Scalable Graph-based Retrieval-Augmented Generation via Locality-Sensitive Hashing

Fangyuan Zhang*
Huawei Hong Kong Research Center
Hong Kong, China
zhang.fangyuan@huawei.com

Zhengjun Huang*
HKUST
Hong Kong, China
zhuangff@connect.ust.hk

Yingli Zhou*†
CUHK-Shenzhen
Shenzhen, China
yinglizhou@link.cuhk.edu.cn

Qintian Guo
HKUST
Hong Kong, China
qtguo@ust.hk

Wensheng Luo
Hunan University
Changsha, China
luowensheng@hnu.edu.cn

Xiaofang Zhou
HKUST
Hong Kong, China
zxf@cse.ust.hk

ABSTRACT

Graph-based Retrieval-Augmented Generation (Graph-RAG) has been proven to be effective in enhancing the performance of large language models (LLMs) by incorporating structured knowledge retrieval. Among various Graph-RAG designs, Tree-Organized RAG methods have shown particularly promising results due to their ability to organize and retrieve relevant context efficiently. However, these tree-based methods often face scalability challenges when applied to large-text datasets, resulting in high computational costs during graph construction phase and suboptimal query performance. To address these limitations, we propose a novel underlying architecture that utilizes random projection-based Locality-Sensitive Hashing (LSH) to enhance both the efficiency and scalability of Tree-Organized RAG systems. By exploiting LSH to guide data partitioning and node insertion in a multi-phased graph construction fashion, our method significantly accelerates graph construction and improves the accuracy and speed of query retrieval in large-text settings. Experimental results demonstrate that our LSH-enhanced multi-phased RAG system maintains the advantages of hierarchical organization while offering substantial improvements in both construction time and query effectiveness on large-scale datasets.

VLDB Workshop Reference Format:
Fangyuan Zhang, Zhengjun Huang, Yingli Zhou, Qintian Guo, Wensheng Luo, and Xiaofang Zhou. Scalable Graph-based Retrieval-Augmented Generation via Locality-Sensitive Hashing. VLDB 2025 Workshop: LLM+Graph.

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/EverM0re/Scalable-GraphRAG.

*Fangyuan Zhang, Zhengjun Huang, and Yingli Zhou are joint first authors.
†Yingli Zhou is the corresponding author.

1 INTRODUCTION

The rapid development of Large Language Models (LLMs) like GPT-4 [1], Gemini [33], Qwen [39], and LLaMA [34] has revolutionized natural language processing, achieving state-of-the-art performance in many tasks [26, 45, 47]. However, despite their scalability and generalization, LLMs face challenges in handling domain-specific questions, multi-hop reasoning, and deep contextual understanding [12, 50], often producing weak or incorrect answers and suffering from hallucinations [19, 20, 42], where fabricated or misleading information is produced due to the lack of domain-aligned or real-time knowledge in the model's pretraining corpus. While fine-tuning LLMs with domain-specific data [43] can help, it often incurs high costs and offers limited improvements, especially in specialized domains with few resources [2, 13].

To overcome these challenges, Retrieval-Augmented Generation (RAG) [6, 11, 24, 37, 38, 44] has emerged as a promising paradigm that augments LLMs with external knowledge sources, thereby improving factual accuracy, interpreting ability, and trustworthiness [23, 27, 48, 49, 51]. RAG methods typically extract pertinent information from external text corpora, relational datasets, or graph structures to assist in question-answering tasks. Recent RAG approaches have increasingly explored graph structured knowledge as external memory, leading to the development of graph-based RAG methods. These methods capture complex relationships among entities and support more structured, multi-hop retrieval [15–17, 29, 46]. More recently, tree-based RAG variants have gained attention for their hierarchical representation and efficient top-down traversal strategies, achieving promising results in multi-hop QA and long-context reasoning tasks [8, 18, 40].

Nevertheless, graph-based RAG methods face critical scalability issues when deployed on large-scale datasets. Specifically, the construction of large tree graphs often encounters failures due to memory and time limitations arising from information redundancy, which also significantly impacts query accuracy. [7, 52]. To address these limitations, we propose a novel framework that integrates Locality-Sensitive Hashing (LSH) [22] with tree-based RAG in a multi-phased construction fashion, enabling efficient and scalable retrieval in massive data regimes. Our method leverages LSH to perform approximate similarity search, effectively pruning the candidate space before hierarchical tree traversal. This design significantly boosts retrieval efficiency while maintaining high answer accuracy.

In summary, our contributions are as follows:

- We identify and analyze the limitations of current LLMs in handling domain-specific and multi-hop questions.
- We propose a novel LSH-enhanced tree-based RAG architecture in a multi-phased construction fashion to support light-weighted graph construction and reduce time consumption due to information redundancy.
- We conduct extensive experiments across multiple domains and QA benchmarks in both specific and abstract styles, demonstrating the superior scalability and accuracy of our method compared to existing Graph-RAG systems.

