CAGRA: Context-Aware Graph Retrieval Augmentation for Query Summarization

Sharique Pervaiz, Navtegh Singh Gill, Stuti Wadhwa
Department of Computer Science
University of Toronto
{sharique,navtegh,stutiwadhwa,}@cs.toronto.edu

Abstract

Large Language Models (LLMs) often struggle with factual consistency and reasoning across large or unseen knowledge sources. Retrieval-Augmented Generation (RAG) offers a solution by incorporating external documents, but conventional RAG approaches are limited in handling multi-hop queries and maintaining global coherence. In this work, we propose CAGRA, a novel framework that combines Retrieval-Augmented Fine-Tuning (RAFT) [1] with GraphRAG [2], enabling both fine-grained and global understanding through knowledge graphs and community-based summarization. We finetune the Phi-2 model using a parameter-efficient QLoRA strategy on multi-hop questions and integrate it with graph-based summaries derived from hierarchically clustered knowledge graphs. Evaluation on curated multi-hop and global question sets shows that CAGRA consistently outperforms baseline RAG, RAFT, and GraphRAG models across five metrics, including BERTScore, ROUGE-1, and Semantic Similarity. These results highlight CAGRA's effectiveness in producing more contextually grounded and coherent responses, particularly in complex multi-document reasoning tasks. The codes can be found at `https://github.com/shariquep/Context-Aware-Graph-Retrieval-Augmentation-for-Query-Summarization`.

1 Introduction

LLMs have shown remarkable ability to reason over natural languages but their knowledge can be limited to what's in their training dataset. As a result, they often struggle with hallucinations, coherence, and factual consistency, particularly with new knowledge bases, private knowledge bases and domain-specific tasks. RAG has emerged as a powerful technique to overcome these challenges by incorporating external knowledge sources. It works by retrieving a small set of relevant documents from a knowledge base and using it to augment the LLM's generation process.

RAG systems, while powerful, face challenges in effectively handling multi-hop queries and maintaining global context across large document corpora. As they incorporate information from multiple sources, the context provided to the LLM can become extensive, leading to confusion or incoherent responses. This is especially problematic in multi-hop scenarios where each retrieval step adds to the overall context, leading to the ineffective use of previously retrieved knowledge. Knowledge graphs, due to their intrinsic "nodes connected by edges" nature, encode massive heterogeneous and relational information across documents, making them a golden resource for addressing the multi-hop and global query-focused summarization limitations of RAG systems.

When using LLMs for downstream applications, it is common to additionally incorporate new information into the pretrained model either through RAG-based-prompting, or finetuning. In this study, we utilise RAFT, a training recipe which improves the model's ability to answer questions in "open-book" in-domain settings. We also utilise knowledge graphs with the finetuned RAG model in order to make the answers more coherent across multi-hop queries and globally contextual to the document corpora. Specifically, use use GraphRAG, a graph-based approach to question answering

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

that scales with both the generality of user questions and the quantity of source text. Combining the finetuning recipe of RAFT and the query-focused global summarization of GraphRAG, we propose CAGRA, an approach that captures global and local information within the documents through extracted communities from a knowledge graph. Our results show that CAGRA outperforms all other models, and both RAFT and Baseline GraphRAG outperform the baseline RAG model, which indicates that the combination of fine-tuning and knowledge graphs improves answer quality further. Furthermore, compared to baseline RAG and RAFT, the baseline GraphRAG model and CAGRA both show better performance on global questions than non-global questions on most evaluation metrics.

2 Related work

RAG enhances LLMs by retrieving relevant information from an external knowledge base before generating a response, enabling them to handle queries beyond their context window. This transforms LLMs from static knowledge repositories into dynamic, contextually aware agents without retraining the model. It also allows for a more transparent reasoning process, as the retrieved information can be used to explain the LLM's output. However, many RAG models rely on vector search methods that retrieve only top-k passages, which may lead to fragmented, incomplete, or redundant information.

To address these limitations, many advanced techniques were proposed such as Forward-Looking Active Retrieval augmented generation (FLARE) [3] where, predicted sentences that have low confidence tokens are used as input queries to retrieve new information from knowledge base. Retrieval-Augmented Reinforcement Learning[4] uses an RL agent to leverage past experiences for decision-making, reducing reliance on parametric updates. Dense Knowledge Retrieval [5] employs siamese sequence embedding models to retrieve relevant documents. Learning-to-rank generative retrieval (LTRGR) [6] was proposed to enable generative retrieval to learn to rank passages directly.

These techniques still struggle with multi-step reasoning and sensemaking queries, which require global understanding of the dataset. In such cases, knowledge graphs have been used to model the relationships between different pieces of information. Some techniques use subgraphs, elements of the graph, or properties of the graph structure directly in the prompt [7],[8],[9] or as factual grounding for generated outputs [10]. Other techniques [11] use an LLM-based agent to dynamically traverse the graph at query retrieval time. GraphRAG is an approach that extends prior graph-based RAG approaches and enables sensemaking over a large text corpus. It begins by using an LLM to construct a knowledge graph, which is then hierarchically partitioned into nested communities. It recursively creates increasingly global summaries spanning the community hierarchy in a bottom-up fashion, with higher-level summaries recursively incorporating those from lower levels, enabling comprehensive corpus-level insights.

