Mathematisch-Naturwissenschaftliche Fakultät

Robust ML Group

Masters Thesis

Investigating Retrieval Augmented Generation for Virtual Adaptive Tutors

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Robust Machine Learning Group
Devank Tyagi, devank.tyagi@tuebingen.mpg.de, 2025

Bearbeitungszeitraum: 01.10.2024-02.05.2025

Betreuer/Gutachter: Prof. Dr. Matthias Hein, Universität Tübingen
Zweitgutachter: Dr. Wieland Brendel, Max Planck Institute for Intelligent Systems

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Devank Tyagi (Matrikelnummer 5966585), May 16, 2025

Abstract

Retrieval Augmented Generation (RAG) provides a critical segue between LLM retraining and knowledge enhancement by allowing external data sources to be referred by the response generating LLM without any parametrization, thus improving the reliability, context-awareness and transparency of the generation process. We are motivated by the potential use of virtual adaptive tutor systems in school classrooms, which requires constant updation of the tutor's knowledge base and accountability for replies provided to students based on textbook resources, which makes RAG an effective solution. We first implement popular RAG systems in increasing order of complexity, moving from purely vector-based systems to graph-based systems. Our primary contribution is an external RAG system-specific evaluation pipeline, that provides a quantitative basis of comparison between the different systems within a specific domain. The evaluation pipeline fulfills a need in existing literature, since we were unable to find a domain-specific RAG evaluation suite that could be adapted to different domains of implementation. We provide a dataset suite that suits the requirements of an education domain and a simple algorithm to evaluate any RAG system. The source code for the thesis can be found at: `https://github.com/idonthaveacommit/rag-for-virtual-tutors`.

Acknowledgments

They say that every man is an oasis. I have never agreed with this statement, and this course and masters thesis have proved me to be right over and over again. There are a number of people who have my eternal gratitude.

I would like to thank my parents before anyone else, for giving me the opportunity to pursue this degree and forever helping me stay focused when the going got tough. I would also like to thank my grandfather, who is my lifelong motivation to pursue the sciences and who showed me the thrill of a scientific solve for the first time.

I would like to thank Dr. Wieland Brendel for his patience and guidance as the team lead in charge of this project. I also thank Alex Braun for his willing support and cooperation in handling the code and life's surprises. I am honoured to have been a part of this team of erudite minds that have helped shape my own.

I would like to thank Rahi for keeping everything together and for her unwavering belief in my ability to push through. I did.

I thank Siddharth for his constant support and companionship since the first tutorial session I attended in Tübingen. I also want to thank my circle of friends here in the city, who made the feeling of being away from home take a back seat.

Finally, I would like to thank myself for undertaking this journey and coming out a better, wiser person.

Contents

1. Introduction

At the time of writing, Retrieval Augmented Generation (RAG) is one of the most avidly researched topics in the field of Large Language Models (LLMs), which is an endorsement of its appeal and viability. While the family of prevalent LLMs has been quite successful in being able to conduct open-domain conversations, hallucinations remain an indetachable part of their conversation sets [SPC+21]. The ability to converse is borne from data stored implicitly in the pretrained weights of the LLM, which is a fixed phenomenon and is thus inherently devoid of the capability to re-learn or augment that pool of data without a complete overhaul of the training process [GLT+20]. In principle, RAG techniques present an elegant solution to the two prevalent issues of supplementing the knowledge (either chronological or domain-specific) of already-trained LLMs, and addressing hallucinations and attempts by the LLM to synthesize dubious data to answer user queries. It follows that this paradigm would also find potential applicability in the virtual adaptive tutor setting, where we want to set up a conversational agent that can refer to textbook/study materials and reliably answer student questions without forgoing that material grounding. Moreover, the ever-moving goalposts and advancements in learning and school subjects compel us to find solutions that can allow incremental updation of the tutor knowledge base without incurring the base costs of re-training and fine-tuning the response-generating LLM.

This project is thus motivated by a need to not only investigate and compare pre-existing RAG techniques, but to do so under the banner of the educational domain. This domain brings several additional requirements for the model to remain viable and interactive in a real-time setting. The RAG system needs to be able to:

- Provide references and citations to the source material as needed, in order to verify the absence of hallucinated facts in the responses.
- Retrieve information and respond over longer context lengths than a single-turn conversation, i.e. be able to function well over a continued dialog with the user.
- Retrieve and respond with low latency, in order to avoid degradation of user experience.
- Scale well with increases in the size of the retrieval knowledge base.
- Preserve a semantic understanding of the articles/sources in the dataset that retrieval will be carried out from, in order to enhance the quality of con-

textual retrieval and minimize irrelevant or factually incorrect responses.

- Ideally learn to retrieve multimodal data (images, videos, tasks) and integrate these into the context provided to the LLM to answer the question.

With a veritable myriad of RAG techniques to choose from and a fast-moving research community finding new solutions by the week, it has also become necessary to have a unified method of comparing these different techniques and models on the basis of their retrieval quality and response quality, while ensuring fidelity with the domain-specific requirements. This heralds the need for a benchmark or test suite that can perform these comparative checks and quantitatively assess RAG techniques for their effectiveness and applicability. At the time of writing, we were unable to find any widely accepted benchmark or unified comparative technique exists in this domain. All evaluation frameworks are either general [EJEAS25] [FBS25] or focus solely on the response-generating LLM as a point of difference [LPR+24] [SFKPZ24].

1.1. Problem Statement

This focus of this work is to implement pre-existing RAG models from lower to higher complexity, in order to gain a working knowledge of the moving pieces and have candidate models that can be attached to enhance the response generation of a conversational LLM in the classroom setting. This shall be followed by a detailed proposal for a domain-specific evaluation pipeline that could compare the performance and enhancement capabilities of these implemented RAG models in quantitative fashion. Finally, we shall use this evaluation pipeline to test our candidate models and obtain some intuition as to which class of RAG systems works best in our chosen domain.

We initiate this project by delving into baseline RAG models that serve the purpose in a bare-bone manner. The base models shall be enhanced with simple modifications to the retrieval mechanism and query comprehension models, which will ultimately lead to the state-of-the-art, complex RAG systems that are favoured at the time of starting. Once our base models are ready, we shall begin the development of the evaluation pipeline.

Our initial attempts considered generating artificial data that could simulate school materials such as textbooks and literature-related articles. However, as we noticed the absence of any ironclad evaluation suites for any specific knowledge domains, we decided to devote our efforts towards the development of a reusable mode of evaluation with reliable metrics and interpretable results that could be extended into any domain with its own set of restrictions and requirements.

2. Background and Related Work

2.1. Retrieval Augmented Generation

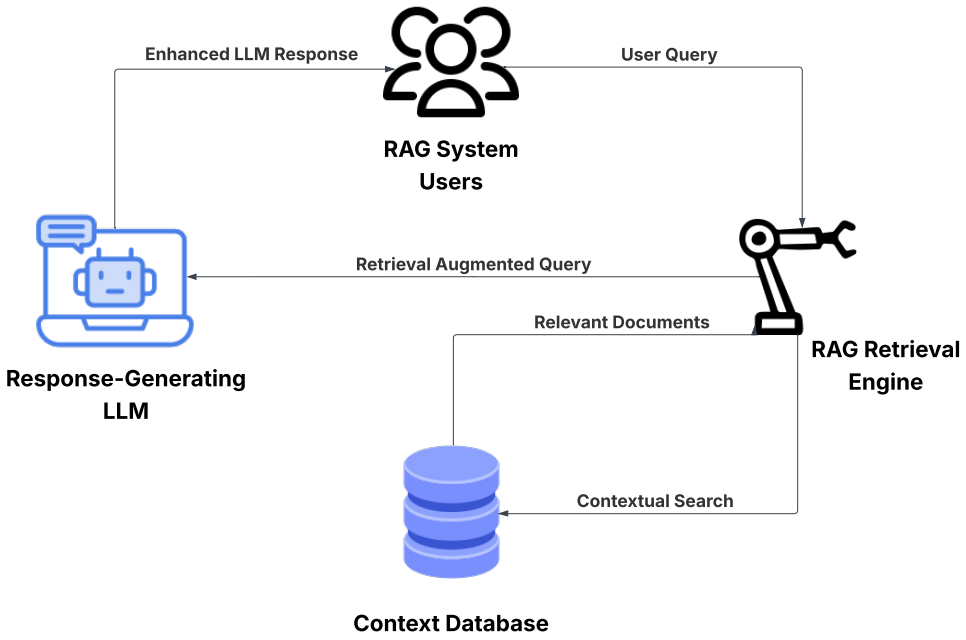

Figure 2.1.: The RAG framework in a nutshell.

While Retrieval Augmented Generation was birthed in its current state in [LPP+21], the seeds of the traditional RAG framework were planted in previous years across natural language processing tasks where domain-specific queries were problematic to answer for a regular language model fine-tuned on a large corpus of general data. The efforts of [GLT+20] conceived the knowledge retrieval engine to be used in conjunction with language representation models such as BERT [DCLT19], although the paper models this retrieval process as a latent variable model that augments the data used to fine-tune the model. We quickly realized that the simplification to a non-parametrized, engineering-inspired approach performed admirably without incurring the significant costs related to LLM fine-tuning.

The basic idea of a RAG system is illustrated in Figure 2.1. The users of the system send their queries to the RAG system as they would to a vanilla LLM, since the RAG architecture is abstracted from them. The user query is first

directed to the RAG retriever engine instead of directly reaching the LLM. The retriever engine preprocesses the query and performs a very efficient search of the context document database, which houses the external data meant to augment the response generation process. Relevant documents, called contexts, are selected and retrieved by the retriever engine, which then concatenates the query with these augmenting contexts and sends the package to the waiting LLM. The LLM agent utilizes the augments to provide a higher quality response that is ideally grounded in the facts provided by the external data and possibly more informative than the response generated by the vanilla LLM could have been.