2 RELATED WORK

In this section, we introduce some related works of RAG and Graph-RAG, along with the introduction to Tree-Organized Retrieval.

2.1 Graph Retrieval Augmented Generation

When faced with domain-specific queries or multi-hop queries, current LLMs may generate a weak or meaningless and non-factual responses, i.e., hallucinations [19, 20]. To mitigate this problem, Retrieval-Augmented Generation (RAG) [6, 11, 38] has emerged as a powerful framework for enhancing the factual accuracy and grounding capabilities of LLMs which combines a retriever that fetches relevant documents from an external corpus with a LLM that uses them to produce more accurate and informed responses(i.e., Vanilla RAG).

In contrast, Graph-RAG [16] introduces a structured, offline pre-processing stage that transforms the raw corpus into a graph or hierarchical structure, enabling more efficient and accurate retrieval during the generation phase [46]. By encoding semantic relationships between documents, passages, or entities ahead of time, GraphRAG reduces redundancy and improves the contextual coherence of retrieved results. This offline organization significantly accelerates retrieval at inference time and enhances the relevance of the supporting evidence, leading to improved response quality for the LLM compared to traditional RAG [15].

2.2 Tree-Organized Graph Retrieval

Tree-Organized graph retrieval [7] is a novel approach in retrieval-augmented generation that organizes knowledge or documents into a hierarchical tree or tree-like structure, enabling faster and more semantically meaningful retrieval in large-scale systems.

Instead of treating knowledge as a flat graph (like traditional Graph-RAG), tree-based methods leverage semantic hierarchies, grouping related nodes under shared "topics" or "supernodes", allowing for logarithmic-time access paths and multi-resolution reasoning. Recent research has introduced significant variants of tree-structured RAG, including hierarchical organization trees [3, 7], hierarchical entity trees [25], Monte Carlo Tree Search (MCTS) frameworks [8, 18], and other tree-based structures [31]. These approaches have demonstrated notable improvements in both query accuracy and the efficiency of graph construction. Furthermore, recent evaluations of Graph-RAG systems have highlighted the

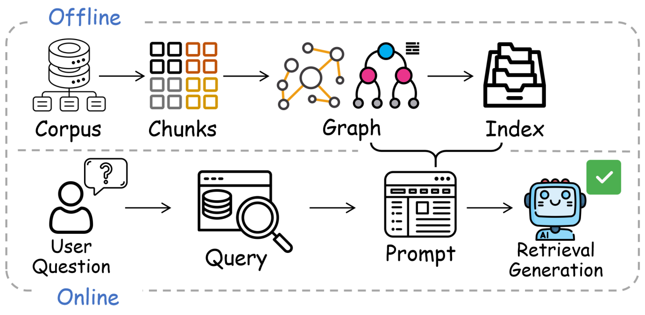

Figure 1: Basic GraphRAG workflow.

benefits of Tree-Organized graph retrieval methods [52], but due to applications of Gaussian Mixture Models (GMMs), large-scaled datasets face the problem of extension or failure of the graph building stage.

3 PRELIMINARIES

In this section, we offer basic introduction to the usual workflow of Graph-RAG systems and concepts of locality-sensitive hashing (LSH).

3.1 GraphRAG workflow

Graph-RAG systems enhance language model outputs by incorporating external knowledge through a three-stage workflow: graph building, index construction, and retrieval based generation [52]. As shown in figure 1, in the offline stage, documents are encoded into dense vector representations and optionally structured into a semantic graph structure to reflect relationships such as topical similarity. The second stage constructs an efficient index to support fast retrieval of relevant content. Finally, in the online stage, the system retrieves the top-matching documents based on the query and feeds, allowing the generative model to produce more informed, contextually grounded responses [16].

3.2 Locality-Sensitive Hashing

Locality Sensitive Hashing (LSH) is a well-established technique for efficient similarity search in high-dimensional space [22]. It provides a probabilistic method for hashing input vectors such that similar vectors are mapped to the same or nearby buckets with high probability. As can be seen in figure 2, unlike traditional hash functions, the hashing process of LSH intends to maximize the collision of similar vectors to form clusters. LSH accelerates similarity search by hashing high-dimensional vectors into buckets using functions that ensure similar vectors are likely to collide, allowing approximate nearest neighbors to be found efficiently from a reduced candidate set. In RAG systems, LSH is usually employed to quickly retrieve semantically relevant documents from a large corpus given an input query embedding [21]. Compared to exhaustive similarity search or exact nearest neighbors (e.g., via FAISS [4] or brute-force dot product), LSH offers a faster alternative with lower memory and compute requirements, making it attractive for latency-sensitive applications such as real-time QA or chatbot systems.

