Exploring RAG Solutions to Reduce Hallucinations in LLMs

Samar AboulEla[1], Paria Zabihitari[1], Nourhan Ibrahim[1,2], Majid Afshar[1,3], Rasha Kashef[1]
[1] *Electrical, Computer, and Biomedical Engineering*
Toronto Metropolitan University, Toronto ON M5B 2K3, Canada
[2] Faculty of Engineering, Alexandria University, Alexandria, Egypt
[3] Department of Electronics and Computer Engineering Technology, Indiana State University, Terre Haute, USA
{samar.g.aboulela@torontomu.ca, pzabihi@torontomu.ca, nourhan.ibrahim@torontomu.ca, Majid.AfsharNoghondari@indstate.edu
, rkashef@torontomu.ca}

2025 IEEE International systems Conference (SysCon) | 979-8-3315-0818-0/25/$31.00 ©2025 IEEE | DOI: 10.1109/SYSCON64521.2025.11014810

***Abstract*—Large Language Models (LLMs) often face challenges in generating accurate and reliable information, particularly in knowledge-intensive tasks. This limitation, referred to as hallucination, occurs when models produce content that is incorrect, irrelevant, or unsupported by evidence. Retrieval Augmented Generation (RAG) solutions provide a promising approach by integrating relevant external knowledge, enabling models to generate factually grounded responses. This study evaluates the performance of a base LLM model, a fine-tuned DistilBERT model, and two RAG architectures, Naïve RAG and Graph RAG, to study their impact on reducing hallucinations and enhancing contextual understanding. Using subsets of HaluEval, Squad-V2, and TriviaQA benchmark datasets, the base model achieved accuracies of 10.18%, 12.67%, and 5.46% respectively; Naive RAG resulted in 44.56%, 19.04%, and 35.32% accuracies; while the fine-tuned LLM model's accuracies were 72.5%, 72.31%, and 88.7% respectively. Graph RAG resulted in 8.85% and 15.12% accuracies using Squad-V2 and TriviaQA, respectively. Our findings show that while fine-tuned LLMs outperform baseline models, incorporating RAG solutions did not result in significant performance improvements, suggesting that the incorporation of external knowledge may not always align with the needs of the task. Experiments demonstrate that Graph RAG handles complex queries by leveraging relationships within structured knowledge graphs. Data organized as a knowledge graph may enable Graph RAG solutions reach their full potential by utilizing their capacity to efficiently retrieve contextually relevant information. Although computing complexity remains a restriction, this study shows that RAG topologies might not consistently enhance LLM reliability in practical situations.**

***Index Terms*—LLM, RAG solutions, Hallucination, Naive RAG, Graph RAG.**

I. INTRODUCTION

The evolution of Large Language Models (LLMs) has initiated a transformative shift in artificial intelligence (AI), particularly in natural language processing (NLP). These models excel in understanding, interpreting, and generating human language. Their advanced capabilities are now driving impactful changes across sectors such as healthcare [1] [2], finance [3], and education [4], where they facilitate more sophisticated and effective interactions between humans and machines. However, these models may have deficiencies due to a lack of domain-specific knowledge, up-to-date information in real-time, and knowledge that is not included in LLMs' pre-training corpus. These deficiencies may lead to a process in which the model produces false or erroneous data, which are called hallucinations.

In NLP, hallucinations refer to the generation of content by language models that is factually incorrect, irrelevant, or entirely fabricated. It is a significant issue in natural language generation (NLG) tasks, especially when models produce information unsupported by the input data. Hallucinations are broadly categorized into two types: intrinsic hallucinations, where the model generates incorrect or distorted information based on the input, and extrinsic hallucinations, where the generated content is unrelated to or ignores the input entirely [5].

In order to address this issue, it is essential to inject external knowledge to LLMs. Retrieval Augmented Generation (RAG) [6]–[9] improves the quality and relevance generated output by introducing a retrieval component into the LLM generation process. One of the most prominent advantages of RAG is to dynamically query a large text corpus of factual knowledge related to the domain of the questions and integrate them into the generated response of the LLM model. In addition to enhancing the contextual depth of the responses, this integration guarantees a higher level of factual correctness and specificity. RAG has become a major focus in the NLP sector owing to its remarkable performance and wide range of applications. RAG is particularly valuable in tasks such as machine translation, text summarization, and question answering, where the generated output must closely align with the input or the expected ground truth [10]. Given its versatility and high performance, RAG has emerged as a cornerstone of contemporary AI research, pushing improvements in knowledge-grounded conversational systems and decision-support frameworks [11].