The basic features of a good RAG system thus follow:

- **Efficient Retrieval**: The retrieval process is a supplement to the pre-existing query-response paradigm that conversational LLMs follow. This process can thus introduce latency at inference time, which can severely degrade user experience. However, we cannot sacrifice the quality of context retrieval to maintain low latency either, since the purpose of the feature is to introduce reliable external data resources.
- **Query and Context Database Preprocessing**: A simple word-matching algorithm is highly inefficient and can cause severe delays in response generation. Hence, the context document database is usually preprocessed to present its documents in a format that is quick to go through and easy to perform query matching with. In that spirit, the query is preprocessed using either the same or different recipes as well.

When we refer to RAG models in this work, we refer to the combination of the RAG retriever engine (or just retriever), the context document database and any extra moving parts that contribute to the enhancement of context retrieval. Since the initiation of research in this field, there has been an explosion of advancements and modifications that introduce engineering-based or artificial intelligence-based solutions and enhancements.

2.2. Related Work

Various research into the current best practices for RAG [WWG+24] [GXG+24] [ZZY+24] recommend different classifications of RAG techniques prevalent today. One school of thought separates RAG systems into naive RAG that follow the simple linear workflow of query → retrieval → response, advanced RAG that introduces pre-retrieval and post-retrieval modifications [GMLC22] and query enhancements methods [MGH+23], and modular RAG that modularizes these components and introduces the concepts of iterative and adaptive retrieval. Another school of thought instead splits these models based on their retrievers' methods of augmenting the LLM. This gives rise to categories such as speculative RAG [AWW+23] that attempts to learn when external retrieval should even be attempted, and latent representation-based RAG such as [IG21]. For the purpose

of this study, we stick to the classic prompt-based RAG models that augment the prompt with fixed contexts that show similarities to the queries in question.

The RAG framework has specific research happening in every stage. The performance of the RAG model is significantly affected by the chunking strategies employed [ZLC+25], since these chunks serve as our retrieval contexts. Different embedding models are present for use, ranging from simple-use models from OpenAI to trainable embedders that optimize their training methodology based on the type of content encountered to be embedded [ZXL+23]. Our takeaways from this set of research is to start with the simplest RAG models and work our way upwards in terms of complexity, leaving these upgraded moving parts to be trialled by future researchers.

We also look at the evaluation benchmarks and suites already present in existing literature. RAGAs [EJEAS25] is an automated RAG evaluation framework developed by DeepMind AI, that uses the LLM-as-a-judge technique and allows flexible metric creation along with a set of predetermined metrics to choose from. We trialled this evaluation framework and achieved interesting preliminary results - however, the framework demands evaluation data to be synthesized in a very particular form for its use and is suitable only for single-turn evaluation. ARES [SFKPZ24] provides its own synthetic data generated from a few human-annotated samples. They were able to show results scaling across unseen domains, but did not actually evaluate any RAG techniques and focused on the innate ability of the vanilla LLMs to identify when the query can be answered using provided context. More recently, Comprehensive RAG [YSX+24] was released as a RAG evaluation benchmark where they show strong evaluation strength for vanilla and RAG-based language models. However, this benchmark too is unsuitable for domain-specific RAG evaluation.

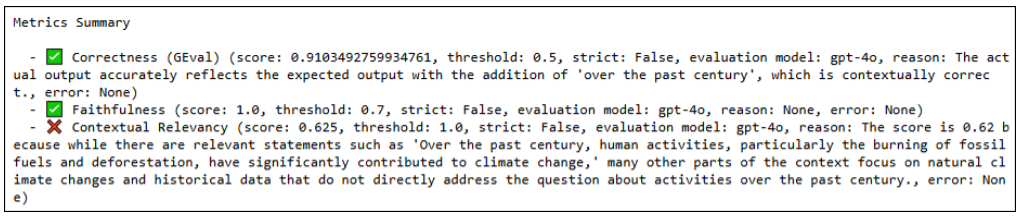

Figure 2.2.: An instance of evaluation provided by the RAGAs framework, with three in-built metrics for RAG evaluation. The framework also prompts the LLM to explain the reasoning behind the metric scores.

We had selected CORAL [CMZ+24] as a reference benchmark due to their claims of creating a multi-turn evaluation benchmark sourced from Wikipedia articles that tackled natural conversation patterns such as topical shifts, progressively knowledge-intensifying dialogs and freeform responses altogether in one dataset. To our disappointment, the evaluation benchmark source code is not being maintained and runs into errors, while the curated dataset is not available on any public platform anymore.

3. The Models

Our segmentation of the different RAG systems to be implemented is based upon the method of preprocessing and indexing done with the context document database before inference. We thus classify the RAG systems into two categories - **vector-based RAGs** and **graph-based RAGs**.

- **Vector-based RAG** systems utilize vector similarity matching using a mathematical distance metric such as the cosine similarity [LPP+21]. They use text embedding models to convert the context documents into embedding vectors, which are stored as keys to these context documents. At inference time, the user query is converted into a similar embedding vector and then matched very quickly with a potentially large set of context embeddings. The documents with embeddings most similar to the query embedding are retrieved for query augmentation. These vectors are stored in a vector database such as FAISS [DGD+25] or Milvus [WYX+22].

- **Graph-based RAG** systems create a Knowledge Graph (KG) of either the corpus documents or specific entities and relationships contained within those documents. Knowledge graphs allow preservation of semantic meanings contained within the text, which help create connections between the various entities/documents as the edges of the graph. Depending upon the implementation, these edges contain information such as size or descriptions about how the nodes of the graph are connected, At inference time, the query undergoes a similar form of entity extraction (if needed) and this structure is used to pinpoint the region where the relevant contexts reside in the knowledge graph, which offers optimized traversal and reduces search latency. The method of isolating the region of correct context residency differs from method to method.

Table 3.1 details further key differences in the two types of systems. The vector-based systems are much more simplistic in implementation, so we look at these systems first. We attempt to standardize the parameter values shared across multiple models to ensure similar templates of responses generated and contexts retrieved.

We will be implementing three vector-based approaches and two graph-based approaches as part of this thesis, to be evaluated by the evaluation pipeline in the next section.

Aspect	Vector-based RAG	Graph-based RAG
Retrieval Method	Dense or sparse vector similarity (cosine similarity, dot product etc.)	Graph traversal or subgraph extraction (eg: Personalized PageRank [Cir24])
Data Structure	Vector index (e.g., FAISS, Milvus)	Knowledge graph or semantic graph
Knowledge Representation	Unstructured text chunks represented as embeddings	Structured entities, relations and possibly text annotations
Query Encoding	Encoded into embedding vectors using transformer models (BERT [DCLT19], text-embedding-3-small)	Entity linking or query graph construction
Retrieval Granularity	Typically retrieves top-k documents or passages	Retrieves subgraphs, paths or relevant entities
Advantages	Fast and scalable for medium-sized data stores, works well with unstructured data	High interpretability, supports reasoning over structured knowledge
Challenges	May retrieve similarly embedded but irrelevant text, lacks explainability	Requires structured data and graph construction, more complex pipeline

Table 3.1.: A detailed comparison between vector-based and graph-based RAG systems.

3.1. Base RAG

This implementation follows from the traditional, linear RAG model as presented in [LPP+21]. Figure 3.1 shows a detailed description of the control flow in this RAG model. Before any inference takes place, the context documents are chunked with some overlap allowed. They are then converted into embedding vectors and stored as keys in the vector database for efficient similarity matching. At inference time, the user query is accepted and converted using the same embedding model into its embedding vector. This vector is used to match up with the context embeddings to find the ones with the least mathematical distance. We select the k (a hyperparameter) nearest ones and the original query is concatenated with the relevant contexts, before this augmented query reaches the response-generating LLM.

This approach is very straightforward in application and does reasonably well for small context databases where sequential retrieval and vector similarity matching is computationally inexpensive. We use chunk sizes of 600 tokens each, with an overlap of 100 tokens allowed. For embedding, we use the *text-embedding-3-small* transformer model as provided by the OpenAI library [Ope24]. We designed the Base RAG approach using two popular platforms - LangChain [Cc22a] and LlamaIndex [Cc22b]. While LlamaIndex provides better end-to-end indexing solutions and faster RAG implementation, we selected LangChain as

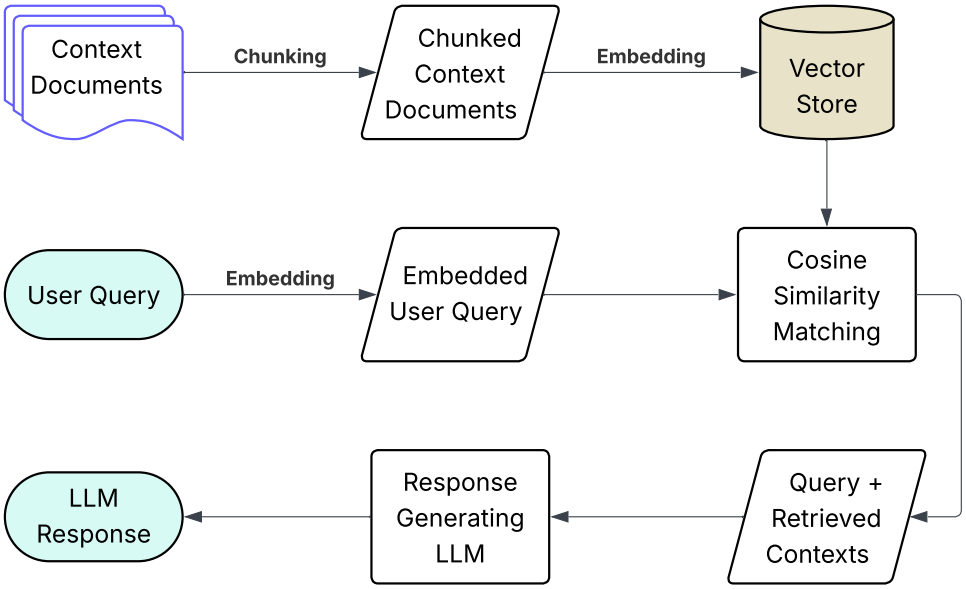

Figure 3.1.: A flowchart depicting the flow of control in Base RAG.

our framework moving forward because of the intuitive Chain-of-Thought (CoT) reasoning-inspired modularity and chaining mechanisms. We selected FAISS as our vector store, and the GPT-4o-mini model is used as our response-generation LLM. All of these design decisions remain consistent across the implementations of the other RAG models as well.