Supervised fine-tuning is widely used to adapt LLMs for tasks like question answering and classification. However, many methods either ignore retrieval during inference or fail to address retrieval imperfections during training. This disconnect limits performance, akin to taking an open-book test without studying or studying without using the book. Conventional fine-tuning approaches resemble studying without referencing the book — they either memorize documents [12] or keep answering practice questions [13]. RAFT addresses this challenge by combining instruction fine-tuning with RAG to enable models to incorporate domain knowledge while improving performance. It accomplishes this by differentiating between relevant and irrelevant documents through a chain-of-thought fine-tuning process. In this study, we utilise GraphRAG's methodology to make answers globally contextual, combined with RAFT-inspired finetuning on a multi-hop reasoning task to better identify relevant information during query retrieval.

3 Formal Description and Experimental Details

The complete algorithm is described in Algorithm 1, and the overall methodology workflow is illustrated in Figure 1.

3.1 RAFT-based Finetuning:

We follow the approach outlined in RAFT by fine-tuning a compact yet powerful causal language model, Microsoft Phi-2 [14] using the Parameter-Efficient Fine-Tuning (PEFT) technique, specifically

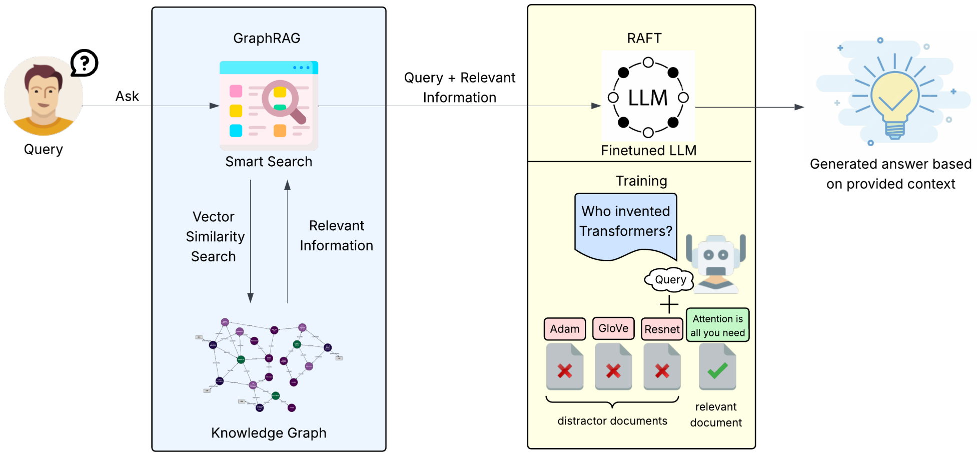


Figure 1: Overview of CAGRA: Combining Fine-Tuned Language Models with Graph-Based Summarization

utilizing the Quantized Low-Rank Adaptation (QLORA) method. This model is loaded from the Hugging Face Hub using a BitsAndBytesConfig that loads it in 4-bit precision and performs computations in half (float16) precision, enabling memory-efficient fine-tuning and making it possible to run Phi-2 on a single GPU. Subsequently, the tokenizer associated with the Phi-2 model is loaded with autoregressive left padding. We also use a separately configured evaluation tokenizer, to ensure consistency in padding and token boundaries.

3.1.1 Data Preprocessing

The dataset used for finetuning is a subset of the HotPotQA dataset (20,000 samples) [15], loaded from Huggingface datasets library with the "distractor" configuration. To align with the RAFT's fine-tuning framework, we configured this subset such that each training example consists of a question, an answer, and a set of passages: 1-2 oracle passages required to answer the question, derived from sentences specifically identified in the supporting facts and 3-4 irrelevant distractor passages, derived from all other paragraphs. Below are the key components of the pipeline:

1. **Prompt Formulation**: A unified instruction-style prompt was crafted for every example as:
   ### Instruct: With the given context, please answer the question in one word.
   Question: <QUESTION>
   Context: <CONTEXT>
   <ANSWER>

2. **Context Truncation Strategy**: Given the token limit constraints of the transformer model, we implemented the following truncation logic:
   - Static Prompt Length Estimation: We first computed the token length of the instruction prompt, question, and answer to determine the space available for the context.
   - Truncation Policy: If relevant context alone exceeded available token length, it was truncated. If it fit within bounds, irrelevant context was appended up to the remaining budget. Final combined context was randomly shuffled at the sentence level to prevent overfitting to context order.
   - Token-level truncation was applied at the end to strictly ensure sequences adhered to the maximum length constraint (dynamically determined from model.config or defaulted to 1024).

This preprocessing approach ensures that: (1) The model learns to perform reasoning over both relevant and distractor documents. (2) The input remains within the model's context length limits. (3) Positional bias is reduced through random shuffling of the passages.