Having all the mentioned improvements that RAG will add to the accuracy of responses and wide applications across various domains, Naive RAG has some drawbacks in real-world scenarios. One of them is discarding the structured relation and interconnection in a textual content that cannot be represented through semantic similarity [12]. RAG systems frequently retrieve redundant or highly similar information from multiple sources, which is then concatenated into a single, extensive prompt for the language model. This approach can exceed the model's context window, causing the "Lost-in-the-Middle" phenomenon, where critical information located in the middle of the prompt is overlooked [13]. As a result, responses may be incomplete or inaccurate. Another major drawback is the absence of a global context. RAG retrieves a restricted amount of documents connected to a particular query, thus it may overlook relevant information included in other, less immediately relevant documents [14]. This selective retrieval can result in knowledge gaps and reduce the overall comprehensiveness of the given response.

Graph RAG [15]–[17] emerges as an innovative solution to address these challenges. Graph RAG retrieves data from a graph database, a structure that stores data as interconnected nodes, triples, paths, or subgraphs. Unlike traditional RAG methods, Graph RAG considers interconnections between pieces of information. Knowledge graphs (KGs) help simplify and summarize large amount of text by organizing it in a structured, interconnected format. Graph RAG captures a greater context by extracting closely connected node clusters (subgraphs), making it perfect for jobs like Query-Focused Summarization, which requires specific information within a large context to effectively answer complicated questions.

This paper explores RAG solutions to mitigate hallucinations in LLMs. By leveraging retrieval mechanisms and structured KGs, we examine their potential to enhance the accuracy and reliability of LLM-generated responses in practical applications. Experiments conducted on the HaluEval, SQuAD 2.0, and TriviaQA datasets show that fine-tuned models achieve the highest Exact Match (EM) scores, with 58.15% on HaluEval, 48.3% on SQuAD 2.0, and 69.45% on TriviaQA. Naive RAG improves performance moderately, achieving 35.20% on HaluEval, 10.15% on SQuAD 2.0, and 26.9% on TriviaQA, while Graph RAG struggles on these datasets with 3.33% EM on SQuAD 2.0 and 9.62% on TriviaQA. In comparison, the base models perform significantly lower, with EM scores of 0.4% on HaluEval, 0.25% on SQuAD 2.0, and 0.2% on TriviaQA, highlighting the importance of fine-tuning and retrieval-based methods. These findings underscore the strengths and limitations of RAG solutions in improving LLM performance across diverse datasets.

The rest of the paper is structured as follows: Section II provides the literature review of related work in the field. Section III describes RAG solutions architectures. Benchmark datasets are described in Section IV. Section V explains the hallucination metrics for RAG solutions. Section VI discusses experimental procedures and results. Finally, Section VII outlines the conclusions drawn from our study and discusses potential directions for future work.

II. Literature Review

To reduce hallucination in LLMs, a lot of research has been done in traditional NLG situations [18], [19]. RAG [8], [20], [21] has been a prominent evolution to decrease hallucinations by using external, authoritative knowledge stores. By adding a retrieval component to the generation process, concentrating on the more pertinent documents to the query, and supplementing them to the prompt as model input, RAG has proved successful in lowering hallucinations and improving precision, recall, and accuracy.

The research in [14] explores two RAG models: RAG-Sequence and RAG-Token. In RAG-Sequence, the model retrieves a set of relevant documents and uses the same document throughout the entire response generation process, while in RAG-Token, different documents are retrieved for each token during generation. Both models retrieve the top K documents, allowing the generator to select content from multiple sources when generating an answer. The generator calculates probabilities over these documents for either the sequence or token and marginalizes the results.

Compared to standard RAG techniques which retrieve and augment the relevant document to the prompt to reduce the hallucination, the paper [22] introduces two novel methods rewrite-retrieve-read and a trainable scheme. The first method prompts the LLM to rewrite user queries for more effective search using a few-shot prompt. However, reliance on a frozen LLM can lead to reasoning errors and irrelevant searches. To address this, the authors propose incorporating a small trainable model, initialized with T5-large, that is first trained on pseudo data and later fine-tuned via reinforcement learning, using feedback from the LLM reader to improve query rewriting and retrieval.

While naive RAG has been successful in reducing hallucination, it faces key challenges in real-world applications. Due to its focus on isolated text, it struggles with capturing structured relational knowledge and tends to generate redundant or excessively long content. In addition, RAG lacks the ability to access global information, limiting its performance in tasks like query-focused summarization. Graph RAG [16] addresses these limitations by retrieving relational knowledge from graph databases, including nodes, triples, paths, or subgraphs. This approach enhances retrieval accuracy, reduces verbosity, and incorporates broader relational insights, offering a more effective solution for complex tasks [23]. For instance, in the paper [24], a Structured-Graph RAG processes the query using an LLM. The LLM translates the user's natural language query into a Cypher query, a specialized query language for graph databases. This query is then used by a smart search tool, which navigates through the graph database to identify relevant nodes and edges, effectively retrieving the most pertinent pieces of information from the constructed KGs.