3.2. Hypothetical Document Embeddings (HyDE)

Base RAG runs into a few problems when the query is worded in specifically difficult ways. If the user query contains words that the context documents have not seen or if the query is ambiguous or poorly worded, standard RAG can suffer and become unable to retrieve the right context. In these situations, a bridge is needed to connect what the user is saying and what they are meaning to say. We attempt to build this bridge using a query rewriting technique called Hypothetical Document Embedding (HyDE) [GMLC22].

Figure 3.2 shows the miniscule change in control flow of the RAG system employing the HyDE technique. While everything else remains the same, the user query is sent to an LLM to generate a hypothetical response to the query, determined completely on the basis of data implicitly learned by the LLM creating the document. The principle behind undergoing this step is that no matter how bad the hypothetical response is, it will still contain more marker words that can help find similarly-worded context documents. Even a hallucinated answer usually helps in figuring out the premise of the query, which makes the model quite robust to hypothetical answer failure. We used the GPT-4o-mini to generate

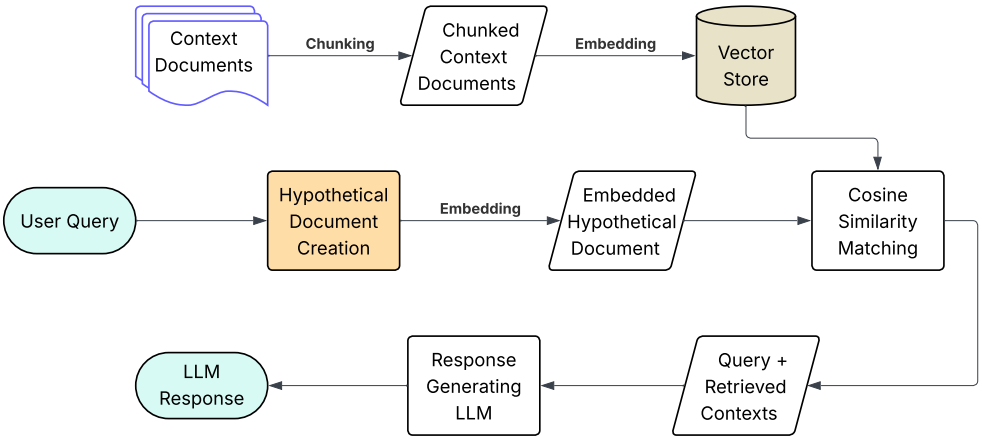

Figure 3.2.: A flowchart depicting the flow of control in HyDE. The only addition is the creation of a hypothetical document from the user query during inference.

Original Query	**Hypothetical Document**
Can technology play a big role in causing climate change to reduce in today's world?	**Title: The Role of Technology in Mitigating Climate Change** Technology can significantly reduce climate change impacts in today's world. Renewable energy sources, such as solar and wind, have advanced, making them more efficient and cost-effective. These technologies decrease reliance on fossil fuels, which are major contributors to greenhouse gas emissions. Smart grids enhance energy distribution, optimizing consumption and reducing waste. Electric vehicles (EVs) are becoming mainstream, lowering emissions from transportation. Innovations in carbon capture and storage (CCS) technology allow industries to mitigate their carbon footprints. Moreover, precision agriculture utilizes data analytics to optimize resource use, minimizing environmental impact. Sustainable building technologies improve energy efficiency in construction. However, technology alone cannot solve climate change; it must be paired with policy changes, public awareness, and global cooperation. By harnessing technological advancements, we can create a sustainable future and significantly reduce climate change effects.

Table 3.2.: An example of the hypothetical document constructed from a simple user query.

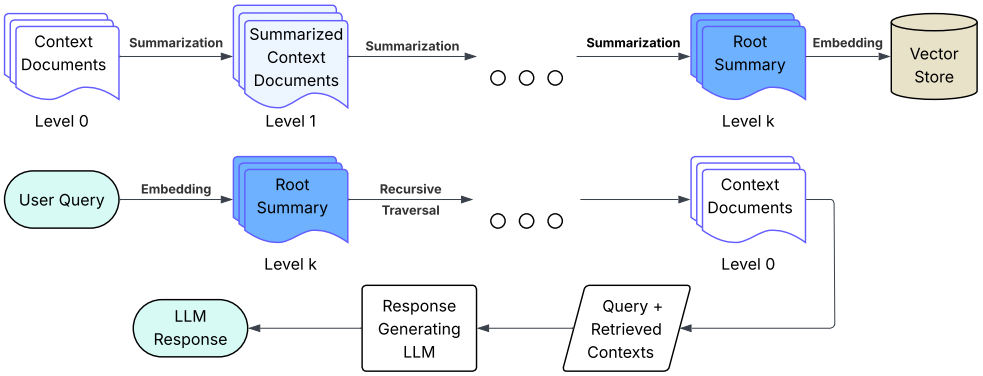

Figure 3.3.: A flowchart depicting the flow of control in RAPTOR. Hierarchical summarization and child-parent relationships allow for narrowing of the search context search space.

this hypothetical document as well, but it is perfectly acceptable to use an even smaller, cheaper model. Table 3.2 shows an example of a hypothetical document encountered on our sample dataset.

3.3. Retrieval-Augmented generation with Prompting and Tree-Oriented Retrieval (RAPTOR)

The previous models both searched the entire context document space for relevant candidates, while it is easily imaginable that the relevant contexts are actually present in a narrow, semantically coherent segment of the document corpus. If we can leverage the structure of the documents such that we can start finding higher-level headings and subheadings that are relevant to the user query, we can narrow the search space down. RAPTOR [KPX+23] is a RAG technique that uses hierarchical summarization over multiple levels to utilize the structure of the document/s provided for contextual recall. It is inspired by tree search methods in information retrieval, and is the first model we implement that actually considers the loss of information through embedding, and attempts to preserve some of that information. The RAPTOR algorithm was originally designed to work on a single long document that could be recursively broken down into its constituent structural atoms, but we simply use our document chunks as the Level 0 nodes in the process.

Figure 3.3 gives a breakdown of what RAPTOR does behind the scenes. The original context documents are considered as leaf nodes, and an LLM begins to cluster and summarize the documents in a bottom-up fashion to create new parent nodes that contain the summaries. This process is repeated until a root node is approached (or up till $k = 5$ in our implementation). The summaries are embedded and stored in the vector store as well. During inference, the embedded user query is compared with the highest-level embedded summaries.

The summaries that are closest to the user query act as parent nodes of the subtrees that the algorithm recursively goes down, till it finally reaches the leaf nodes (original contexts) and searches for the most relevant contexts. This vastly reduces the number of documents that the user query needs to check.

We note that this approach is more useful when we have only a few documents that have been broken up into multiple segments, since the clustering and summarization would yield meaningful abstractions. When we have multiple short-length texts that are independent of each other as our leaf nodes, RAPTOR is expected to have a much more difficult time and actually perform worse than usual in terms of response generation (even though it should still take lesser time in inference).

3.4. Microsoft GraphRAG

RAPTOR acts as the segue between vector-based models and graph-based models, since it utilizes a graph (tree) to traverse through a collection of embedding vectors. We now enter the space of pure graph-based models with Microsoft GraphRAG [ETC+25], the most popular graph-based approach. Its popularity owes to the ease of casual use due to the visualizations and sandbox provided, along with the ability to outperform most other RAG models in terms of context retrieval, citation provision and global search capabilities. However, our use case does not require the global search capabilities, which makes the MS GraphRAG a highly complex candidate to execute a simpler, local search on the context database.

Figure 3.4 shows a high-level summary of the indexing and querying processes that are carried out in the GraphRAG architecture. For the indexing process,

- The context documents are accepted raw into the chunking stage.
- Entity and relationship extraction is performed by a language model, which form the nodes and edges of the semantic knowledge graph index respectively.
- The embedded versions of these entities and relationships are stored in a vector store.
- Meanwhile, the nodes are clustered via an algorithm like Leiden clustering to recognize communities. These communities generate reports along with the singular entities through further LLM calls.

The querying process for local search looks as follows:

- The user's query is submitted to an LLM for entity-relationship extraction.
- These elements are used to uncover the starting points for graph traversal. The entities are either matched by simple keyword search, or the embedding vectors of the individual reports are scored.

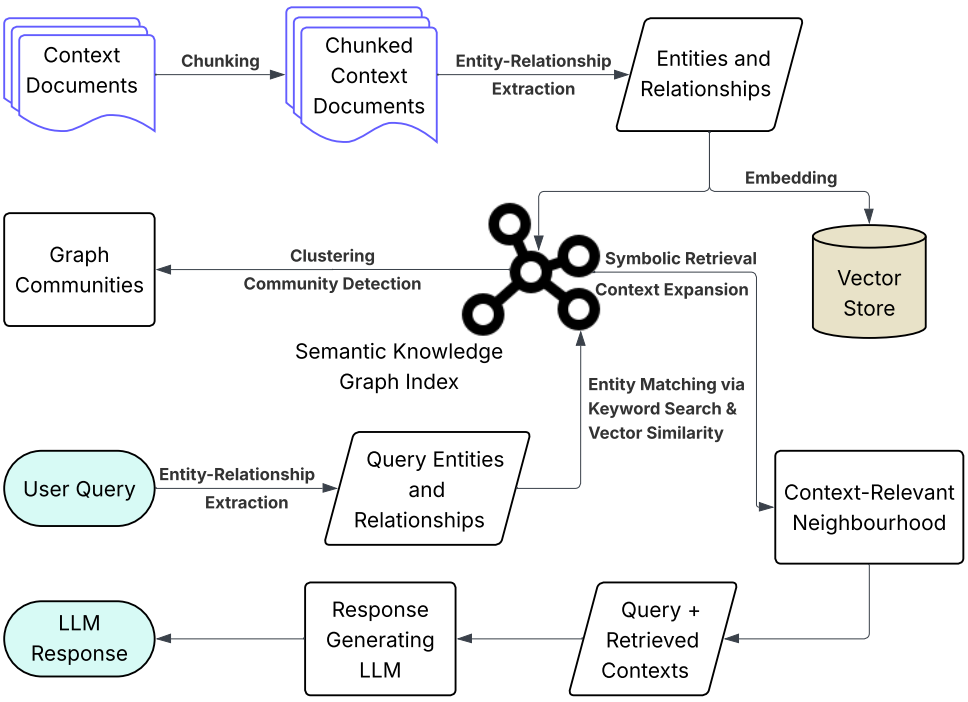

Figure 3.4.: A high-level depiction of the indexing and querying processes in Microsoft GraphRAG. Graph communities and community answers are almost never used for local queries.