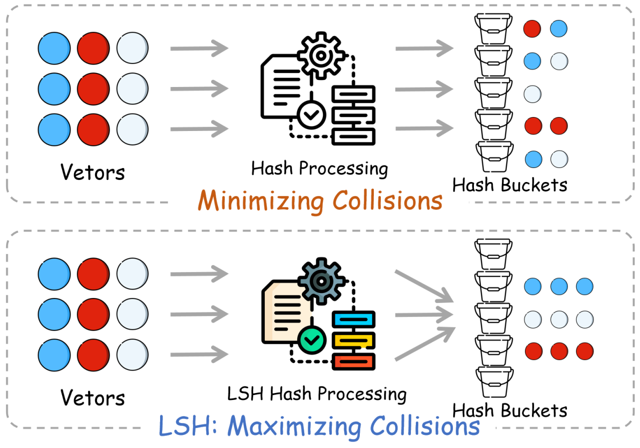

Figure 2: Comparison of LSH hashing process and regular hash functions.

4 PIPELINE

Building on the idea that LSH can not only be applied in the retrieving stage but can also be utilized in graph building, our method uses a novel hyperplane-based LSH method to construct a layer-based tree building pipeline. In this section we will break down our pipeline and introduce component by component.

4.1 LSH with Random Projections

To address the problems proposed above, a novel method of **random projection-based LSH** is proposed, which is not only efficient but also well-suited to clustering embeddings generated for chunk similarity matching. This approach maintains the inherent computational advantages of LSH while offering a more robust and interpretable clustering process.

Hyperplane Construction. The key idea behind our approach is to project high-dimensional vectors onto a set of random hyperplanes. These hyperplanes serve as decision boundaries that allow us to categorize vectors based on their orientation relative to these planes. The external corpus is first chopped up into equal sized chunks and transformed into normalized high dimensional vectors, denoted by $v_i \in \mathbb{R}^d$. Let $\{h_1, h_2, ..., h_k\}$ denote a set of k randomly generated hyperplanes, each represented as a vector in the same dimensional space. The hash function for a given vector v is computed as follows:

$$\text{hash}(v) = [\text{sign}(v \cdot h_1), \text{sign}(v \cdot h_2), ..., \text{sign}(v \cdot h_k)]$$

Here, each projection $v \cdot h_i$ corresponds to the inner product:

$$v \cdot h_i = \sum_{j=1}^{d} v_j \cdot h_{i,j}$$

which determines the side of the hyperplane h_i on which the vector v lies. If the dot product is positive, the vector lies on one side of the hyperplane. Otherwise, it lies on the opposite side. This produces a binary value for each projection: 1 for positive and 0 for negative. The resulting binary vector is concatenated to form a k-bit hash code. In our method, the dimensionality of the hash codes corresponds to the dimensionality of the chunk embeddings, with each hyperplane contributing a single bit to the resulting hash code.

Similarity and Bucket Assignment. Once hash codes are generated, bucket assignment is performed by comparing the hash codes using Hamming distance[28], a common metric for binary vectors. Given two hash codes $h(v_1)$ and $h(v_2)$, the Hamming distance is defined as:

$$\text{Ham}(h(v_1), h(v_2)) = \sum_{i=1}^{k} \mathbb{I}[h_i(v_1) \neq h_i(v_2)]$$

where $\mathbb{I}$ is the indicator function. A smaller Hamming distance implies higher similarity between the original corpus chunks, guiding the clustering process and facilitating efficient retrieval. As can be seen in Figure 3, this technique effectively preserves local semantic similarity while enabling scalable hashing for high-dimensional inputs.

Theoretical Analysis. Given prior analysis, we now provide an analysis of the correctness of our approach of LSH using random projections. Let $v_1, v_2 \in \mathbb{R}^d$ be two normalized vectors, the angular distance between the two vectors is given by θ, which is the angle between them in the high-dimensional space. Then the cosine similarity between these vectors is then defined as:

$$\text{sim}(v_1, v_2) = \cos(\theta)$$

where $\cos(\theta)$ represents the cosine of the angle θ between the vectors. This measure is a key component in determining how similar the two vectors are: the closer $\cos(\theta)$ is to 1, the more similar the vectors are. Then consider a random hyperplane h that is sampled from a distribution which satisfies the properties of LSH. The hyperplane acts as a decision boundary that will divide the space into two halves.

Given the above definitions, we now present the following theorem that quantifies the probability of a collision between the two vectors v_1 and v_2.

Theorem 4.1. *Given two normalized vectors $v_1, v_2 \in \mathbb{R}^d$ and a random hyperplane h, the probability $P(h(v_1) = h(v_2))$ that both vectors v_1 and v_2 are on the same side of hyperplane h increases as their angular distance θ decreases. The probability is given as follows:*

$$P(h(v_1) = h(v_2)) = \frac{1 + \cos(\theta)}{2}$$

This theorem provides a key insight into how the method of random projection-based LSH ensures that similar vectors are more likely to collide (i.e., be mapped to the same bucket). The probability that two vectors lie on the same side of the proposed hyperplane is directly related to the cosine similarity between the vectors. According to the method, whether the vectors are on the same side of the hyperplane is reflected in the corresponding bits of their hash codes. Therefore, across a set of n hyperplanes, the greater the number of hyperplanes on which the vectors fall on the same side, the smaller their Hamming distance will be. This, in turn, increases the likelihood that they will be hashed into the same bucket. In conclusion, this demonstrates that the greater the similarity between the vectors, the higher the probability they will be assigned to the same bucket, thereby validating the correctness of the random projection-based LSH approach.