Several other Graph RAG approaches have been proposed. One framework [16] consists of four key stages: indexing of k-hop ego-graphs, graph retrieval, soft pruning to minimize the influence of irrelevant entities, and generation using the pruned textual subgraphs. The proposed framework efficiently identifies relevant subgraph structures while avoiding the computational complexity of exhaustive subgraph searches, which are NP-hard. Additionally, it introduces a novel prompting technique that enables a transition from textual subgraphs to hierarchical text descriptions while preserving information.

By integrating relational knowledge and maintaining graph structures, GraphRAG overcomes the limitations of traditional RAG methods.

Existing methods that convert graphs into natural language [25], [26] face scalability challenges due to excessive token counts or loss of information. A proposed RAG-based design (G-Retriever) [27] reduces hallucinations by directly retrieving data from graphs, accommodating structures that exceed LLM input limits using a call. G-Retriever addresses this by framing subgraph retrieval as a Prize-Collecting Steiner Tree (PCST) [28] optimization problem. The study also confirms that hallucinations occur in Graph LLMs, as they struggle to recall full graph structures, leading to inaccuracies. In contrast, G-Retriever's direct graph retrieval effectively mitigates these issues.

To evaluate the performance of RAG models in specific domains, [29] introduces DomainRAG, a dataset specifically designed for the context of college enrollment. The dataset includes HTML and text corpora, with sub-datasets for tasks like conversational analysis, structural comprehension, and multi-document interactions. The study finds that RAG models outperform LLMs in expert-level queries but still require improvements in handling conversational context, structural information, and maintaining accuracy for domain-specific knowledge.

Over time, researchers [30]–[32] have examined the use of KGs to mitigate hallucinations in LLMs. Their method involves retrieving data from KGs, enhancing the prompt, and feeding it into the LLMs. However, even with using Graph RAG and KG-based approaches, hallucinations persist in query-focused abstractive summarization over large corpora [15]. Large amounts of text can exceed the context window limits of LLMs, and merely increasing window size may not prevent information from being ”Lost-in-the-Middle” [33]. The authors in [34] focus on the modularity of graphs and the use of community detection algorithms to partition graphs into closely related node communities [35], contrasting with previous research that utilized the structured retrieval capabilities of graph indexes. By generating summaries for these communities, LLMs can provide thorough coverage of the underlying graph index and the associated documents. A map-reduce strategy facilitates query-focused summarization of the entire corpus by independently addressing the query with each community summary in parallel and then integrating the relevant partial responses into a final global answer.

III. RAG Architectures

In this paper, we investigate two different RAG architectures: Naive RAG and Graph RAG. Each architecture represents a distinct approach to integrating external knowledge into LLMs, progressively improving retrieval accuracy, response relevance, and adaptability. In the following, we examine each architecture, detailing its structure, strengths, and limitations in the context of retrieval-augmented generation.

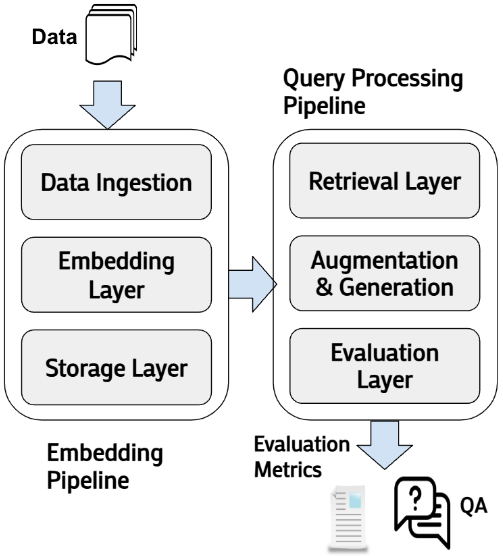

Fig. 1. Naive RAG Architecture

A. Naive RAG

The Naive RAG architecture uses a ”Retrieve-Read” framework to store documents in a vector database. It retrieves top K chunks based on semantic similarity to a query and incorporates them as context in a prompt for the language model to generate a response. This simplicity improves response quality in simple scenarios. However, Naive RAG struggles with precision, as it may retrieve irrelevant or redundant chunks, leading to incoherent or inaccurate outputs. It lacks advanced filtering or refinement processes, making it less suitable for complex queries and limiting adaptability and coherence in cases requiring deeper contextual understanding [14]. Figure 1 shows our Naive RAG architecture.

B. Graph RAG

The Graph RAG architecture is a new approach to document representation and retrieval, based on a graph structure. It organizes document chunks as nodes, with edges representing semantic relationships between them. This dynamic and interconnected process reduces redundancy and irrelevant outputs. Graph RAG retrieves relevant chunks and considers their graph neighbors, providing a richer response. It is effective for handling complex queries, capturing nuanced relationships and providing context-aware responses. It also mitigates the risk of hallucinations by retrieving interconnected information, leading to higher precision and coherence in the output. However, Graph RAG introduces additional computational complexity, requiring advanced algorithms for semantic relationship detection and significant storage and processing resources. Despite these challenges, the benefits of enhanced retrieval precision and contextual relevance make it a powerful solution for sophisticated information retrieval needs [15]. Figure 2 illustrates our Graph RAG architecture.

