. “Time-Aware Language Models as Temporal Knowledge Bases”. In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 257–273. ISSN: 2307-387X. DOI: 10.1162/tac1_a_00459. URL: http://dx.doi.org/10.1162/tac1_a_00459.
- [4] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre. *Improving language models by retrieving from trillions of tokens*. 2022. arXiv: 2112.04426 [cs.CL].
- [5] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig. *Active Retrieval Augmented Generation*. 2023. arXiv: 2305.06983 [cs.CL].
- [6] J. Weston and S. Sukhbaatar. *System 2 Attention (is something you might need too)*. 2023. arXiv: 2311.11829 [cs.CL].
- [7] N. Craswell. “Mean Reciprocal Rank”. In: *Encyclopedia of Database Systems*. Ed. by L. Liu and M. T. Özsu. Boston, MA: Springer US, 2009, pp. 1703–1703. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_488. URL: https://doi.org/10.1007/978-0-387-39940-9_488.
- [8] S. Falk and J. Plüer. *Wenn Chatbots antworten*. Available online at: <https://www.duz.de/beitrag/!/id/1617/wenn-chatbots-antworten>. 2024.
- [9] Y. Bengio, R. Ducharme, and P. Vincent. “A neural probabilistic language model”. In: *Advances in neural information processing systems* 13 (2000).
- [10] I. Sutskever, O. Vinyals, and Q. V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 [cs.CL].
- [11] T. Gao, X. Yao, and D. Chen. *SimCSE: Simple Contrastive Learning of Sentence Embeddings*. 2022. arXiv: 2104.08821 [cs.CL].
- [12] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan. *Query Rewriting for Retrieval-Augmented Large Language Models*. 2023. arXiv: 2305.14283 [cs.CL].

- [13] O. Khattab and M. Zaharia. *ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT*. 2020. arXiv: 2004.12832 [cs.IR].
- [14] L. Gao, X. Ma, J. Lin, and J. Callan. *Precise Zero-Shot Dense Retrieval without Relevance Labels*. 2022. arXiv: 2212.10496 [cs.IR].
- [15] L. Wang, N. Yang, and F. Wei. *Query2doc: Query Expansion with Large Language Models*. 2023. arXiv: 2303.07678 [cs.IR].
- [16] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen. *Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy*. 2023. arXiv: 2305.15294 [cs.CL].
- [17] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou. *Take a Step Back: Evoking Reasoning via Abstraction in Large Language Models*. 2024. arXiv: 2310.06117 [cs.LG].
- [18] X. Wang, Q. Yang, Y. Qiu, J. Liang, Q. He, Z. Gu, Y. Xiao, and W. Wang. *KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases*. 2023. arXiv: 2308.11761 [cs.CL].
- [19] Z. Rackauckas. “Rag-Fusion: A New Take on Retrieval Augmented Generation”. In: *International Journal on Natural Language Computing* 13.1 (Feb. 2024), pp. 37–47. ISSN: 2319-4111. DOI: 10.5121/ijnlc.2024.13103. URL: <http://dx.doi.org/10.5121/ijnlc.2024.13103>.
- [20] D. Cheng, S. Huang, J. Bi, Y. Zhan, J. Liu, Y. Wang, H. Sun, F. Wei, D. Deng, and Q. Zhang. *UPRISE: Universal Prompt Retrieval for Improving Zero-Shot Evaluation*. 2023. arXiv: 2303.08518 [cs.CL].
- [21] O. Yoran, T. Wolfson, O. Ram, and J. Berant. *Making Retrieval-Augmented Language Models Robust to Irrelevant Context*. 2023. arXiv: 2310.01558 [cs.CL].
- [22] B. Wang, W. Ping, L. McAfee, P. Xu, B. Li, M. Shoeybi, and B. Catanzaro. *InstructRetro: Instruction Tuning post Retrieval-Augmented Pretraining*. 2024. arXiv: 2310.07713 [cs.CL].
- [23] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih. *REPLUG: Retrieval-Augmented Black-Box Language Models*. 2023. arXiv: 2301.12652 [cs.CL].
- [24] Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin. *Retrieval-Generation Synergy Augmented Large Language Models*. 2023. arXiv: 2310.05149 [cs.CL].
- [25] J. Liu. *Building Production-Ready RAG Applications*. Available online at: <https://www.ai.engineer/summit/schedule/building-production-ready-rag-applications>. 2023.
- [26] I. Nguyen. *Evaluating RAG Part I: How to Evaluate Document Retrieval*. <https://www.deepset.ai/blog/rag-evaluation-retrieval>. 2023.

- [27] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL].
- [28] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. *Mistral 7B*. 2023. arXiv: 2310.06825 [cs.CL].
- [29] S. Robertson and H. Zaragoza. “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. doi: 10.1561/1500000019.