Algorithm 1 CAGRA: RAFT-Finetuned GraphRAG Methodology

```
 1: Input: Pretrained LLM $\mathcal{M}$ (Phi-2), HotPotQA data subset $\mathcal{D}_{qa}$, Wikipedia corpus $\mathcal{D}_{wiki}$, Multi-hop query set $\mathcal{Q}_{\Updownarrow\langle} = \{q_1, ..., q_n\}$, global query set $\mathcal{Q}_{\}} = \{Q_1, ..., Q_n\}$
 2: Output: Finetuned model $\mathcal{M}^*$, Community summaries $\mathcal{S}$, Answers $\mathcal{A}$, Evaluation metrics
 3: // RAFT-Based Finetuning
 4: for each $(q, a, P_{rel}, P_{irr}) \in \mathcal{D}_{qa}$ do
 5:     $P \leftarrow P_{rel} \cup P_{irr}$
 6:     Construct prompt with instruction-style template and shuffle sentences in $P$
 7:     if prompt exceeds max token length then
 8:         Truncate $P_{rel}$, append $P_{irr}$ to fit budget
 9:     end if
10:     Store prompt
11: end for
12: Configure QLoRA: Rank=32, Alpha=32, Dropout=0.05, Target={q_proj, k_proj, v_proj, dense}
13: Train with PagedAdamW8bit, batch=1, grad_accum=8, max_steps=500, lr=$2e{-}4$
14: Store resulting model $\mathcal{M}^*$
15: // GraphRAG Knowledge Graph and Summarization
16: for each document $d \in \mathcal{D}_{wiki}$ do
17:     Split $d$ into overlapping chunks
18:     for each chunk do
19:         Extract entities and relations using LLM; Add nodes and edges to knowledge graph $\mathcal{G}$
20:     end for
21: end for
22: Merge duplicates and upsert into $\mathcal{G}$
23: Partition $\mathcal{G}$ using Leiden to obtain hierarchical communities $\mathcal{C}$
24: for each level in $\mathcal{C}$ do
25:     for each community $c$ at this level do
26:         if $c$ has no subcommunities then
27:             Prompt LLM with entities and relations
28:         else
29:             Concatenate sub-community summaries
30:         end if
31:         Generate JSON + text summary, store it in $\mathcal{S}$
32:     end for
33: end for
34: // Query-Focused Summarization
35: for each query $q \in \mathcal{Q}$ do
36:     Map Phase:
37:     for each batch of summaries do
38:         Extract key points and scores via LLM
39:     end for
40:     Reduce Phase:
41:     Combine key points across batches
42:     Prompt LLM to synthesize final answer
43:     Store answer in $\mathcal{A}$
44: end for
45: // Inference and Evaluation
46: Embed all document chunks using Sentence-BERT and index using FAISS
47: for each $q \in \mathcal{Q}_{\Updownarrow\langle}$ do
48:     if Baseline/Finetuned RAG then
49:         Retrieve top-3 chunks from $\mathcal{D}_{wiki}$ in FAISS
50:     else
51:         Retrieve top-1 GraphRAG community summary
52:     end if
53:     Construct prompt and generate answer using $\mathcal{M}$ or $\mathcal{M}^*$
54:     Store answer for multi-hop evaluation
55: end for
56: for each $q \in \mathcal{Q}_{\}}$ do
57:     Repeat steps 48-54
58: end for
59: Compute evaluation metrics: BERTScore, BLEU, ROUGE-1, Semantic Similarity, Token-level F1
```

3.1.2 PEFT using QLoRA:

To adapt the pre-trained LLM to our task without incurring the computational cost of full fine-tuning, we employed QLoRA using Hugging Face's PEFT library. This method introduces trainable low-rank decomposition matrices into specific layers of the model, enabling efficient fine-tuning with significantly fewer parameters. The configuration was as follows:

- Rank (r): 32, LoRA Alpha: 32, LoRA Dropout: 0.05
- Target Modules: q_proj, k_proj, v_proj, and dense, corresponding to key components of the transformer attention mechanism
- Bias Adaptation: Disabled (as per LoRA best practices on pre-trained LLMs)
- Task Type: CAUSAL_LM (causal language modeling)

3.1.3 Training Configuration:

We used the HuggingFace Trainer API, subclassed with a custom trainer to provide step-level logging during training. The model was optimized with 8-bit Paged AdamW, an efficient optimizer designed for large-scale model training. The key training hyperparameters and strategies were:

- Batch Size: per_device_train_batch_size=1 with gradient_accumulation_steps=8, resulting in an effective batch size of 8
- Max Steps: 500, ensuring quick convergence and efficient early-stage experimentation
- Number of epochs: 1
- Learning Rate: 2e-4, chosen relatively high as per LoRA's low-rank update strategy
- Gradient Checkpointing: Enabled to reduce memory footprint
- Model Cache: Disabled to support checkpointing during training
- Checkpointing and Logging: Every 25 steps, allowing fine-grained tracking and recovery
- Evaluation Strategy: Disabled during initial training to reduce computational overhead

This parameter-efficient setup allowed us to train on a consumer-grade NVIDIA RTX A4500 GPU while retaining strong model performance on our specific task.

3.2 GraphRAG based query-focused global text summarization

For this part, we utilize Nano-GraphRAG, a smaller, faster, and cleaner implementation of GraphRAG, which retains the same core functionality and results as obtained by GraphRAG.

3.2.1 Knowledge Graph Construction via LLM Extraction

GraphRAG constructs a knowledge graph by splitting the input documents into manageable chunks, for each of which, an LLM identifies entities and relationships in a structured list format. These are then parsed into a graph data structure where each entity becomes a node (with attributes like entity type and description), and each relationship becomes an edge (with attributes like relationship description and a numeric weight indicating relationship strength). Duplicate entities and relationships across chunks are merged or updated. The result is a consolidated knowledge graph capturing the key entities in the corpus and the relations among them, as inferred by the LLM.