TABLE I
DATASET SPECIFICATION

Dataset Name	Type	Suitable RAG Solutions	Key Task	Size	Techopath (Hugeness)
HaluEval[1]	Hallucination Detection, Question-Answering	Sequence	Detecting hallucinations in text	35K samples	Medium
SQuAD 2.0[2]	Question-Answering	Sequence & Graph	Answering questions	150K questions	Medium
TriviaQA[3]	Reading Comprehension, Question-Answering	Sequence & Graph	Answering questions	650K questions	Large

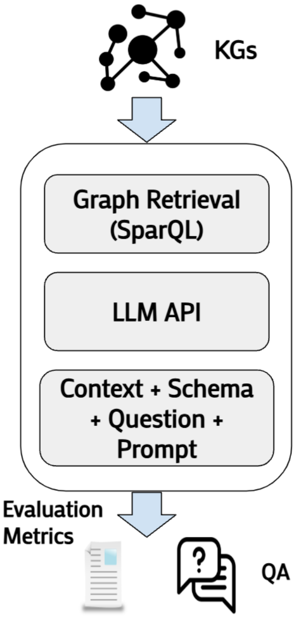

Fig. 2. Graph RAG Architecture

IV. DATASETS

The datasets used in this study are summarized in Table I, highlighting their key characteristics and relevance to the experiments. HaluEval focuses on hallucination detection and question-answering, providing 35,000 samples suitable for sequence-based RAG solutions. SQuAD 2.0, with 150,000 questions, supports both sequence and graph-based solutions for general question-answering tasks. TriviaQA, the largest dataset with 650,000 questions, combines reading comprehension and question-answering, making it ideal for evaluating the scalability of RAG approaches. These datasets offer a diverse range of tasks and data sizes, ensuring comprehensive testing of the proposed solutions.

V. HALLUCINATION METRICS

Hallucination occurs when content in LLMs is erroneous or unreliable, particularly in knowledge-intensive activities such as question answering. RAG attempts to address this by adding external knowledge. Still, if the model exaggerates or misinterprets details, hallucinations may happen. Recent research emphasizes evaluating and decreasing hallucinations in RAG-based models to assure their trustworthiness and dependability [36], [37].

It is necessary to specify metrics for model accuracy and dependability to quantify hallucination reduction in RAG systems. These measures enable technique comparisons and evaluate the efficacy of RAG models. In this paper, we measure our models' performance using widely established measures. Some of the commonly used relevant metrics are explained in the next subsections.

A. Exact Match

Exact Match measures the proportion of generated answers that exactly match the reference answer, requiring a perfect match. This strict metric evaluates the model's ability to generate factually correct responses. A high Exact Match score in RAG models denotes the regular retrieval of trustworthy data.

$$\text{Exact Match} = \frac{|\text{Exact Matches with Reference}|}{|\text{Total Responses}|} \tag{1}$$

Equation 1 calculates Exact Match by dividing the number of exact matches by the total number of generated responses. This metric has been used in studies such as [38] to evaluate question-answering based on the SQuAD-v2.0 dataset.

B. Partial Match

Partial Match assesses the model's ability to generate partially accurate answers, awarding credit for solutions that contain relevant information but do not fully match the reference answer. Partial Match is helpful for evaluating situations in which partial correctness is acceptable.

$$\text{Partial Match} = \frac{|\text{Partially Correct Responses}|}{|\text{Total Responses}|} \tag{2}$$

Equation 2 calculates Partial Match by dividing the number of partially correct responses by the total number of responses. This metric is useful when the LLM may add extra text, along with the expected correct answer.

C. Accuracy

The accuracy metric looks at the total generated output (both correct and incorrect), so it's the number of factually correct outputs over all outputs generated. Accuracy is the proportion of the total generated output that is correct and based on fact. High accuracy suggests that the model consistently produces accurate, fact-based content throughout its responses.

[1] https://github.com/RUCAIBox/HaluEval
[2] https://huggingface.co/datasets/squad_v2
[3] http://nlp.cs.washington.edu/triviaqa/

$$\text{Accuracy} = \frac{|\text{Correct Generated Information}|}{|\text{Total Information Instances}|} \quad (3)$$

Equation 3 calculates accuracy by dividing the total number of factually correct outputs by the total number of information instances generated. This metric has been applied in various studies, such as [37], [39].

D. Precision

Precision metric looks at only the relevant information generated (not including irrelevant parts), so it measures the correct relevant information over the total relevant information generated. Precision in RAG models measures how much of the generated content accurately represents the retrieved data without introducing hallucinated information. A high precision score indicates that the model minimizes unjustified claims by making appropriate use of retrieved content.

$$\text{Precision} = \frac{|\text{Relevant Correct Information}|}{|\text{Total Generated Information}|} \quad (4)$$

Equation 4 calculates precision by dividing the amount of correct information generated by the total generated output. Precision has been used in various studies such as [36] to evaluate factual grounding in RAG models.