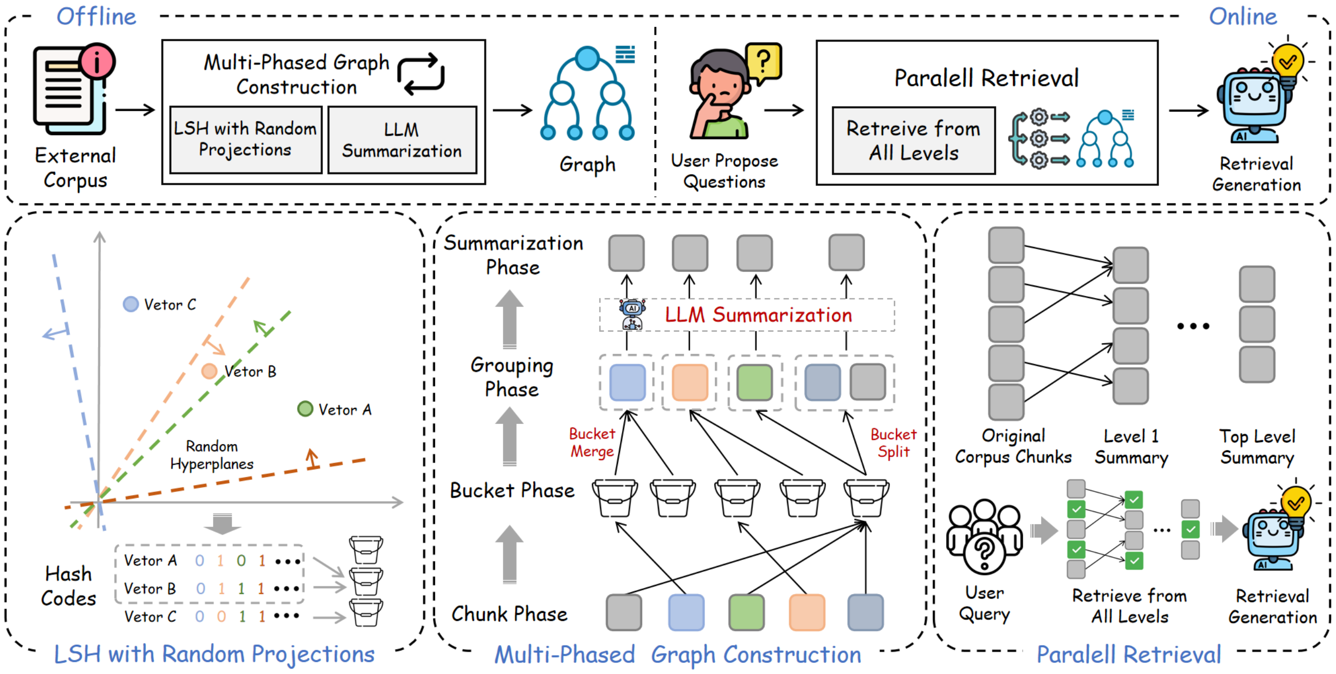

Figure 3: Overview of our method.

4.2 Multi-phased Graph Construction

Utilizing the proposed random projection LSH method, the multi-phased graph construction is carried out in the following order.

Bucket Rearranging and Grouping. Once the hash codes have been computed, the vectors are assigned to buckets according to their Hamming distance. While this step groups similar vectors together, the distribution of vectors across buckets is often imbalanced. Some buckets may contain a large number of vectors, while others may have very few.

To address this issue, we employ a post-processing step that ensures the final clusters meet predefined size constraints. Let B_i denote the i-th bucket, and $|B_i|$ its size. Let $s_{\min}$ and $s_{\max}$ represent the lower and upper bounds of acceptable cluster sizes. Then each bucket is evaluated based on the condition:

$$s_{\min} \leq |B_i| \leq s_{\max}$$

Buckets not satisfying this condition are processed accordingly:

- **Splitting** overly large buckets: If $|B_i| > s_{\max}$, the bucket is split into smaller clusters, typically using a secondary clustering strategy or random partitioning.
- **Merging** small buckets: If $|B_i| < s_{\min}$, it is merged with its most similar neighboring bucket B_j, minimizing Hamming distance $\mathrm{Ham}(B_i, B_j)$.
- **Enforcing cluster size constraints**: After merging or splitting, each final cluster C_k satisfies:

$$\mathbb{I}\left[s_{\min} \leq |C_k| \leq s_{\max}\right] = 1$$

This post-processing step is critical for balancing the trade-off between maintaining the advantages of LSH (e.g., speed and locality preservation) and ensuring that the resulting clusters are both meaningful and appropriately sized for downstream tasks.