- Once this symbolic retrieval is completed, the neighbourhood around the defined area is explored for 1-2 hops, to broaden the context. This finally defines the context-relevant neighbourhood where our documents reside.
- The query is augmented by the texts of the entities and relationships within the neighbourhood, along with descriptions of their connections.
- This augmented query is sent to the response-generating LLM.

Every step of the querying process involves using a separate LLM, which makes this one of the most financially expensive models that we see. The graph communities and community reports and summaries are used for global searches, which is a functionality that we do not test for. This approach is the state-of-the-art in multi-hop reasoning using graph traversal.

3.5. LightRAG

A huge drawback of Microsoft GraphRAG is that updation of the knowledge graph is not incremental, i.e. we would need to run the whole indexing process again to allow the addition of knowledge resources. Combined with the significant number of LLM calls made during a single indexing run, this can become a

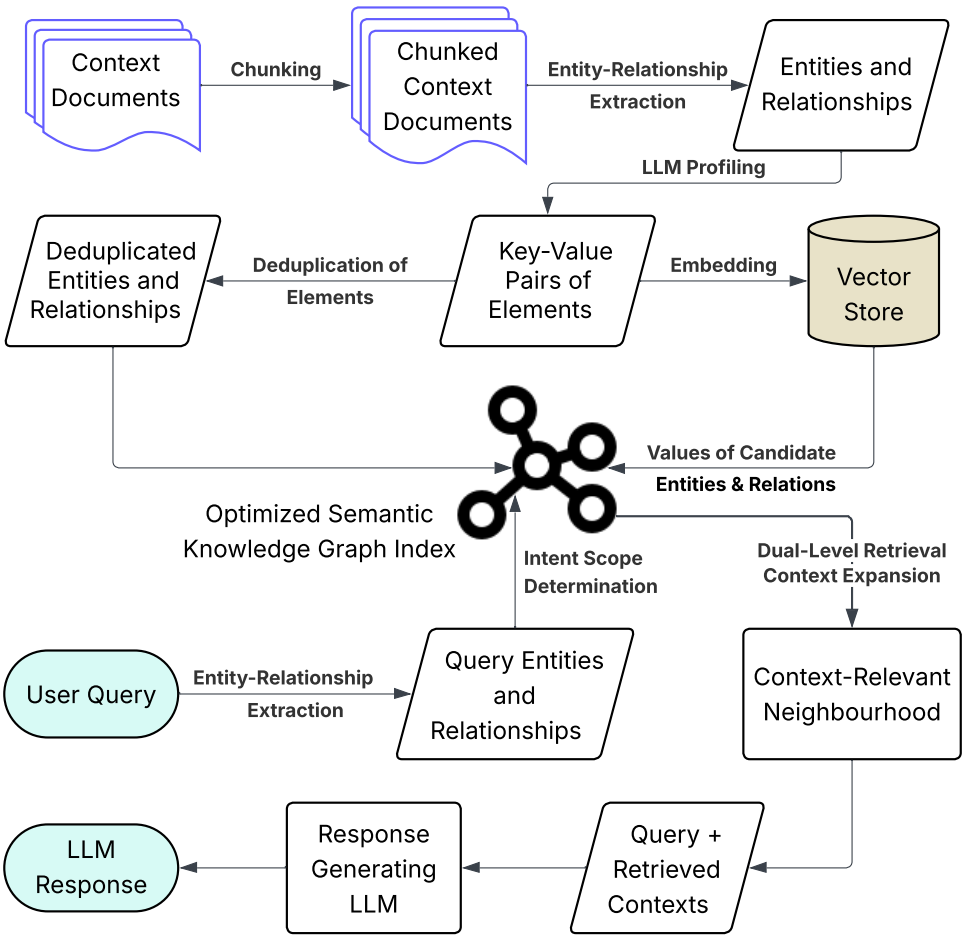

Figure 3.5.: A high-level depiction of the indexing and querying processes in LightRAG. A dual-level retrieval procedure allows for specific as well as abstract details.

cumbersome overhead to manage every time a teacher wants to upload a new textbook, or even a small piece of information. This problem is mitigated by LightRAG [GXY+25], where the knowledge graph updation can be incremental due to no summarization of communities/clusters as in GraphRAG.

The LightRAG algorithm is depicted in Figure 3.5. The creation of the knowledge graph is similar to that of GraphRAG, but with a few important differences:

- An LLM is used to generate key-value pairs for each entity and relationship, where the values are short descriptions that enhance semantic understanding of these elements.
- The entities and relationships are deduplicated, to make sure there is no redundancy in the undirected graph. Duplicate elements are simply taken under union and combined.

- The values are embedded and stored in the vector database, while the entities and relationships combine to form an optimized version of the knowledge graph without community formation.

The querying procedure sees a dual-level retriever being implemented, which sees local and global keywords being extracted from the user query. The local keywords are used for low-level retrieval, in which the candidate entities are mapped to these keywords to identify the starting points of the context-relevant neighbourhood. At the same time, the global keywords are mapped to relations connected to global values, which gives us a better understanding of the direction of traversal in the knowledge graph. Finally, a double-hop expansion of the neighbourhood gives us our desired entity-relationship combinations that provide the desired context augmentation.

LightRAG is a relatively lighter model in terms of financial and inference time cost which still manages to maintain the power of semantic understanding that graph-based systems have, by using an intelligent retrieval system that leans on vector-based approaches for speedups. We used the standard provided values for all parameters except chunking size when implementing this model, and kept track of the original contexts even during the entity-relationship extraction.

4. The Evaluation Pipeline

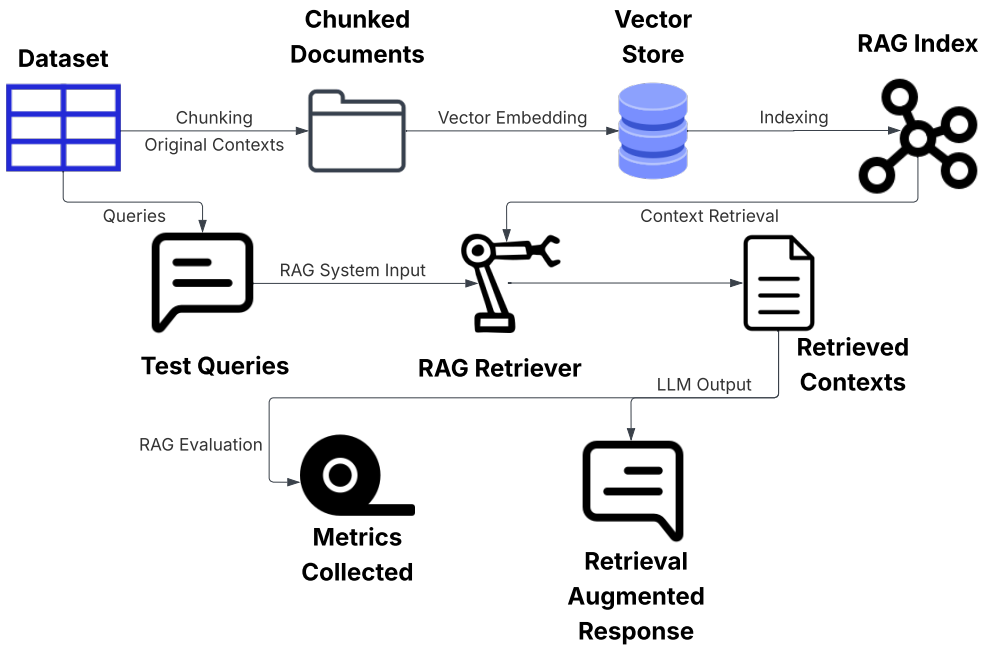

Figure 4.1.: The overall evaluation pipeline.

The RAG models implemented in the previous section successfully generate answers to simple queries in varying amounts of time and costs behind them, and with differing degrees of success when it comes to picking the correct context present in the contextual document store. However, manual perusal is ineffective in distinguishing between the various models and their usability for our specific use case. Although it is easily observable through brute force that naive RAG models perform less poorly than large-scale ones, we see the need for a standardized, quantitative evaluation pipeline that can generate multiple metrics for the implementer to choose from and make a more informed decision as to which model to use.

Figure 4.1 illustrates the RAG model evaluation pipeline designed and proposed as part of this work. For each dataset selected from a predetermined collection of question-answering datasets, we pick the underlying base contexts for the queries present and convert them into chunked documents that fit the context length of the large language model. The chunked documents are embedded (if the RAG model demands it) and stored in a vector embedding store. The RAG

system accesses this vector store, either for direct retrieval or to construct an index for the retriever to use.

The evaluation queries are then selected from the original data set and given as input to the RAG system as questions that can be asked of the virtual tutor. The retriever of the RAG system uses this input to perform context retrieval using the specific technique pertinent to the model. The retrieved documents are ultimately supplied to the base LLM as the augmented context that is used to answer each test query. In addition, these contexts and the performance of the entire system are also examined quantitatively to provide the metrics of measurement for the particular RAG model.