E. Recall

Recall measures the percentage of relevant information properly recovered by the model out of all potential relevant information. Recall evaluates how well a RAG model can produce and retrieve all pertinent content depending on the query that was input. A high recall score implies that the model successfully obtains and generates the majority of important, factually correct information, but it does not take into account whether irrelevant information is also generated.

$$\text{Recall} = \frac{|\text{Relevant Correct Information}|}{|\text{Total Relevant Information}|} \quad (5)$$

As given in equation 5, the recall formula divides the correct information used in the output by the total amount of relevant information present in the retrieved content. This metric has been used in recent studies such as in [36].

F. F1-score

By considering both false positives and false negatives, the F1-score is a statistic that helps models balance recall and precision. It is employed in RAG models to evaluate the completeness and accuracy of information gathered and generated. When a model has a high F1-score, it correctly recovers the majority of true content while minimizing the creation of false information.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (6)$$

Equation 6 computes the F1-score as the harmonic mean of precision and recall. F1-score has been used in various studies such as [36], [39]

G. ROUGE-L Score

The ROUGE score [40] captures both accuracy and recall for naturally structured text by calculating the longest word sequence that matches the model's response and the reference response. A high ROUGE-L Score indicates that the generated response closely resembles the reference answer's content and structure.

$$\text{ROUGE-L} = \frac{LCS(\text{Reference}, \text{Generated})}{|\text{Reference}|} \quad (7)$$

Equation 7 computes the ROUGE-L Score using the longest common subsequence (LCS) between the reference and generated answers. ROUGE-L has been widely used for evaluating textual overlap such as in [41]–[43].

H. BLEU Score

The BLEU score [44] compares short word sequences to determine how similar the model's response is to the reference answer. A high BLEU Score indicates that the model's answer closely matches the wording of the reference solution.

$$\text{BLEU} = \text{Brevity Penalty} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (8)$$

Equation 8 computes the BLEU Score using a brevity penalty and precision for n-grams. w_n is the weight applied to the n-gram accuracy score. Weights are often set to $\frac{1}{n}$, where n refers to the number of n-gram sizes utilized. p_n represents the precision rating for the n-gram size. Studies such as [42], [43], [45] have used BLEU for assessing word-level accuracy in QA tasks.

I. Other Metrics

Other metrics that have been used in the literature include: cosine similarity [45], PR_{AUC} (Precision-Recall Area Under Curve) [39], hallucination density [36], error detection rate [37], Factualness Evaluations via Weighting LLMs (FEWL) [46], Laziness Penalty [46], BERTScore [47], and BARTScore [47].

VI. Experiments and Results

We compare four models based on the following machine specifications and experimental setup, ensuring a fair and consistent evaluation across all configurations.

A. Machine Specifications

The experiments were conducted using Python 3.10.12, PyTorch, and the Transformers library for model implementation. The runtime environment was Google Colab, equipped with an Nvidia T4 GPU featuring 16GB of memory to handle computationally demanding tasks. Pre-installed packages in the Colab environment were utilized to streamline the setup process. A random seed of 42 was used to ensure reproducibility, enabling other researchers to replicate the experiments under similar cloud-based conditions.

B. Experimental Setup

The experiments utilized three question-answering (QA) datasets: HaluEval, SQuAD-v2, and TriviaQA. For HaluEval, the QA-data subset containing 10,000 records was used, with top-1 retrieval applied in Naive RAG to accommodate the large context sizes exceeding the 512-token input limit of the DistilBERT model. In SQuAD-v2, 10,000 records were randomly sampled from the full dataset of 130,319 rows, with shorter contexts prioritized through token-size filtering to optimize retrieval compatibility with the RAG framework.

TriviaQA was evaluated in two configurations. A modified SQuAD format was used for baseline models and Naive RAG, providing a cleaned version of question-answer pairs. The unfiltered validation split was utilized for the Graph-RAG implementation, which integrated structured knowledge retrieval from DBpedia. SPARQL queries were constructed to retrieve entities and relationships relevant to the dataset's context.

Across all datasets, 80% of the data was allocated for training (fine-tuning DistilBERT), while 20% was reserved for evaluation. This setup ensured a fair and consistent comparison of model performance across different RAG configurations.

C. Models

We used the DistilBERT-base-uncased model for its balance of performance and efficiency [48]. As a distilled version of BERT, it offers competitive accuracy while being lightweight and suitable for real-time applications. Its open-source availability and hosting on Hugging Face made integration and fine-tuning straightforward.