Multi-phased construction and parallel retrieving. Once the grouping phase has been completed, the process transitions into the summarization phase, where a LLM is employed to distill the essential content of each chunk group, thereby generating a new set of summarized chunks. These newly created chunks serve as a refined abstraction of the original content, capturing the core elements while reducing redundancy. The resulting layer of summarized chunks is then subjected to the graph-building phase once again. This iterative process continues until the top-level summary layer reaches a sufficient level of condensation, ensuring that the information retained is both relevant and contextually rich.

This multi-layered approach culminates in a tree structure, with the base layer corresponding to the original corpus chunks and each subsequent layer representing a progressively more condensed and concentrated aggregation of information. As the tree structure ascends, the nodes represent higher-level abstractions, each retaining the essential content of its predecessors while omitting unnecessary details.

In the retrieval stage, the system leverages an optimizer to perform an efficient top-k search across all phases of the graph construction. This search process allows the model to retrieve the most relevant information from each layer of the hierarchical graph structure, selecting the top-k results that best match the query's context. By utilizing the multi-phased graph construction, the retrieval system can dynamically access information at various levels of abstraction, ensuring that the retrieved content is both accurate and contextually aligned with the query's requirements. This approach significantly enhances the scalability and effectiveness of the retrieval process, particularly for large-scale datasets with complex hierarchical structures.

Complexity analysis. The multi-phased graph construction process proceeds hierarchically, where each layer of the graph is built by clustering the chunks from the previous layer. The process stops under two conditions: (1) when the maximum number of layers L has been reached, or (2) when the number of chunks in the current layer drops below a predefined minimum threshold, $n_{\min}$. Through the setting of bucket merging and splitting process, at each layer the number of chunks is reduced by at least k. Assign the initial number of chunks to ben_c.

For the first grouping stage, the clustering operation involves n_c chunks and has a time complexity of $O(n_c \cdot d)$, where d is the dimensionality of each chunk. At each subsequent layer l, the number of chunks is reduced to approximately $\frac{n_c}{k^l}$, and the clustering operation at layer l takes $O\left(\frac{n_c}{k^l} \cdot d\right)$. The total time complexity across all layers can be expressed as the sum of the complexities at each layer:

$$T_{\text{total}} = O\left(n_c \cdot d\right) + O\left(\frac{n_c}{k} \cdot d\right) + O\left(\frac{n_c}{k^2} \cdot d\right) + \cdots$$

This sum forms a geometric series, which can be approximated as:

$$T_{\text{total}} = O\left(n_c \cdot d \cdot \frac{k^L}{k-1}\right)$$

Thus, the time complexity of the graph construction process is $O\left(n_c \cdot d \cdot \frac{k^L}{k-1}\right)$ in worst case, which scales with the initial number of chunks and the maximum number of layers. In the cases where the number of chunks falls below $n_{\min}$ before reaching the maximum number of layers, the process stops even earlier.

4.3 Advantages of the Proposed Approach

The proposed hyperplane based LSH clustering method offers several compelling advantages, making it a valuable tool for tasks involving high-dimensional embeddings:

- **Semantic Alignment**: By leveraging angular similarity through dot products, this method aligns well with the characteristics of modern embedding models, such as those based on transformer architectures, where semantic similarity is often represented by small angular distances[36].
- **Scalability**: The hash computation is highly efficient, with a time complexity of $O(kd)$, and the bucket assignment process can be easily parallelized, allowing for scalability in large datasets.
- **Interpretability**: The binary hash codes provide an interpretable way to track and understand the cluster formation process. Each cluster can be traced back to a specific hash code, enabling transparency in the clustering process.
- **Robustness to Size Variability**: The ability to split large buckets and merge small ones ensures that the final clusters are balanced and reflect the inherent structure of the data, while still maintaining efficient grouping.

The proposed multi-phase graph construction and parallel retrieval strategy also offers several notable advantages, particularly in improving retrieval quality, representation granularity, and response robustness:

- **Hierarchical Abstraction**: The system iteratively summarizes information into a hierarchy, enabling access to both detailed and abstracted data, which is useful for high-level reasoning and synthesis tasks.
- **Multi-Level Semantic Coverage**: By preserving all graph layers during retrieval, the system allows extraction of both detailed and summarized information, supporting context-specific and generalized evidence retrieval in a single query.
- **Parallel Retrieval Efficiency**: The parallel top-k retrieval across layers reduces latency and improves evidence gathering, enhancing both relevance and factual accuracy in model outputs.

5 EXPERIMENTS

This chapter elaborates the experiments we conducted utilizing our method and frontier baseline models. Section 5.1 introduces the setup of the experiments, including dataset and baseline introduction along with evaluation metrics. The results are then displayed in section 5.2.

5.1 Experiment Setup

Datasets: To evaluate the efficiency and accuracy of our method, we employ 4 real-world datasets, including both specific and abstract queries. The following are basic information about the specific datasets utilized in our work.