We also test the RAG system's ability to recognize unanswerable scenarios, i.e. to be able to identify when none of the contexts present in the contextual document database provide a correct answer to the given input query, as inspired from [LPR+24]. This analysis is done by using specific datasets that contain these unanswerable queries, and we observe the capability of the RAG models we test to supply a non-answer instead of a hallucinated response. We use the counts of the number of queries that were truly (and falsely) unanswerable and the counts of the queries that were correctly (and incorrectly) classified as unanswerable, and construct our evaluation metrics accordingly.

4.1. Datasets

The collection of question-answering datasets used for this evaluation pipeline was carefully handpicked, with the viewpoint of evaluating one or more important qualities of the RAG system with each dataset. Each of these datasets covers wide fields of information and can have query threads related to a narrow or broad spectrum of topics within the same conversation, which tests the flexibility and adaptability of the system and tries to simulate the real-world setting of a natural conversation between the virtual tutor and an inquisitive student. To model this, the selected datasets usually contain conversations over multiple turns, where the RAG model can access both the query and the conversation history to inform its retrieval.

4.1.1. QuAC (Question Answering with Context)

The QuAC dataset [CHI+18] is a set of simple, freeform, multi-turn dialogs that are grounded on Wikipedia articles. The structure of queries simulates a conversation between a student who only has access to the initial part of an unknown article, and a teacher who has access to the entire article. The student attempts to extract as much information about the article as possible over the course of the dialog, while the teacher is restricted to answer the queries solely with snippets and passages from the article. The dataset is also built to house a significant portion of queries whose answers are not available in these articles, where the teacher is supposed to indicate that they do not have an answer.

We use the development split of the QuAC dataset for the regular analysis, which contains exactly 1K dialogs with a total of 7354 student-teacher turns. We also utilize the high percentage of context-absent turns (1486 or 20.2%) for our unanswerable scenario evaluation. This dataset encourages short answers to the freeform queries, which has been shown to be more effective in assessing core RAG capabilities and and retrieval quality as compared to long answer datasets [BSB25].

4.1.2. Doc2Dial (Document-Grounded Dialog

The Doc2Dial dataset [FWG+20] provides training and evaluation data for scenarios where a user needs assistance pursuing a goal and they have a conversation with an agent regarding a provided document that explains the process of achieving that goal. Using this dataset allows us to simulate conversations where a student may need help with school instructions or a procedure outlined in class, and they look to the virtual tutor for clarification.

We use the development split for this dataset as well, and we focus solely on the textual components of the contexts and not the HTML elements of the documents, since we are more interested in seeing whether we are able to extract the relevant document to answer the question alone. There are 719 conversations spanning 3939 turns, which are grounded in long documents that would otherwise not be suitable to provide as context to the LLM directly.

4.1.3. TopiOCQA (Topic switching in Open-domain Conversational Question Answering)

The TopiOCQA dataset [ADS+22] is the first dataset we use that allows freeform responses based on the contexts needed to answer the question. As the name suggests, we assume interconnected topical switches through the course of a conversation to be a natural occurrence. Retrieving the right context to answer queries present in this dataset is a greater challenge, because the conversation history may actually trick the retriever into selecting contexts related to subjects that are no longer relevant to the dialog. The domain of contexts used here are Wikipedia articles, similar to QuAC.

We use the development split of this dataset, which contains 205 different dialogs and sums to 2514 turns between the system and the questioner. Each dialog covers a maximum of seven different topics and does not exceed twenty turns. The system answers expected are freeform and not conformed to directly quote the context, instead focusing on providing natural language answers.

4.1.4. CoQA (Conversational Question Answering)

The CoQA dataset [RCM19] contains dialogs grounded in text passages that originate from seven different domains, including children's stories, Reddit, and

literature among others. This selection combines offbeat contexts that are less encountered as factual occurrences with freeform responses, which simulates the scenario of referencing a textbook to respond to the student's queries, regardless of subject. To answer the student queries provided, the text passage needs to be "understood" by the model using prior real-world knowledge.

The development split of this dataset contains 500 conversations with a total of 7983 different turns. The questions in these turns can be repetitive and indistinguishable without context, which provides a good test of the retriever's capability to extract the best possible answer for each scenario separately. The LLM is encouraged to provide short, succinct answers to queries from this dataset. The text passages that provide the contexts for these queries are shorter than the documents present in the previous three datasets, measuring around 450 words on average.

4.1.5. DoQA (Domain-specific Question Answering)

DoQA [COS+20] is a multi-domain question-answering dataset that references domain-specific FAQs across the three specific domains of cooking, travel and movies. The FAQs are sourced directly from StackExchange websites and attempt to provide real-world instances of user queries and answers provided by other users, instead of existing factual statements. This dataset was selected for its expressiveness and its unique selection of the contexts being preexisting human responses with possibly no factual backing, which may mimic a school setting as a teacher's instructions tailored for the student.

The test set of this dataset is used, which contains 400 dialogs in each of the three domains mentioned above and 1797, 1713 and 1884 turns for cooking, travel and movies respectively. This dataset also contains a significant proportion of unanswerable queries (ranging between 22-30% for each domain), which we use along with the QuAC queries to perform our unanswerable scenario analysis.

4.2. Contextual Document Database

The contextual document database consists of all possible contexts/documents related to the dataset being used for evaluation. Ideally, the contexts should be retrieved from the original sources that are used to construct the user-agent turns and dialogs, which would span entire Wikipedia articles in the case of QuAC and TopiOCQA, long manuals for Doc2Dial, text passage excerpts for CoQA and StackExchange forums for DoQA. However, we do not have access to the original contexts for most of these datasets, since they create a hidden document by design that is actually one of the aforementioned sources. All of the datasets we use are constructed to quantify the performance of regular LLMs as information-gathering machines and not to actually evaluate RAG performance.

We worked around this roadblock by extracting all the correct contexts present

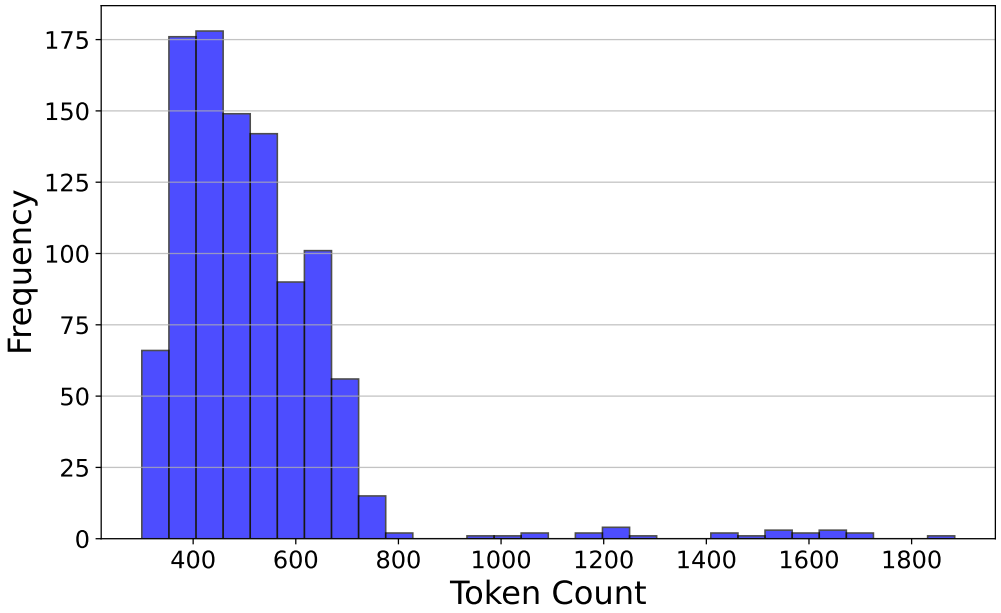

Figure 4.2.: An example histogram of the retrieved contexts from the QuAC dataset, with over 500K tokens comprising the raw contexts retrieved from all queries present.

in each of the datasets and creating our contextual document databases directly from them. This approach is justified by the quantity of contexts that we still obtain through this approach. An example is provided through Figure 4.2, which shows a histogram of the various contexts retrieved from the QuAC dataset's development split. We obtain a thousand different contexts with an average size of about 520 tokens per document. The total size of this contextual document database is about 520K tokens, which is the total token amount for multiple novels or extensive research journals. This simulates the real-world scenario where multiple textbooks and sources are fed into the document database to provide contexts for retrieval.

A similar process is carried out for all datasets in this evaluation pipeline except CoQA, for which the text passage contexts are readily available. The raw contexts are either directly given as inputs to the RAG indexing process, or first embedded as vectors and stored in a FAISS vector database [DGD+25].

4.3. Metrics Used

The metrics used as part of the pipeline are split into two categories, the regular query scenario metrics and the unanswerable query scenario metrics. The former focus on the performance of the RAG system in terms of resource usage and retrieval quality, while the latter looks at the number of queries classified as unanswerable correctly (or otherwise). In our experiments, we will interpret

these metrics in the manner that best suits our use case - however, different users may be able to make different decisions as they desire, based on the metrics' values.

4.3.1. Regular Evaluation Metrics

- **Cost (in cents)**: The monetary cost of carrying out a specific operation in the pipeline, measured in US cents. These costs include the OpenAI pricing for using the GPT-4o mini models to carry out the RAG model processes responsible for indexing the contextual data provided (CI), as well as the cost of generating responses at inference time (CQ). We show the split between indexing costs and inference costs separately.
- **Time (in seconds)**: Provided the same computational resources, which RAG model takes the most time to become ready to generate responses, and which model is able to generate responses quickly at inference time? We measure the time taken for indexing (TI) and the latency during inference (TQ) separately.
- **Context Retrieval Recall (CRR)**: Each of the RAG models works towards retrieving the contexts with highest fidelity with the user query. For every query, we select the first k contexts with the highest retrieval score (or simply the best k retrieved contexts) and return a 1 if the expected context is one of the k retrieved contexts, else we return a 0. The Context Retrieval Recall score is finally calculated by taking the average score for all the queries in a given dataset, per RAG model.

$$\text{CRR} = \frac{1}{N}\sum_{i}^{N} \mathbb{1}\{C_i \in C^*_{ki}\}$$

where N is the number of queries, C_i is the expected context for the i^{th} query and C^*_{ki} is the set of k most relevant contexts picked by the retriever. For our experiment, we select $k = 3$.