Our experiments involved four model variants. The baseline model used direct prompting, tested 2,000 samples and compared predicted answers with ground truth. The fine-tuned variant further trained DistilBERT on 8,000 samples and evaluated it on the same 2,000 test samples, demonstrating improved accuracy. The fine-tuned models for each dataset have been saved so that we can use them with RAG model variants. The Naive RAG solution combined dense retrieval using the "sentence-transformers/all-MiniLM-L6-v2" embedding model and ChromaDB as a vector database. It retrieved top-k relevant contexts for each query, which were used by the fine-tuned DistilBERT model for generating responses.

The Graph RAG solution employed DBpedia as the primary knowledge graph, integrating components such as entity detection, relationship extraction, and SPARQL-based retrieval. This system constructed contextual inputs for DistilBERT, enhancing response coherence. Evaluation on subsets of SQuAD and TriviaQA demonstrated its effectiveness due to their alignment with DBpedia. However, the HaluEval dataset could not be tested with Graph RAG, as it lacks structural alignment with DBpedia, highlighting the importance of matching datasets to the knowledge base. Our experiments explore the potential of RAG systems in improving the accuracy and contextual relevance of LLM outputs.

TABLE II
PERFORMANCE OF DIFFERENT MODELS ON HALUEVAL DATASET

Model	**Exact Match**	**Partial Match**	**ROUGE-L Score**	**BLEU Score**	**Running Time**
Base Model (distilbert-base-uncased)	0.4%	9.4%	7.80%	1.44%	295.12
Fine-tuned Base Model (distilbert-base-uncased)	58.15%	72.60%	69.22%	38.56%	25.07
Fine-tuned Base Model + Naive RAG Solution	35.20%	41.75%	44.51%	23.59%	274.19

TABLE III
PERFORMANCE OF DIFFERENT MODELS ON SQUAD 2.0 DATASET

Model	**Exact Match**	**Partial Match**	**ROUGE-L Score**	**BLEU Score**	**Running Time**
Base Model (distilbert-base-uncased)	0.25%	10.55%	9.23%	1.97%	340.60
Fine-tuned Base Model (distilbert-base-uncased)	48.3%	66.15%	65.80%	38.98%	8.08
Fine-tuned Base Model + Naive RAG Solution	10.15%	16.05%	17.23%	9.09%	517.35
Fine-tuned Base Model + Graph RAG Solution	3.33%	4.76%	8.08%	3.66%	5074.02

TABLE IV
PERFORMANCE OF DIFFERENT MODELS ON TRIVIAQA DATASET

Model	**Exact Match**	**Partial Match**	**ROUGE-L Score**	**BLEU Score**	**Running Time**
Base Model (distilbert-base-uncased)	0.2%	6.6%	3.41%	0.56%	720.64
Fine-tuned Base Model (distilbert-base-uncased)	69.45%	82.7%	85.79%	54.83%	27.28
Fine-tuned Base Model + Naive RAG Solution	26.9%	33.85%	35.62%	21.56%	772.91
Fine-tuned Base Model + Graph RAG Solution	9.62%	13.81%	14.27%	7.61%	5930.19

D. Results and Discussion

Tables II, III, and IV describe the outcomes of experiments employing the datasets: HaluEval, SQuAD 2.0, and TriviaQA. Also, figures 3 4, and 5 visualize some of the performance metrics. The performance of the implemented models on these datasets is examined below.

On the HaluEval dataset (Table II), the fine-tuned base model beat the base model by a large margin, achieving 58.15% Exact Match (EM) and 72.60% Partial Match. Using retrieval to enhance performance, the naïve RAG solution outperformed basic prompting with the base model. However, it still fell short of the fine-tuned model, owing to the quality of the recovered contexts and embeddings. The RAG solution's capacity to deliver various or more full contextual information was further constrained due to its reliance on top-1 retrieval. Furthermore, the overhead of retrieval procedures increased the assessment time for the naïve RAG model.

The SQuAD 2.0 dataset performance is highlighted in Table III. The fine-tuned basic model produced an EM of 48.3% and a Partial Match of 66.15%, significantly improving on the base model's performance. Compared to the base model, the Naive RAG solution with top-3 retrieval had better performance for most metrics, but it took more time to run and evaluate. The Graph RAG implementation produced a high