- **HotpotQA** [41] is a large-scale question-answering dataset comprising approximately 113,000 Wikipedia-based question-answer pairs. It is specifically designed to test and enhance multi-hop reasoning capabilities in natural language processing models, regarded as one of the most authoritative and tested benchmark dataset in the field.
- **MuSiQue** [35] is a dataset designed to advance research in multi-hop question answering which comprises approximately 25,000 questions that require reasoning across 2 to 4 interconnected facts to derive an answer, testing ability on handling complex question structures and connective reasoning.
- **MultiHopRag** [32] is a dataset specifically designed to evaluate RAG systems on complex multi-hop queries. It comprises 2,556 queries, each requiring the integration of information from 2 to 4 distinct documents, reflecting real-world scenarios where answers are not confined to a single source.
- **ALCE** [10] is the first benchmark for automatically evaluating citation quality in long-text responses generated by LLMs. It focuses on realistic, end-to-end question answering tasks that require both retrieving information and citing relevant sources accurately.

Aside from the above mentioned specific datasets, we employ a abstract dataset **UltraDomain** [30] which comprises tasks with long context and high-level query on over 20 specialized domains, requiring higher understanding and summary abilities. We will employ the domain dataset of computer science, legal and mixed knowledge. We also employ the abstract summary problems offered by Multihop-RAG, known as **MultihopSum** [32].

Baselines: Three graph-based RAG systems are chosen as the baseline models for our method.

Table 1: Graph construction time on specific datasets

Methods	MultiHop	MuSiQue	ALCE	HotpotQA
Our Method	71.98s	866.93s	2442.82s	1979.32s
GraphRAG	2215.66s	19392.49s	50274.21s	45381.43s
Raptor	945.41s	-	-	-
RaptorBal	103.76s	1061.72s	3203.82s	2918.33s

- **GraphRAG** [5]: GraphRAG is an innovative approach that enhances RAG by integrating structured knowledge graphs, enabling more accurate and context-aware retrieval. By modeling semantic relationships between entities and passages, it supports multi-hop reasoning and improves the relevance and coherence of retrieved content, particularly for complex or domain-specific queries.
- **RAPTOR** [31]: RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval) is an advanced Tree-Organized RAG technique developed to enhance the performance of LLMs when processing extensive and complex documents. RAPTOR employs a recursive process of embedding, clustering, and summarizing text chunks to construct a hierarchical tree structure from the bottom up. Studies have demonstrated that RAPTOR significantly outperforms traditional retrieval-augmented LLMs, particularly in tasks requiring complex and multi-step reasoning[52].
- **RaptorBalanced** [52]: Recent evaluations of RAPTOR shows that on large-scaled datasets, the Gasussian mixture classifying method fails to separate the chunk embeddings, so RaptorBalanced is introduced.

Evaluation metrics: According to recent work [52], specific QA tasks will be evaluated with adjusted accuracy and recall, except for ALCE, which metric will follow existing work [31] using string recall (STRREC), string exact matching (STREM) and string hit (STRHIT). For the abstract QA tasks, prior work [5, 14] proposed a LLM-guided evaluation method, where a LLM evaluator is utilized to evaluate the performance of two models based on comprehensiveness, diversity, etc. In this work, the performance will be evaluated based on comprehensiveness, diversity and empowerment and a final overall result will be given. The result of the evaluation will be displayed in a win rate percentage form.

Implementation details: All of our experiments are produced on NVIDIA A100 80GB GPUs. For clarity and justice, our method and all baselines will be implemented on the recently proposed unified GraphRAG framework [52]. Baselines will inherit the parameter settings proposed in their original paper.

5.2 Results

Three main results are conducted to evaluate our method. Performance on specific QA and abstract QA will be tested and compared with state-of-the-art Graph-RAG systems, where accuracy, recall and graph building time will be evaluated. Also, a chunk size experiment is conducted to test the robustness of our method against different chunking method.

Specific QA: The performance of our proposed method is evaluated on four real-world, domain-specific QA datasets that have been widely adopted in recent research. Given that RAPTOR has been demonstrated as one of the strongest graph-based RAG systems [52], and that GraphRAG is among the most established models in this category, we adopt GraphRAG, RAPTOR, and RaptorBalanced as baseline methods. The evaluation focuses on both accuracy and recall, as shown in Figure 4, while the corresponding graph construction times are summarized in Table 1.

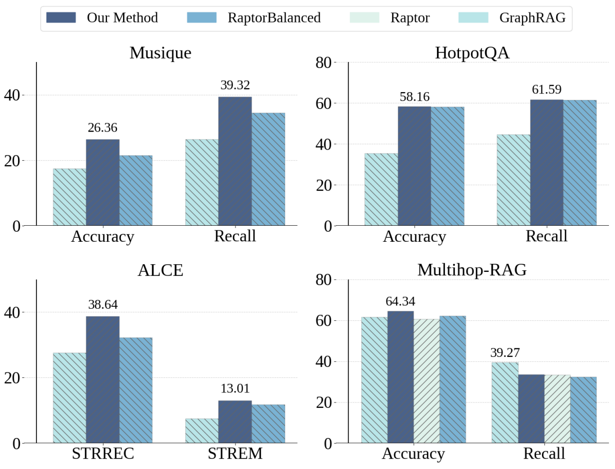

Figure 4: Accurcay and recall on specific datasets

Across all four benchmark datasets, our method achieves consistent and substantial improvements over both RaptorBalanced and GraphRAG. Notably, RAPTOR fails to complete graph construction on certain datasets due to instability in its Gaussian Mixture-based clustering component, and is thus only evaluated on the MultiHopRAG dataset.