3.2.2 Graph Partitioning into Communities

GraphRAG hierarchically partitions the knowledge graph into nested communities using methods like Louvain or Leiden [16] (the default) to find clusters of closely related entities. This results in a set of community assignments for each node at multiple levels of granularity. Internally, each node's record is annotated with one or more cluster identifiers corresponding to the community it belongs to at each level. Then, a community schema is derived, mapping each community ID to its member nodes, internal edges, and hierarchical level. Higher-level communities are composed of several lower-level sub-communities, structurally partitioning the entire knowledge graph into thematically grouped subsets of entities.

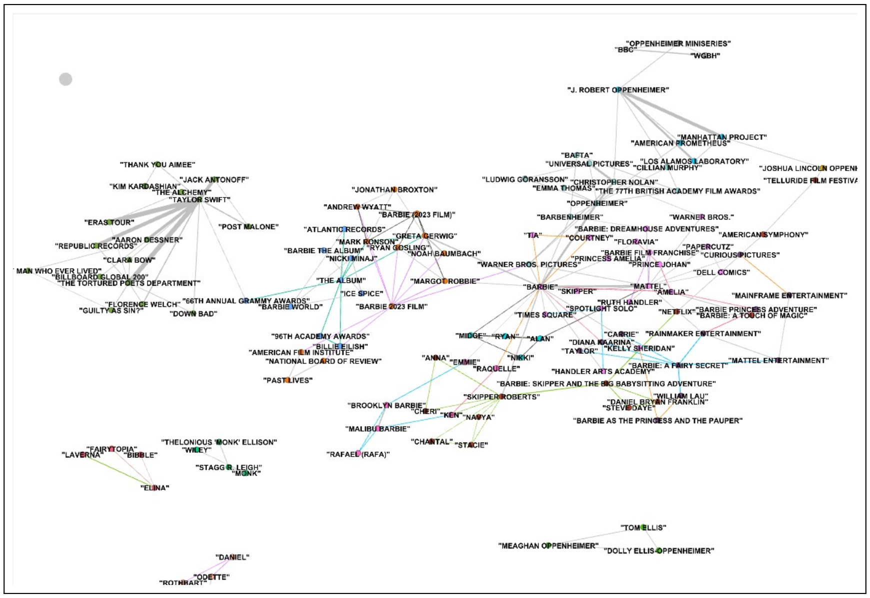


Figure 2: GPT4o Knowledge Graph, color-coded by community. The graph is constructed based on Wikipedia articles on Oppenheimer, The Barbie Movie (2023), and The Tortured Poets Department.

3.2.3 Hierarchical Community Summarization

In this step, GraphRAG summarizes each community's content using the LLM, proceeding in a bottom-up hierarchical fashion. For the lowest-level communities, the system composes a prompt to the LLM comprising the community's entities and their relations. This template instructs the model to produce a comprehensive report of the community in a structured JSON output containing a title, a summary, an "impact severity" rating (a numeric importance score), a brief explanation of that rating, and a set of detailed findings. This JSON constitutes the community's summary report. For higher-level communities, which consist of multiple sub-communities, GraphRAG assembles an input that incorporates the reports of its constituent sub-communities – for example, by concatenating the sub-community summaries as context. In this way, summaries are generated recursively from fine-grained to coarse community levels, resulting in a nested set of reports and a final top-level summary of the entire corpus. Each community report is stored in a key-value store and includes both machine-readable JSON and a human-readable text version.

3.2.4 Query-Focused Summarization via Map-Reduce

Finally, GraphRAG can answer user queries by drawing on the precomputed community summaries. The process operates in two phases: map (partial answer generation) and reduce (aggregation).

1. **Map Phase:** The system utilizes all communities up to a specified level, constrained by token limits. If needed, communities are split into batches, and each batch is processed with a query-specific prompt that instructs the LLM to extract key points along with descriptions and importance scores in JSON format. Acting as an "analyst" for that subset of data, the model produces a set of answer fragments and their relevance scores. This is done in parallel for all batches of communities, yielding multiple intermediate JSON outputs.

2. **Reduce Phase:** In this stage, GraphRAG aggregates the key points from all community batches, ranking or filtering them by importance. These aggregated points are then passed into a second prompt that guides the LLM to generate a coherent, comprehensive answer by integrating perspectives from multiple "analysts" while eliminating redundancies.

This map-reduce approach enables scalable processing of large corpora by generating partial answers from community summary batches independently and then combining them into a single, globally informed response. The final output is a query-focused summary that captures high-level insights grounded in the LLM-generated knowledge graph and community narratives.

3.3 Evaluation

3.3.1 Dataset for Multi-hop Query Evaluation

To evaluate multi-hop factual accuracy, we used a document corpus containing 74 articles from the Wikipedia 2024 corpus, published after the language model's training cutoff date. These articles contain information across three domains—Taylor Swift's album The Tortured Poets Department, Greta Gerwig's film Barbie and Christopher Nolan's film Oppenheimer. From these articles, we manually curated a set of 59 question-answer triplets such that each triplet includes:

- A natural language question requiring reasoning over multiple context segments.
- A corresponding ground-truth answer.
- A set of supporting document chunks, as context for answering the question accurately.