4.3.2. Unanswerable Scenario Evaluation

Unanswerable queries are queries that cannot be answered correctly by any context present in the contextual document database. For our use case, a positive result is defined as the model classifying a query as unanswerable. Within that view, we are able to define a confusion matrix using the following metrics.

- **True Positives (TP)**: A count of all the truly unanswerable queries that were classified as unanswerable.
- **False Positives (FP)**: A count of all the answerable queries that were classified as unanswerable.

- **True Negatives (TN)**: A count of all the truly answerable queries that were classified as answerable.
- **False Negatives (FN)**: A count of all the unanswerable queries that were classified as answerable.
- **Precision**: A measure of the RAG system's ability to classify unanswerable queries as such only when they are truly unanswerable. A high precision score indicates that the positive classifications will usually be correct.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: A measure of the RAG system's ability to correctly classify all the truly unanswerable queries. A high recall score indicates that if there are any queries that should classify as positive, the model will usually classify them positively.

$$\text{Recall} = \frac{TP}{TP + FN}$$

There are many other binary score metrics that can be utilized based upon the confusion matrix, but we shall primarily be looking at the precision and recall metrics since they serve as building blocks for other, more nuanced metrics.

4.3.3. Response Generation Evaluation

Evaluating the performance of RAG systems should also include response generation evaluation, which was originally included as part of this study. We used an F_1-score metric to evaluate the correctness of the responses upon matching them with the expected responses, which were also part of the selected datasets. We went about generating this score as detailed in [LPR+24], through the LLM-as-a-judge method where a small-scale LLM compares the generated and expected responses and returns a numeric estimation of their similarity through an F_1-score. However, using an LLM to generate numerical evaluations is fraught with risk, since LLMs are biased in favour of responses with lower token perplexity and exhibit inexplicable changes in evaluation when the response text is modified in a manner that is indistinguishable in meaning for a human evaluator [SAS24]. Furthermore, using F_1-scores as propagated in multiple corporate tutorials such as [AI23] also seemed like a step in the wrong direction, since a token count-based metric is prone to returning great evaluation scores for responses that are perfect negations of the expected response, among other edge cases.

Eventually, we decided to place the response generation evaluation module outside the scope of this study. Response generation is a factor of the quality of retrieval that provides the foundation to build the response on, and the quality of the language model that accepts these inputs and generates the actual response. Since we will be using the GPT-4o-mini LLM to generate our responses in every

scenario of the evaluation pipeline, we believe that the quality of response generation would not be significantly different for quantitatively similar context retrieval methods.

4.4. Experiment and Results

		BaseRAG	HyDE	RAPTOR	GraphRag	LightRAG
QuAC	CI	0	0	1008	6917	2331
	CQ	0.09	0.11	0.14	0.3	0.19
	TI	0	0	437	2556	2893
	TQ	59	58	23	14.1	7.7
	CRR	51.58%	65.17%	67.73%	88.39%	87.55%
Doc2Dial	CI	0	0	917	6218	1921
	CQ	0.1	0.12	0.15	0.32	0.21
	TI	0	0	408	2491	2731
	TQ	67	68	27	14.4	7.5
	CRR	31.04%	47.09%	60.30%	81.71%	80.93%
TopiOCQA	CI	0	0	1252	8389	2566
	CQ	0.09	0.1	0.14	0.29	0.19
	TI	0	0	588	2937	3111
	TQ	65	67	28	13.8	7.7
	CRR	30.62%	49.39%	70.14%	91.53%	89.36%
CoQA	CI	0	0	754	5031	1636
	CQ	0.1	0.11	0.16	0.31	0.22
	TI	0	0	282	1885	2109
	TQ	51	51	19.7	14.1	8.2
	CRR	57.02%	68.37%	71.26%	94.22%	90.81%
DoQA	CI	0	0	515	3871	1372
	CQ	0.11	0.12	0.15	0.31	0.2
	TI	0	0	254	1583	1816
	TQ	62	63	23	13.6	7.5
	CRR	52.18%	66.30%	65.83%	87.16%	86.37%

Table 4.1.: A comparison of the five RAG techniques developed, with respect to the evaluation pipeline's metrics. CI = Cost (Indexing, cents), CQ = Cost (Querying, cents), TI = Indexing Time (seconds), TQ = Querying Latency (seconds), CRR = Context Retrieval Recall.

The evaluation pipeline was given five different RAG models implemented in the previous section - Base RAG, HyDE, RAPTOR, GraphRag and LightRAG. The chunking size was standardized across all the methods to 600 tokens to ensure consistency, even though other models may produce better results with different chunking strategies. The large language model used was the OpenAI GPT-4o-mini. The evaluation pipeline was executed separately for the five datasets, using the same allocated resources to enable time of execution comparisons. The costs were monitored using the OpenAI APIs for counting input and output

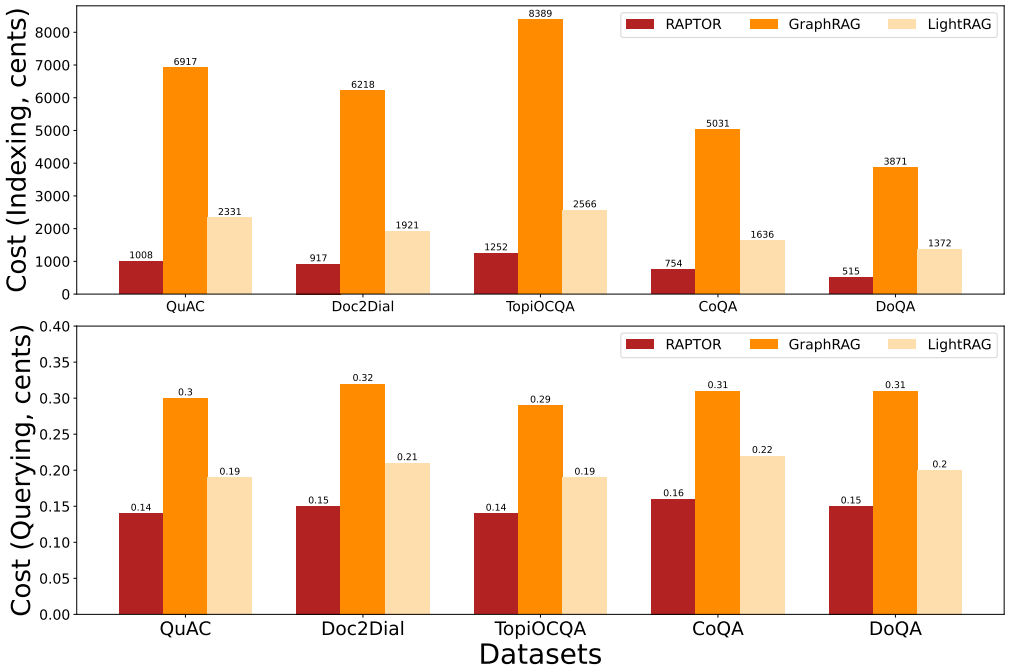

Figure 4.3.: Comparison of costs for all models across datasets. Graph-based models are more expensive to set up as well as do inference with.

tokens consumed, and by referencing the developer API pricing provided on the OpenAI webpage.

Table 4.1 shows the results of our basic experiments using the evaluation pipeline below.

4.4.1. Costs: Graph-Based Models Cost More!

Figure 4.3 shows a comparison of the indexing and querying costs in our selected RAG systems. We elected to leave out the Base RAG and HyDE costs from these depictions because they cost very little to set up and would not be visible in comparison to the costs of the more complex models. The only cost involved in setting up a Base RAG model is the creation of the vector embedding store, which is a process that is shared by all methods at some point. HyDE makes one additional API call to convert the user query into a hypothetical document, so it costs next to nothing as well.

Among the higher-complexity models, RAPTOR consumes the least money to create its index, which makes sense since the technique simply clusters the existing document embeddings based on cosine similarity and summarizes these clusters over multiple levels of abstraction. This is relatively inexpensive as compared to the entity-relationship extraction, clustering, summarization of clusters and community extraction steps that are seen in GraphRAG - all of which require multiple calls to an LLM and hence incur costs. LightRAG is designed to

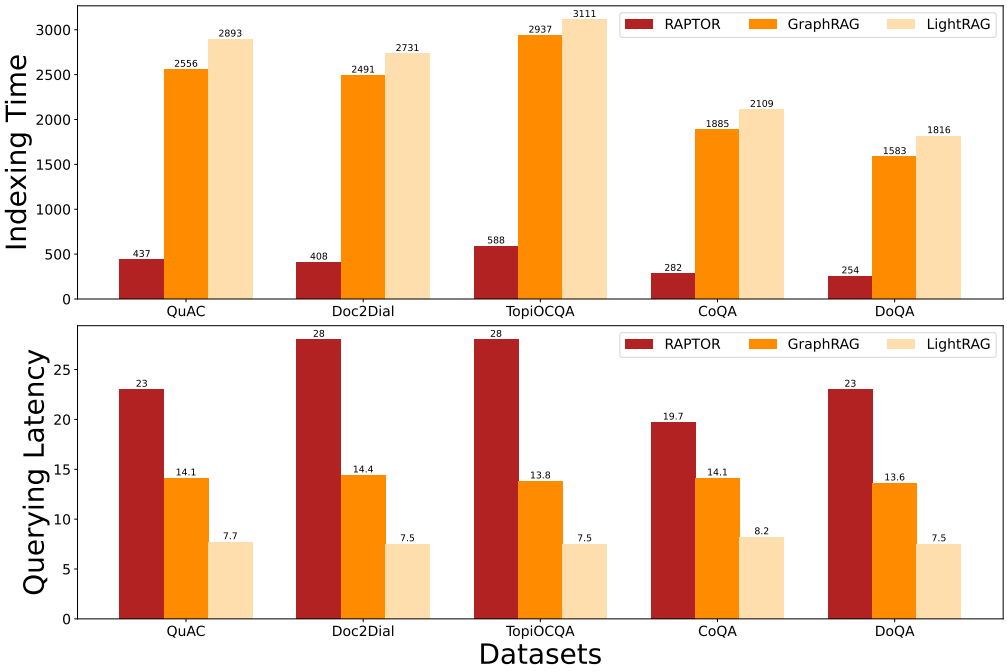

Figure 4.4.: Comparison of times taken (in seconds) for all models across datasets. The increased cost of graph-based models are traded off by the significantly reduced time taken for inference.