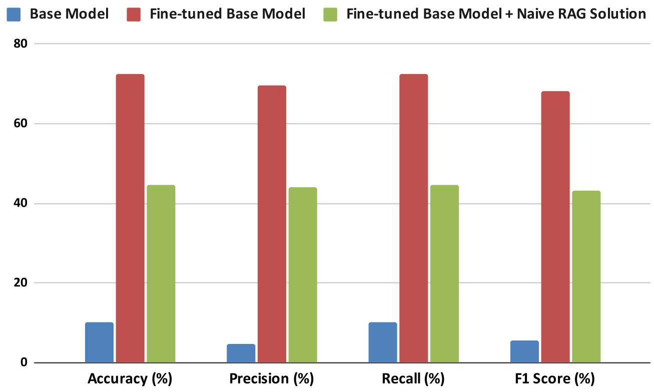

Fig. 3. Performances on HaluEval dataset

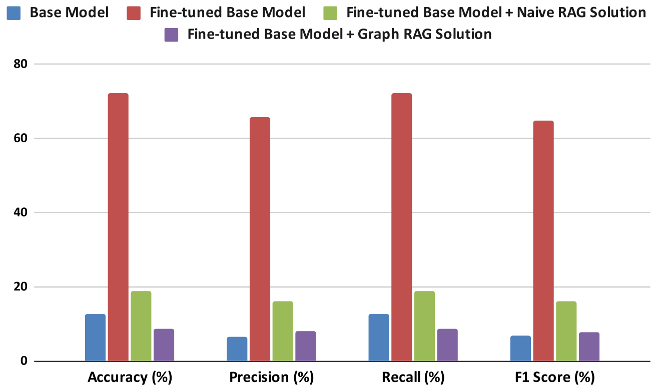

Fig. 4. Performances on SQuAD 2.0 dataset

evaluation time (5074.02 seconds) and a comparatively poor EM (3.33%). This suggests that our sampled subset from the SQuAD dataset may not fully exploit the advantages of graph-based knowledge retrieval.

On the TriviaQA dataset (Table IV), the fine-tuned base model had the maximum performance, with an EM of 69.45% and a Partial Match of 82.7%. Although less effective (26.9% EM), the naïve RAG solution outperformed the basic model (0.2% EM). The Graph RAG solution achieved 9.62% EM after struggling with TriviaQA's many question kinds, despite being able to handle structured knowledge inquiries. The computational expense of graph-based methods was reflected in the much longer retrieval time for Graph RAG.

Overall, the fine-tuned base model continuously beat other methods, proving the efficacy of task-specific fine-tuning. Despite its promise for structured knowledge retrieval, graph RAG performed poorly on datasets not specifically prepared for graph-based inquiries and had greater computing costs.

VII. Conclusion

This study examines the potential of RAG solutions, precisely Naive RAG and Graph RAG, to reduce hallucination in LLMs. According to our results using DistilBERT, open-source LLM, fine-tuned base models consistently outperformed naive prompting methods, while Naive RAG improves retrieval precision and contextual depth. By incorporating structured knowledge from graph databases, Graph RAG provides nuanced and interconnected responses, making it suitable for complex queries. However, its computational overhead and limited dataset compatibility highlight the need for further optimization.

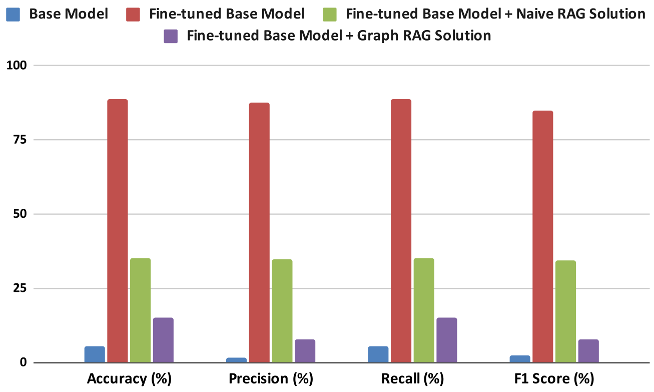

Fig. 5. Performances on TriviaQA dataset

Our findings also emphasize that fully unlocking the potential of Graph RAG solutions necessitates using graph-based knowledge storage. Transitioning from traditional datasets to graph knowledge structures enables more effective utilization of interconnected data, allowing Graph RAG to achieve its full capability in handling complex information needs. Future research should optimize data storage in graph knowledge formats and further refine graph-based retrieval mechanisms to balance accuracy, scalability, and computational efficiency. While RAG designs improve retrieval precision and contextual relevance, our results show that fine-tuned LLMs yield higher overall accuracy. Further study should look into ways to reduce processing cost in structured retrieval while also improving Graph RAG's adaptability across varied domains, giving it a more practical solution for real-world applications.

References

[1] Lei Liu, Xiaoyan Yang, Junchi Lei, Xiaoyang Liu, Yue Shen, Zhiqiang Zhang, Peng Wei, Jinjie Gu, Zhixuan Chu, Zhan Qin, et al. A survey on medical large language models: Technology, application, trustworthiness, and future directions. *arXiv preprint arXiv:2406.03712*, 2024.

[2] Jinqiang Wang, Huansheng Ning, Yi Peng, Qikai Wei, Daniel Tesfai, Wenwei Mao, Tao Zhu, and Runhe Huang. A survey on large language models from general purpose to medical applications: Datasets, methodologies, and evaluations. *arXiv preprint arXiv:2406.10303*, 2024.

[3] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382, 2023.