3.3.2 Dataset for Global Query Evaluation

To evaluate performance on global question answering, we constructed a set of 67 global questions from the Wikipedia articles corpus. These global questions are designed to assess a model's ability to perform global summarization and answer queries that necessitate comprehending the entire document corpus, rather than focusing on specific details or facts.

The difference between ideal contexts of a global question and a non-global question can be understood from the following example:

Global question	Non-global (multi-hop) question
What are the main themes explored in Taylor Swift's album "The Tortured Poets Department," and how do they reflect her personal experiences?	How many years after the release of Taylor Swift's tenth studio album, Midnights, was 'The Tortured Poets Department' released?

Table 1: An example of a global and a non global question

3.3.3 Models Evaluated

1. **Baseline RAG:** Evaluated using the pre-processed Wikipedia articles document corpus.
2. **Finetuned RAG:** The same documents sourse as Baseline RAG, but the language model is further fine-tuned using RAFT-based strategy described in Section 3.1
3. **Baseline GraphRAG:** Evaluated using community summaries generated by GraphRAG model as input.
4. **RAFT-finetuned GraphRAG or CAGRA:** This model combines graph-based retrieval with the fine-tuned generation model.

3.3.4 Inference and Evaluation Pipeline

Document Corpus Preparation and Embedding
The evaluation starts by loading a Wikipedia document corpus from a serialized Python object ('loaded_articles.pkl'). Each document is split into overlapping chunks using a character-level text splitter (1000 character chunks with 100-character overlap) to maintain semantic continuity. These chunks are embedded using the all-MiniLM-L6-v2 model and indexed with FAISS (IndexFlatL2) for efficient vector similarity search.

Contextual Retrieval and Generation
For each question, the top-k relevant text chunks are retrieved from the FAISS index using cosine similarity of the question's embedding.

- k=3 for baseline RAG and finetuned RAG (to support broader context gathering)
- k=1 for GraphRAG and Finetuned GraphRAG (as summaries are already aggregated)

These retrieved chunks are concatenated to form the contextual prompt fed into the language model. Generation is performed using the microsoft phi-2 model, which has been loaded with configuration described under Section 3.1. Generation parameters are set to encourage focused, deterministic output: do_sample=True, temperature=0.1, num_beams=1, top_p=0.95, and max_new_tokens=100. A custom decoding function extracts the generated answer by removing the prompt prefix and filtering out irrelevant text. Post-processing ensures robustness by handling malformed or empty responses. Each question's predicted answer, expected answer, retrieved context, and expected context are logged into a .csv file.

Evaluation Metrics:
Having run inference on the four RAG-based models and stored results in .csv files, we quantitatively assessed their performance on both multi-hop and global reasoning questions using five diverse evaluation metrics.

- BERT Score [17] was used to compute contextual similarity using deep contextual embeddings, reflecting how well the generated response semantically aligns with the reference.
- BLEU score, a traditional n-gram overlap metric, was included to assess lexical precision, particularly the presence of key terms or phrases.
- ROUGE-1 F1 [18], which captures unigram overlap, was used to reflect both recall and precision in word-level matching.
- Semantic similarity [19] was computed using cosine similarity between Sentence-BERT embeddings, providing a robust measure of overall conceptual alignment.
- Finally, we calculated the token-level F1 score, which quantifies the harmonic mean of precision and recall, offering a balanced view of content correctness and coverage.

4 Results

4.1 Quantitative Results

Tables 2 and 3 present the evaluation results for multi-hop and global questions, respectively. As shown in Table 2, the CAGRA model consistently outperforms the other approaches—Baseline RAG, RAFT, and Baseline GraphRAG—across almost all evaluation metrics: BERTScore (0.86811), BLEU (0.03516), ROUGE-1 (0.28862) and F1 score (0.21699). Even for Semantic Similarity (0.59791), it was very close to the best. The highest score for each metric is highlighted in bold for clarity.

Metric	Baseline RAG model	RAFT	Baseline GraphRAG	**CAGRA**
BERT	0.83452	0.86363	0.86217	**0.86811**
BLEU	0.00897	0.03433	0.02681	**0.03516**
ROUGE-1	0.13509	0.26231	0.22821	**0.28862**
Semantic Sim	0.39983	0.51775	**0.60553**	0.59791
F1 Score	0.09596	0.19501	0.16479	**0.21699**

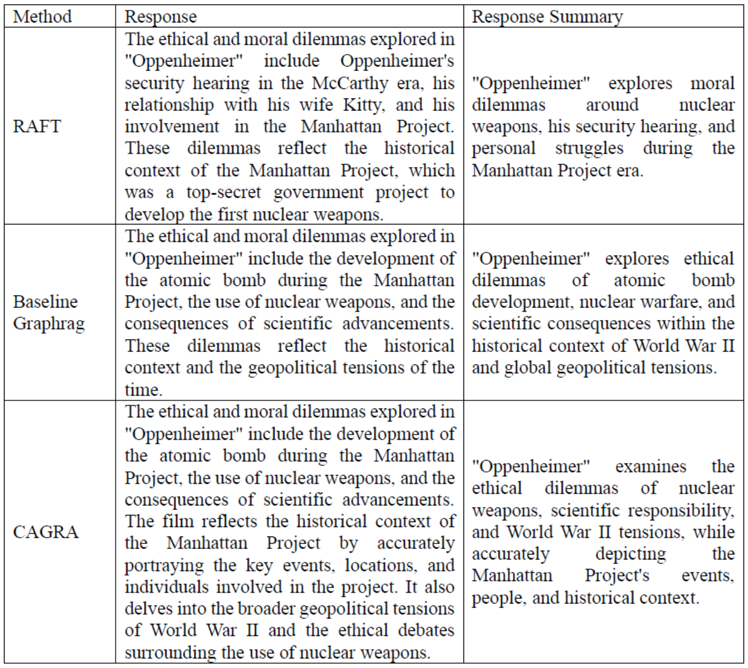
Table 2: Evaluation results for multi-hop questions