be lighter on the wallet as compared to GraphRAG, as community extraction and summarization are not a part of the algorithm. The model instead consists solely of the entity-relationship extraction, LLM profiling of these entities and relationships, and removal of duplicate entities. This cost saving is observed during inference as well, because while GraphRAG traverses the knowledge graph and peruses the communities and their summaries created to find a detailed explanation, LightRAG is able to harness a combination of very local graph-based traversal and vector embedding matching to localize the search space in the knowledge graph.

In general, graph-based approaches are clearly more expensive than vector-based approaches, due to the creation of a knowledge graph that supplies the augmented contexts to the retriever. Vector matching requires vastly reduced LLM usage; however, the abstraction introduced can cause loss of semantic understanding and disconnection of factual logic, which reduces the quality of responses generated.

4.4.2. Times: The Cost-Latency Tradeoff

We see a comparison of the indexing times and query latencies across models for different datasets in figure 4.4. Once again, we do not include the Base RAG and HyDE model times for comprehensibility, although the problem is now reversed

at inference time. These models do not have an indexing procedure to spend time on, but the absence of that process compromises their querying latency. It can take more than a minute to generate a response to the user query, which is prohibitively large in any single-user setting.

As a hybrid between vector-based and graph-based RAG approaches, RAPTOR creates its bottom-up tree-shaped index much faster than either of the pure graph-based approaches. The vector embedding-based clustering and simple summarization is a lightweight approach as compared to other complex models, but the approach has clear drawbacks when we look at the inference latency. Since the model looks at high-level summaries and dives deeper into the tree-shaped index to retrieve the original contexts as candidates for retrieval, we end up looking in a large portion of the document store. Since this implementation was also completely from scratch and without any traversal optimizations introduced, it is possible that inefficient code is a factor in the observed latency.

The graph-based models show the most encouraging results in this domain. While the indexing times scale in magnitude with the indexing costs and are highly comparable to each other, the inference latencies are the best among the group of selected models. It is slightly surprising to see LightRAG consistently take marginally more time to create the knowledge graph, which hints at the LLM profiling and duplicate removal processes to be more time-consuming. GraphRAG lags behind in query latency, which is only to be expected due to the vastly increased number of API calls made to the GPT-4o-mini model as detailed in the previous subsection.

Overall, the results of these two metrics hint at a tradeoff between the cost of setting up the RAG model and the inference time of that model. The initial time and finances invested help create an efficient data structure that can be leveraged for improved responsiveness. The comparison between GraphRAG and LightRAG also indicates that there is room for innovation beyond the cost-latency tradeoff, which can yield more suitable candidates without burning a hole in the organization's budget.

4.4.3. Context Retrieval Recall

The final metric we analyze is the Context Retrieval recall score, which is the average occurrence of the expected true context among the set of best-k retrieved contexts. We use $k = 3$ here, which is a standard number of contexts used for augmented generation using simple RAG models. Figure 4.5 illustrates the difference in CRR across all the models.

Predictably, the vector-based RAG models perform poorly under this metric. Vector-based models lose semantic relationships within data as soon as the contextual documents are converted to vector embeddings without any pre-processing. In the case of Base RAG, the CRR score indicates an alternating hit-and-miss tendency for most short-dialog datasets, and even poorer performance for long-dialog datasets such as Doc2Dial. The retention of meaning

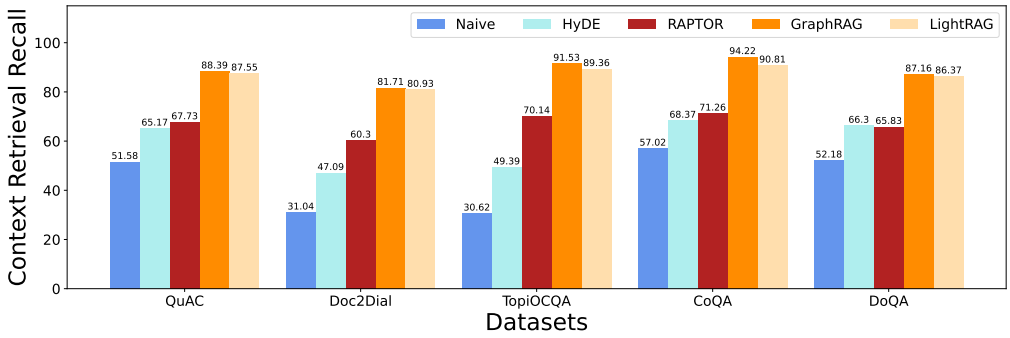

Figure 4.5.: Comparison of Context Retrieval Recall metric for all models across datasets. Graph-based models are better placed to generate higher-quality responses with a firm grasp on correct context relevancy.

within the conversation fades with passing turns, which is especially enhanced in these two datasets due to their follow-up queries being almost impossible to answer correctly without the correct underlying context.

HyDE shows the first significant jump in CRR values among our values. It performs significantly better because of its innate technique of converting the user query into a hypothetical document from the understanding of a pre-trained vanilla LLM. We thus get to see significant gains in CRR over datasets that are grounded on Wikipedia articles and factoids, such as QuAC and TopiOCQA. However, these improvements are not uniform and become less significant for brand new contexts as seen in the CoQA dataset.

RAPTOR serves as the bridge between vector-based and bridge-based models in all senses. The CRR scores reflect an increased capacity in semantic understanding of the documents, which is obtained through the higher-level summarizations while indexing. The capacity is not nearly as much as more complex models because the retrieval process still compares the vector embeddings of these documents and summarizations with that of the query, which can abstract a lot of important information away from the retriever. We see that RAPTOR performs significantly better than HyDE in situations where longer, known contexts generate the conversations and thus the summarization process can grasp the core of the matter. It is actually at par with HyDE in the use cases where shorter text passages and human-created FAQs are used to generate dialogs, and the candidate context documents refer to completely different ideas at a higher frequency, which makes higher-level summarizations more redundant.

The second, and more significant, jump is seen when moving to graph-based models. Entity-relationship extraction preserves semantic relationships between the subjects of the context documents, hence allowing the working knowledge of how one object may be related to another in the real world, as opposed to verbal similarity, is obtained by the RAG system. GraphRAG shows the best CRR scores for our evaluation pipeline, which is a key focus for the model through its extensive knowledge graph that is abstracted upto multiple levels and

communities. The retrieval process ensures that these communities are looked into, so that the neighbourhood of the user query is recognized as accurately as possible in the knowledge graph.

LightRAG performs almost as admirably as GraphRAG in all portions of the pipeline. The knowledge graph created by this technique is less abstracted and relies on accurate entity and relationship matching from the user query to find the right neighbourhood for context retrieval. We see that most of the queries presented to the models are quite related to each other, so the conversation history generally positively enforces the neighbourhood search and mitigates the absence of communities and higher-level summarizations. It would be interesting to observe the presence of any difference in conversations where the topic jumps wildly from turn to turn.

4.4.4. Unanswerable Scenario Analysis

RAG Model	TP	FP	TN	FN	Precision	Recall
BaseRAG	1033	739	761	467	58.16%	68.87%
HyDE	801	583	917	699	57.87%	53.4%
RAPTOR	728	521	979	772	58.29%	48.53%
GraphRAG	917	264	1236	583	77.64%	61.13%
LightRAG	818	323	1177	682	71.69%	54.5%

Table 4.2.: The unanswerable scenario evaluation over 1500 unanswerable turns and 1500 answerable turns. True positives indicate marking an unanswerable query as such.

We move on to the unanswerable scenario evaluation, which is performed by taking an equal number of answerable and unanswerable queries at random from the QuAC and DoQA datasets to avoid the class imbalance problem. As per the ablation studies performed on a similar analysis - albeit for regular LLMs without any RAG complements - by [LPR+24], we take 1.5 thousand unanswerable turns from these datasets, complemented by an equal number of answerable scenarios. The model is prompted to return a staple non-answer in case the query cannot be answered from the provided contexts.

Table 4.2 displays the results of the experiment, to illustrate some interesting results. The Base RAG model seems quite happy to adhere to the unanswerable prompt and identify cases where the answer cannot be found - in fact, it records the highest number of true positive cases found. This may stem from the fact that the retrieval process is inefficient and does not preserve any identifiable elements from the query or contexts. Hence, it tends to return a non-answer result more often than not, which results in a higher recall score but punishes the precision of the system.

HyDE and RAPTOR, on the other hand, show lower recall scores while maintaining similar levels of precision. Both query enhancement and higher-level summarizations introduce extra information that is not necessarily provided by the relevant contexts. This may cause the generation mechanism to hallucinate an answer where an answer should not exist, which indicates an existing bias within these models to give an answer over being confident about their answers.