[4] Aashish Ghimire, James Prather, and John Edwards. Generative ai in education: A study of educators' awareness, sentiments, and influencing factors. *arXiv preprint arXiv:2403.15586*, 2024.

[5] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[6] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

[8] Yucheng Hu and Yuxing Lu. Rag and rau: A survey on retrieval-augmented language model in natural language processing. *arXiv preprint arXiv:2404.19543*, 2024.

[9] Shangyu Wu, Ying Xiong, Yufei Cui, Haolun Wu, Can Chen, Ye Yuan, Lianming Huang, Xue Liu, Tei-Wei Kuo, Nan Guan, et al. Retrieval-augmented generation for natural language processing: A survey. *arXiv preprint arXiv:2407.13193*, 2024.

[10] Vipula Rawte, Amit Sheth, and Amitava Das. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922*, 2023.

[11] Kurt Shuster, Mojtaba Komeili, Leonard Adolphs, Stephen Roller, Arthur Szlam, and Jason Weston. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion. *arXiv preprint arXiv:2203.13224*, 2022.

[12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR, 2020.

[13] Aryaman Arora, Adam Farris, Samopriya Basu, and Suresh Kolichala. Computational historical linguistics and language diversity in south asia. *arXiv preprint arXiv:2203.12524*, 2022.

[14] Patrick Lewis et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

[15] Darren Edge et al. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[16] Yuntong Hu et al. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*, 2024.

[17] Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

[18] Ziwei Ji et al. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[19] Ziyao Zhang et al. Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation. *arXiv preprint arXiv:2409.20550*, 2024.

[20] Wenqi Fan et al. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6491–6501, 2024.

[21] Yunfan Gao et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

[22] Xinbei Ma et al. Query rewriting for retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, 2023.

[23] Boci Peng et al. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*, 2024.

[24] Zahra Sepasdar et al. Enhancing structured-data retrieval with graphrag: Soccer data case study. *arXiv preprint arXiv:2409.17580*, 2024.

[25] Xiaoxin He et al. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907, 2025.

[26] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.

[27] Jianan Zhao et al. Graphtext: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

[28] Ali Ahmadi et al. Prize-collecting steiner tree: A 1.79 approximation. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1641–1652, 2024.

[29] Shuting Wang et al. Domainrag: A chinese benchmark for evaluating domain-specific retrieval-augmented generation. *arXiv preprint arXiv:2406.05654*, 2024.

[30] Xingxuan Li et al. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources. *arXiv preprint arXiv:2305.13269*, 4, 2024.

[31] Jinheon Baek, Alham Fikri Aji, and Amir Saffari. Knowledge-augmented language model prompting for zero-shot knowledge graph question answering. *arXiv preprint arXiv:2306.04136*, 2023.

[32] Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. Chatgpt is not enough: Enhancing large language models with knowledge graphs for fact-aware language modeling. *arXiv preprint arXiv:2306.11489*, 2023.

[33] Nelson F. Liu et al. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[34] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[35] V. A. Traag, L. Waltman, and N. J. Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), 2019.

[36] Yuanhao Wu, Juno Zhu, Siliang Xu, Kashun Shum, Cheng Niu, Randy Zhong, Juntong Song, and Tong Zhang. Ragtruth: A hallucination corpus for developing trustworthy retrieval-augmented language models. *arXiv preprint arXiv:2401.00396*, 2023.

[37] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762, 2024.

[38] Michael Kamfonas and Gabriel Alon. What can secondary predictions tell us? an exploration on question-answering with squad-v2. 0. *arXiv preprint arXiv:2206.14348*, 2022.

[39] Ernesto Quevedo, Jorge Yero, Rachel Koerner, Pablo Rivas, and Tomas Cerny. Detecting hallucinations in large language model generation: A token probability approach. *arXiv preprint arXiv:2405.19648*, 2024.

[40] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[41] HJ Sansford, N Richardson, HP Maretic, and JN Saada. Grapheval: A knowledge-graph based llm hallucination evaluation framework. *arXiv preprint arXiv:2407.10793*, 2024.

[42] Rui Yang et al. Kg-rank: Enhancing large language models for medical qa with knowledge graphs and ranking techniques. *arXiv preprint arXiv:2403.05881*, 2024.

[43] Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Evaluating question answering evaluation. In *Proceedings of the 2nd workshop on machine reading for question answering*, pages 119–124, 2019.

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[45] Jiarui Li, Ye Yuan, and Zehua Zhang. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*, 2024.

[46] Jiaheng Wei, Yuanshun Yao, Jean-Francois Ton, Hongyi Guo, Andrew Estornell, and Yang Liu. Measuring and reducing llm hallucination without gold-standard answers via expertise-weighting. *arXiv preprint arXiv:2402.10412*, 2024.

[47] SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024.

[48] Victor Sanh, L Debut, J Chaumond, and T Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arxiv 2019. *arXiv preprint arXiv:1910.01108*, 2019.