In the Musique dataset, our method achieves absolute gains of approximately 5% in both accuracy and recall over RaptorBalanced, while also significantly reducing graph construction time compared to RAPTOR and GraphRAG, respectively. For the HotpotQA dataset, although the performance gap is narrower, our method still leads in both metrics, highlighting its robustness in multi-hop reasoning.

On the MultiHopRAG dataset, while our recall is slightly lower than that of GraphRAG, our model achieves the highest accuracy at 64.34%, indicating better answer precision. The most significant improvement is observed on the ALCE dataset, where our model surpasses RaptorBalanced by over 6 percentage points in structural recall (STRREC) and more than 1 percentage point in structural exact match (STREM), reflecting stronger capacity in retrieving semantically and structurally relevant evidence. These results suggest that our multilayered graph construction strategy effectively generates more informative and structured graphs for downstream retrieval and generation.

In addition to retrieval effectiveness, our method exhibits markedly superior graph construction efficiency across all datasets. This is attributed to the hyperplane-based LSH mechanism, which enables both faster and more scalable graph construction. The alignment of improved accuracy and reduced offline preprocessing time further demonstrates the robustness and practicality of our approach.

Collectively, these results provide strong empirical evidence for the effectiveness and efficiency of our model in diverse domain-specific QA scenarios. The consistent outperformance across datasets

Table 2: Abstract QA result on our method vs. GraphRAG

Dataset	Comprehensive	Diversity	Empower	Overall
Mix	56%	52%	49%	51%
CS	53%	58%	48%	55%
legal	33%	54%	42%	42%
MultiSum	56%	42%	55%	52%

Table 3: Abstract QA result on our method vs. RAPTOR

Dataset	Comprehensive	Diversity	Empower	Overall
Mix	67%	62%	47%	54%
CS	52%	48%	41%	46%
legal	58%	42%	61%	52%
MultiSum	51%	57%	53%	53%

and metrics affirms the superiority of our method in balancing retrieval quality with offline construction cost, positioning it as a more reliable and scalable alternative to existing graph-based RAG frameworks.

Abstract QA: In this section we evaluate the performance of our method, GraphRAG and RAPTOR on abstract QA datasets, the results are displayed in Table 2 and Table 3 in the form of head to head win rates. The win rate of our model against the corresponding baseline is displayed. Based on the experimental results, our proposed method demonstrates clear superiority over both GraphRAG and RAPTOR in handliing Abstract Queries across multiple datasets. When comparing our method to GraphRAG, we observe consistently higher performance in key metrics such as comprehensiveness, diversity, and empowerment. In particular, our method shows marked improvements in the "CS" and "Legal" datasets, where it outperforms GraphRAG by a significant margin, especially in comprehensiveness and diversity.

Moreover, when compared to RAPTOR, our method again excels across all metrics and datasets. It achieves notable improvements in the "Mix" and "Multihop-Sum" datasets, particularly in comprehensiveness and diversity, while also leading in empowerment across all domains. The overall winning percentage of our method surpasses both competitors, demonstrating its robust and versatile performance.

These results indicate that our approach offers a more effective and reliable solution for Abstract Query generation, outperforming existing systems in terms of both consistency and robustness across a wide range of datasets. The clear advantages observed in this comparison underscore the effectiveness of our method, establishing it as a superior alternative in the field.

Chunk size: Recent studies have shown that chunk size plays a critical role in the performance of RAG systems, where smaller and more focused chunks tend to improve accuracy at the expense of higher computational costs [9]. To systematically evaluate the impact of chunk size on retrieval effectiveness and system efficiency, we conducted a series of experiments measuring accuracy, recall, and graph building time across varying chunk sizes. As illustrated in figure 5, variations in chunk size exert minimal influence on retrieval accuracy and recall. Accuracy remains relatively stable, with

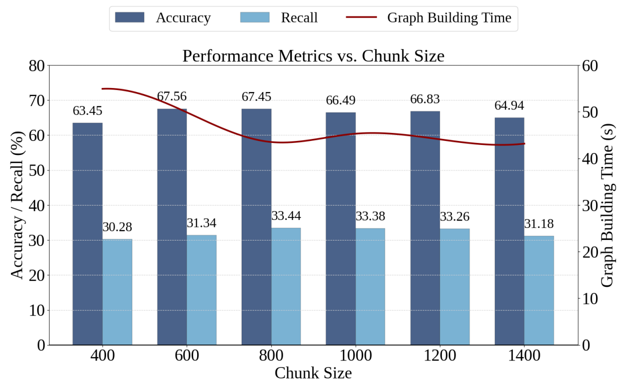

Figure 5: Accuracy, recall and graph building time across different chunk sizes.