The graph-based models start to correct the trend, receiving the highest precision scores that indicate the strength of their knowledge bases to identify when an answerable query is presented to them. The recall scores seem low as compared to the Base RAG, but those can be attributed to the anomalous results of Base RAG as compared to an inherent defect of the complex systems. However, it is important to remember that graph-based models do extract entities and relationships from the user query and attempt to match those to the ones present in their knowledge bases, which may occasionally return more false negatives than expected. This would explain why LightRAG performs noticeably worse in the recall metric, since it is highly dependent on finding the right entities and relationships to determine a region of contextual retrieval.

A more mathematically consistent and sophisticated analysis is warranted to derive more actionable results in this scenario. However, it is highly meaningful to consider which metric is to be given greater priority for our use case. In our view, it is much more problematic for the adaptive tutor to hallucinate than to be underconfident, i.e. it is better for us that the model incorrectly says it doesn't know an answer as compared to incorrectly claiming to know an answer and then providing a self-generated response. In more formal terms, if we select our null hypothesis to be that *"the user query is unanswerable"*, we would want to use models that exhibit a low Type-I error (rejecting the null hypothesis when it is actually true). This makes the recall score more important for us to control for our use case.

5. Future Work and Conclusion

The results derived in the previous section give some valuable insight into the class of RAG models that may be great candidates for the virtual tutor.

- Graph-based RAG systems show consistently better performance across all the selected metrics than vector-based systems. The ability of the system to piece information together in an interconnected fashion is highly conducive to improving contextual retrieval, and the gains are significant enough to justify the low-frequency costs incurred in setting up an optimized data structure that allows efficient traversal and preserves semantic meaning.
- It is also noticeable that certain qualities of vector-based systems are still desirable, which include simplified similarity matching, lightweight architecture and a reduced dependence on external LLMs. Models such as LightRAG that attempt to combine the knowledge graph with intelligent adaptive retrieval techniques that carefully perform vector-based matching on simpler elements instead of entire documents are more likely to produce better results using this evaluation pipeline.
- HyDE exhibited the benefits of query rewriting and enhancement, which can be combined with graph-based systems to further improve contextual retrieval quality. [MGH+23] uses the Rewrite-Retrieve-Read technique as used in HyDE, along with the addition of a feedback loop from the response generation LLM via reinforcement learning instead of static query rewriting. RQ-RAG [CXY+24] introduces the concepts of query decomposition and query disambiguation, in order to break the query down into atomic elements that are simpler to use for context searches. Another interesting idea has been presented by HyPE [VVT25], which generates hypothetical prompts/questions instead of hypothetical answers for each document present in the context database, allowing for enhanced query matching directly at inference time, without needing to generate any hypothetical documents at runtime.
- The paucity of time allowed for a limited number of candidate models to be implemented and evaluated through the pipeline. One of the most interesting candidates yet to be tested is the Self-RAG [AWW+23], which employs an actor-critic model to inculcate reflection tokens into the LLM's vocabulary. The model learns to predict these reflection tokens when faced with user queries, and these reflection tokens dictate whether external information is needed to answer the queries, thus saving on unnecessary retrieval calls and cutting down on runtime latency. Fast GraphRAG [Cir24]

build upon the classic GraphRAG architecture, but uses the Personalized PageRank algorithm to make context retrieval even more efficient.

- Our evaluation pipeline uses five datasets that are not part of any specific RAG benchmark for existing RAG models. It may be worthwhile to explore other question-answering datasets that can be adapted as done in this work, that test the capabilities of the RAG models in differing ways. A promising candidate is KILT [PPF+21], a unified benchmark containing 11 datasets created specifically for question-answering tasks that require external knowledge, with Wikipedia as the base knowledge source.

The aim of this thesis is to introduce the concept of retrieval-augmented generation into an educational domain, that brings certain restrictions and views pertinent solely to this field. In a broader sense, we focused on domain-specific RAG functionality evaluation since most RAG benchmarks that we found in existing literature either catered to the response generating LLMs themselves, or were unable to encapsulate the aforementioned restrictions of entering a particular domain. While we have approached the RAG evaluation pipeline from the lens of virtual adaptive tutor systems, the generality of our approach should make it straightforward to adapt this solution to any narrow use case that requires external referencing of data sources in a non-parametrized fashion.

The limitations of our approach include the absence of a dedicated citation engine in our model implementations, that leave us dependent on the innate functionality of the RAG system to generate the citations we care about. Our evaluation pipeline currently does not perform response generation evaluation due to an unclear metric of measurement. This should be the first next step in advancing this work.

Bibliography

[ADS+22] Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. Topiocqa: Open-domain conversational question answering with topic switching, 2022.

[AI23] Arize AI. Understanding and applying f1 score: Ai evaluation essentials with hands-on coding example, 2023.

[AWW+23] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection, 2023.

[BSB25] Lorenz Brehme, Thomas Ströhle, and Ruth Breu. Can llms be trusted for evaluating rag systems? a survey of methods and datasets, 2025.

[Cc22a] Harrison Chase and contributors. Langchain: Building applications with language models, 2022. Accessed: 2025-05-16.

[Cc22b] Harrison Chase and contributors. Llamaindex: A data framework for llm applications, 2022. Accessed: 2025-05-16.

[CHI+18] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac : Question answering in context, 2018.

[Cir24] Circlemind AI. Fast graphrag: Rag that intelligently adapts to your use case, data, and queries. `https://github.com/circlemind-ai/fast-graphrag`, 2024.

[CMZ+24] Yiruo Cheng, Kelong Mao, Ziliang Zhao, Guanting Dong, Hongjin Qian, Yongkang Wu, Tetsuya Sakai, Ji-Rong Wen, and Zhicheng Dou. Coral: Benchmarking multi-turn conversational retrieval-augmentation generation, 2024.

[COS+20] Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Deriu, Mark Cieliebak, and Eneko Agirre. Doqa – accessing domain-specific faqs via conversational qa, 2020.

[CXY+24] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. Rq-rag: Learning to refine queries for retrieval augmented generation, 2024.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[DGD+25] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025.

[EJEAS25] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation, 2025.

[ETC+25] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization, 2025.

[FBS25] Robert Friel, Masha Belyi, and Atindriyo Sanyal. Ragbench: Explainable benchmark for retrieval-augmented generation systems, 2025.

[FWG+20] Song Feng, Hui Wan, Chulaka Gunasekara, Siva Sankalp Patel, Sachindra Joshi, and Luis A. Lastras. doc2dial: A goal-oriented document-grounded dialogue dataset, 2020.

[GLT+20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

[GMLC22] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels, 2022.

[GXG+24] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[GXY+25] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation, 2025.

[IG21] Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering, 2021.

[KPX+23] Pooya Khorram, Jonathan Pilault, Yujia Xie, Thibault Févry, Xinyun Li, Sanjay Krishnan, Andrew McCallum, and Quoc V Le. Raptor: Recursive abstractive processing for tree-oriented retrieval. *arXiv preprint arXiv:2310.02240*, 2023.

[LPP+21] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[LPR+24] Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Surpassing gpt-4 on conversational qa and rag, 2024.

[MGH+23] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting for retrieval-augmented large language models, 2023.

[Ope24] OpenAI. New embedding models and api updates, 2024. Accessed: 2025-05-16.

[PPF+21] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544. Association for Computational Linguistics, 2021.

[RCM19] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge, 2019.

[SAS24] Rickard Stureborg, Dimitris Alikaniotis, and Yoshi Suhara. Large language models are inconsistent and biased evaluators, 2024.

[SFKPZ24] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. Ares: An automated evaluation framework for retrieval-augmented generation systems, 2024.

[SPC+21] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation, 2021.

[VVT25] Domen Vake, Jernej Vičič, and Aleksandar Tošić. Bridging the question-answer gap in retrieval-augmented generation: Hypothetical prompt embeddings, 2025. SSRN preprint.

[WWG+24] Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. Searching for best practices in retrieval-augmented generation, 2024.

[WYX+22] Jun Wang, Xiang Yu, Yujian Xie, Zhifeng Zhou, Shiyuan Ma, Jun Pan, Yubin Guo, Zhe Wang, Wenxuan Xie, Shaoyu Song, et al. Milvus: A purpose-built vector data management system. *arXiv preprint arXiv:2201.10062*, 2022.

[YSX+24] Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Daniel Gui, Ziran Will Jiang, Ziyu Jiang, Lingkun Kong, Brian Moran, Jiaqi Wang, Yifan Ethan Xu, An Yan, Chenyu Yang, Eting Yuan, Hanwen Zha, Nan Tang, Lei Chen, Nicolas Scheffer, Yue Liu, Nirav Shah, Rakesh Wanga, Anuj Kumar, Wen tau Yih, and Xin Luna Dong. Crag – comprehensive rag benchmark, 2024.

[ZLC+25] Zijie Zhong, Hanwen Liu, Xiaoya Cui, Xiaofan Zhang, and Zengchang Qin. Mix-of-granularity: Optimize the chunking granularity for retrieval-augmented generation, 2025.

[ZXL+23] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. Retrieve anything to augment large language models, 2023.

[ZZY+24] Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey, 2024.

A. Appendix: Evaluation Pipeline

A.1. Prompt for Unanswerable Scenario

We provided the following prompt as an instruction to the response generation models in the unanswerable scenario:

- You are a fact-checker. Provided the following query and set of contexts, you must determine whether the query is answerable based solely on the set of contexts' veracity. Your evaluation must decide whether the query can be answered by directly referencing the contexts, without adding any more information. Provide a response while adhering strictly to this format: [["1"]] if the contexts are sufficient for answering, [["0"]] if the contexts are insufficient for answering. Do not provide any further outputs or characters in your response.

A.2. MS GraphRAG Context Retrieval Recall Score Calculation

It has proven notoriously difficult to extract the underlying original contexts that MS GraphRAG references in its responses, and the citations always return community reports unless restricted. Hence, we restrict the model to only give responses that come from the 'Source' citations, where the underlying original context can be retrieved. Since our evaluation pipeline uses very little global context, community reports have been inconsequential in generating responses and restricting them did not change the quality of responses by a significant degree, for human eyes.