Metric	Baseline RAG model	RAFT	Baseline GraphRAG	**CAGRA**
BERT	0.82754	0.85014	0.87105	**0.87115**
BLEU	0.01617	**0.04093**	0.03647	0.03775
ROUGE-1	0.20951	0.25526	0.29712	**0.30254**
Semantic Sim	0.57187	0.66643	0.75946	**0.76356**
F1 Score	0.10154	0.16103	0.16993	**0.17145**

Table 3: Evaluation results for global questions

Table 3 summarizes performance on global questions, where CAGRA again demonstrates strong results, achieving top scores in BERTScore (0.87115), ROUGE-1 (0.30254), Semantic Similarity (0.76356), and F1 score (0.17145). Interestingly, all models perform better on global questions than on multi-hop ones, suggesting that multi-hop reasoning poses greater challenges. Additionally, both RAFT and Baseline GraphRAG outperform the original RAG model across all metrics, reinforcing previous insights that fine-tuning and graph-based enhancements contribute to improved question-answering capabilities.

4.2 Qualitative Results



Method	Response	Response Summary
RAFT	The ethical and moral dilemmas explored in "Oppenheimer" include Oppenheimer's security hearing in the McCarthy era, his relationship with his wife Kitty, and his involvement in the Manhattan Project. These dilemmas reflect the historical context of the Manhattan Project, which was a top-secret government project to develop the first nuclear weapons.	"Oppenheimer" explores moral dilemmas around nuclear weapons, his security hearing, and personal struggles during the Manhattan Project era.
Baseline Graphrag	The ethical and moral dilemmas explored in "Oppenheimer" include the development of the atomic bomb during the Manhattan Project, the use of nuclear weapons, and the consequences of scientific advancements. These dilemmas reflect the historical context and the geopolitical tensions of the time.	"Oppenheimer" explores ethical dilemmas of atomic bomb development, nuclear warfare, and scientific consequences within the historical context of World War II and global geopolitical tensions.
CAGRA	The ethical and moral dilemmas explored in "Oppenheimer" include the development of the atomic bomb during the Manhattan Project, the use of nuclear weapons, and the consequences of scientific advancements. The film reflects the historical context of the Manhattan Project by accurately portraying the key events, locations, and individuals involved in the project. It also delves into the broader geopolitical tensions of World War II and the ethical debates surrounding the use of nuclear weapons.	"Oppenheimer" examines the ethical dilemmas of nuclear weapons, scientific responsibility, and World War II tensions, while accurately depicting the Manhattan Project's events, people, and historical context.

Figure 3: Comparison of GRAFT, RAFT, and Baseline GraphRAG responses on "What ethical and moral dilemmas are explored in "Oppenheimer," and how do they reflect the historical context of the Manhattan Project?"

To further explore the differences in model behavior, we examine responses generated by CAGRA, RAFT, and Baseline GraphRAG for a representative example, as shown in Figure 3.

RAFT includes irrelevant details, such as Oppenheimer's security hearing during the McCarthy era and his relationship with his wife, Kitty, which detracts from the main ethical dilemmas addressed in the film, particularly those surrounding the atomic bomb, Oppenheimer's guilt and the impact of the bomb on Hiroshima and Nagasaki, which are central to the film's moral debate. Similarly, Baseline GraphRAG misses an important aspect by failing to highlight Oppenheimer's personal struggle, which is a crucial element in understanding the ethical implications of the Manhattan Project.

In contrast, CAGRA clearly addresses the development of the atomic bomb during the Manhattan Project and the use of nuclear weapons, which are pivotal to the film's narrative. It also demonstrates strong contextual accuracy by correctly placing the Manhattan Project within its historical context, linking it to World War II and the geopolitical tensions of the time. Moreover, CAGRA references key events, locations, and figures involved in the project, providing a well-rounded view of how the film portrays the realities of the Manhattan Project.

CAGRA's response stands out as the most accurate because it focuses on the central ethical issues, including the development of the atomic bomb, the dilemmas faced by scientists, and Oppenheimer's feelings of guilt after the bombings. It maintains strong relevance to the film's depiction of Oppenheimer's internal struggles and the broader consequences of nuclear warfare.

To gain insights into the performance of our models on global and non-global questions, we analyzed the embedding space visualizations using PCA as shown in Figure 4 and t-SNE also shown in Figure 5. The PCA and t-SNE plots reveal that CAGRA embeddings, both global and non-global, are

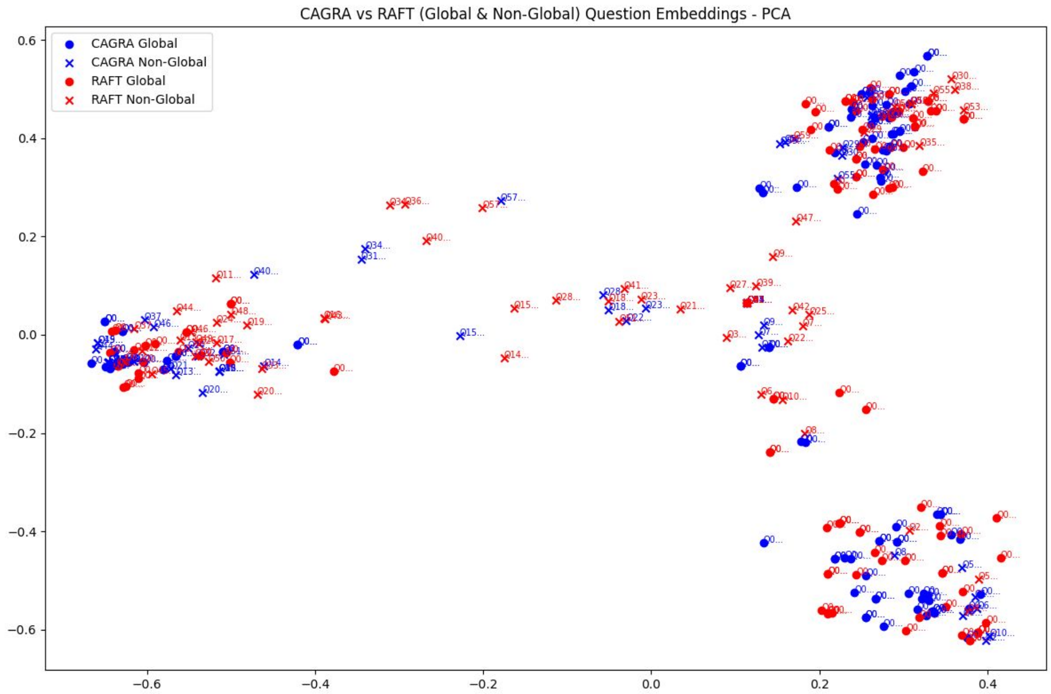


Figure 4: PCA plot of RAFT and CAGRA embeddings on global and non global questions

more compact and better clustered than RAFT, which suggests more consistent representation across question types. In contrast, RAFT embeddings appear more dispersed, especially for non-global questions, indicating less coherence. The tighter grouping in CAGRA highlights its ability to better capture semantic similarity and structure in the question space. Overall, this suggests that CAGRA generalizes better across varying question complexities.

5 Conclusion

In this project, we presented CAGRA, a new method that combines Retrieval-Augmented Fine-Tuning (RAFT) for improved context retrieval with GraphRAG to better handle multi-hop reasoning and global query summarization in large document collections. Our results show that CAGRA consistently outperforms strong baseline models like RAG, RAFT, and GraphRAG across various automatic evaluation metrics. These improvements highlight CAGRA's effectiveness in tackling both multi-hop and global question answering tasks.

6 Limitations and Future Works

Although CAGRA showed strong performance, our work has some limitations. The knowledge graph was built using entity and relation extraction from a limited set of documents, which can result in a sparse graph that misses important context for some queries. Also, due to limited computational resources—specifically using a single GPU—we couldn't experiment with larger or more complex models. Additionally, CAGRA didn't perform best on the BLEU score in our evaluation.

Despite these challenges, there are several directions for future work. One is adaptive graph construction, where the graph could be updated dynamically based on the query to improve relevance and efficiency. Another is explanation generation, using graph reasoning to make the model's answers more interpretable by showing how conclusions are drawn. Finally, improving scalability and efficiency through methods like model compression or faster graph traversal could help apply CAGRA to larger datasets and more complex tasks.

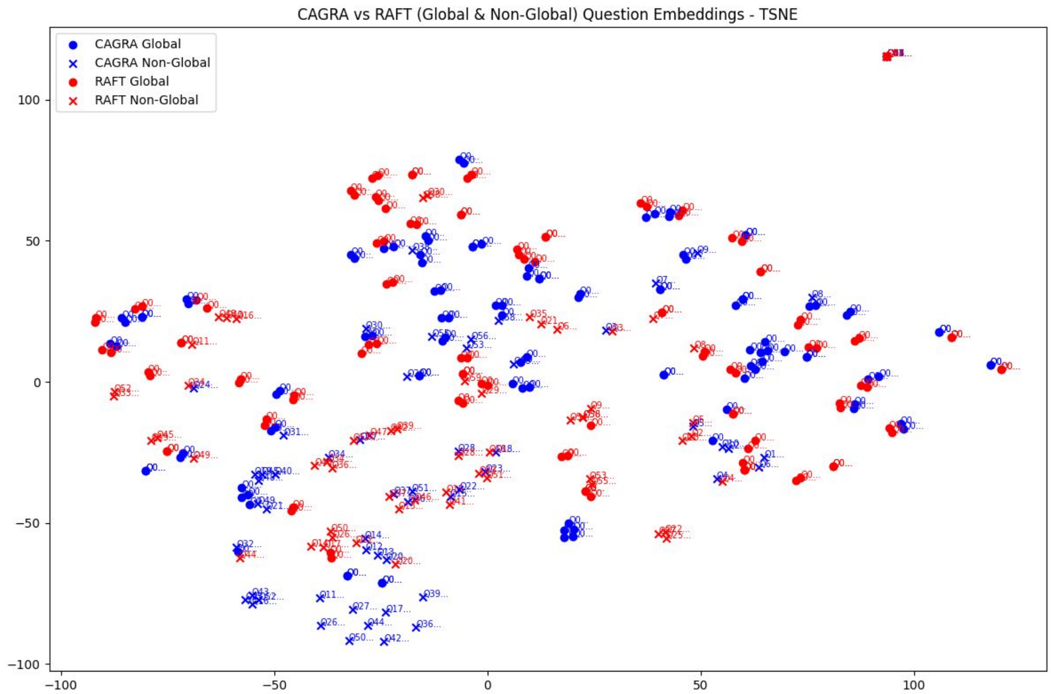


Figure 5: t-SNE visualization of RAFT and CAGRA embeddings on global and non global questions

By addressing these areas, CAGRA can become a more powerful and interpretable tool for question answering.

7 Contributions

- **Sharique Pervaiz:** led the development and integration of the RAFT-based fine-tuning pipeline. He implemented the QLoRA-based parameter-efficient training on the Phi-2 model, designed the instruction-style prompting strategy, and optimized training configurations. He also contributed to the quantitative evaluation setup and final model benchmarking.
- **Navtegh Singh Gill:** handled data curation, preprocessing, and evaluation setup. He curated the multi-hop and global query sets from Wikipedia, configured the FAISS retrieval pipelines, and orchestrated the end-to-end inference process. He also led the development of the embedding visualizations (PCA, t-SNE) and qualitative analysis.
- **Stuti Wadhwa:** was responsible for the design and implementation of the GraphRAG-based knowledge graph construction and community summarization pipeline. She integrated the Leiden partitioning, hierarchical summarization via Nano-GraphRAG, and the map-reduce query summarization logic. She also contributed significantly to writing, structuring the report, and presenting the results.

References

[1] Tianjun Zhang et al. “Raft: Adapting language model to domain specific rag”. In: *First Conference on Language Modeling*. 2024.

[2] Darren Edge et al. *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*. 2025. arXiv: `2404.16130 [cs.CL]`. **available at**: `https://arxiv.org/abs/2404.16130`.

[3] Zhengbao Jiang et al. *Active Retrieval Augmented Generation*. 2023. arXiv: `2305.06983 [cs.CL]`. **available at**: `https://arxiv.org/abs/2305.06983`.

[4] Anirudh Goyal et al. *Retrieval-Augmented Reinforcement Learning*. 2022. arXiv: `2202.08417 [cs.LG]`. **available at**: `https://arxiv.org/abs/2202.08417`.

[5] David Thulke et al. *Efficient Retrieval Augmented Generation from Unstructured Knowledge for Task-Oriented Dialog*. 2021. arXiv: `2102.04643 [cs.CL]`. **available at**: `https://arxiv.org/abs/2102.04643`.

[6] Yongqi Li et al. *Learning to Rank in Generative Retrieval*. 2023. arXiv: `2306.15222 [cs.CL]`. **available at**: `https://arxiv.org/abs/2306.15222`.

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

[8] Xiaoxin He et al. “G-retriever: Retrieval-augmented generation for textual graph understanding and question answering”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 132876–132907.

[9] Jiawei Zhang. “Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt”. In: *arXiv preprint arXiv:2304.11116* (2023).

[10] Minki Kang et al. “Knowledge graph-augmented language models for knowledge-grounded dialogue generation”. In: *arXiv preprint arXiv:2305.18846* (2023).

[11] Yu Wang et al. “Knowledge graph prompting for multi-document question answering”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 17. 2024, pp. 19206–19214.

[12] Wenhan Xiong et al. “Effective long-context scaling of foundation models”. In: *arXiv preprint arXiv:2309.16039* (2023).

[13] Yizhong Wang et al. “Self-instruct: Aligning language models with self-generated instructions”. In: *arXiv preprint arXiv:2212.10560* (2022).

[14] Mojan Javaheripi et al. “Phi-2: The surprising power of small language models”. In: *Microsoft Research Blog* (2023).

[15] Zhilin Yang et al. *HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering*. 2018. arXiv: `1809.09600 [cs.CL]`. **available at**: `https://arxiv.org/abs/1809.09600`.

[16] V. A. Traag, L. Waltman, and N. J. van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific Reports* 9.1 (Mar. 2019). ISSN: 2045-2322. DOI: `10.1038/s41598-019-41695-z`. **available at**: `http://dx.doi.org/10.1038/s41598-019-41695-z`.

[17] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: `1904.09675 [cs.CL]`. **available at**: `https://arxiv.org/abs/1904.09675`.

[18] Max Grusky. “Rogue Scores”. In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 1914–1934. DOI: `10.18653/v1/2023.acl-long.107`. **available at**: `https://aclanthology.org/2023.acl-long.107/`.

[19] Dhivya Chandrasekaran and Vijay Mago. “Evolution of Semantic Similarity—A Survey”. In: *ACM Computing Surveys* 54.2 (Feb. 2021), pp. 1–37. ISSN: 1557-7341. DOI: `10.1145/3440755`. **available at**: `http://dx.doi.org/10.1145/3440755`.