a modest peak observed at chunk sizes between 600 and 800, reaching approximately 67.5%, while recall shows a slight upward trend from 30.3% to 33.4% up to size 1000, after which it plateaus. These findings indicate that retrieval performance is largely resilient to moderate adjustments in chunk granularity. In contrast, graph construction time demonstrates more pronounced changes, decreasing from 54.93 seconds at a chunk size of 400 to 43.17 seconds at 1400. Although increasing the number of chunks introduces additional computational overhead and results in longer graph building times, this remains within acceptable limits. Overall, the results suggest that chunk size can be effectively tuned to balance computational efficiency and retrieval accuracy without significantly degrading system performance for our method.

6 CONCLUSION

In this paper, we presented a scalable and efficient Tree-Organized Graph-RAG framework enhanced with hyperplane-based Locality-Sensitive Hashing. By integrating LSH into the clustering stage, our method improves tree construction speed and retrieval accuracy while maintaining semantic coherence. Experimental results across multiple QA benchmarks demonstrate consistent performance gains over strong baselines, confirming the effectiveness of our approach in both specific and abstract query settings. This work underscores the potential of combining approximate similarity search with hierarchical structures for advanced retrieval-augmented generation tasks.

ACKNOWLEDGMENTS

The research work described in this paper was supported by Hong Kong Research Grants Council and Guangzhou HKUST Fok Ying Tung Research Institute (grant# 16202722, 16210625, T43-513/23-N, T22-607/24N, 2023ZD007). It was partially conducted in JC STEM Lab of Data Science Foundations funded by The Hong Kong Jockey Club Charities Trust.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Jacob Browning. 2024. Getting it right: the limits of fine-tuning large language models. *Ethics and Information Technology* 26, 2 (2024), 36.

[3] Xinyue Chen, Pengyu Gao, Jiangjiang Song, and Xiaoyang Tan. 2024. HiQA: A Hierarchical Contextual Augmentation RAG for Multi-Documents QA. *arXiv preprint arXiv:2402.01767* (2024).

[4] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281* (2024).

[5] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv preprint arXiv:2404.16130* (2024).

[6] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. 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*. 6491–6501.

[7] Masoomali Fatehkia, Ji Kim Lucas, and Sanjay Chawla. 2024. T-RAG: lessons from the LLM trenches. *arXiv preprint arXiv:2402.07483* (2024).

[8] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Jingyi Song, and Hao Wang. 2025. Airrag: Activating intrinsic reasoning for retrieval augmented generation via tree-based search. *arXiv preprint arXiv:2501.10053* (2025).

[9] Paulo Finardi, Leonardo Avila, Rodrigo Castaldoni, Pedro Gengo, Celio Larcher, Marcos Piau, Pablo Costa, and Vinicius Caridá. 2024. The chronicles of rag: The retriever, the chunk and the generator. *arXiv preprint arXiv:2401.07883* (2024).

[10] Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling Large Language Models to Generate Text with Citations. *arXiv preprint arXiv:2305.14627* (2023).

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

[12] Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. 2023. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems* 36 (2023), 5539–5568.

[13] Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, Dinesh Manocha, et al. 2024. A closer look at the limitations of instruction tuning. *arXiv preprint arXiv:2402.05119* (2024).

[14] Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. Available at arXiv or similar venue (specific venue not provided).

[15] Haoyu Han, Harry Shomer, Yu Wang, Yongjia Lei, Kai Guo, Zhigang Hua, Bo Long, Hui Liu, and Jiliang Tang. 2025. RAG vs. GraphRAG: A Systematic Evaluation and Key Insights. *arXiv preprint arXiv:2502.11371* (2025).

[16] Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. 2024. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309* (2024).

[17] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506* (2024).

[18] Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025. MCTS-RAG: Enhancing Retrieval-Augmented Generation with Monte Carlo Tree Search. *arXiv preprint arXiv:2503.20757* (2025).

[19] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint arXiv:2311.05232* (2023).

[20] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* 43, 2 (2025), 1–55.

[21] Qazi Mudassar Ilyas. 2025. Enhancing the RAG Pipeline Through Advanced Optimization. *Generative AI Foundations, Developments, and Applications* (2025), 59.

[22] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. 2021. A survey on locality sensitive hashing algorithms and their applications. *arXiv preprint arXiv:2102.08942* (2021).

[23] Jiarui Li, Ye Yuan, and Zehua Zhang. 2024. 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).

[24] Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399* (2023).

[25] Zihang Li, Yangdong Ruan, Wenjun Liu, Zhengyang Wang, and Tong Yang. 2025. CFT-RAG: An Entity Tree Based Retrieval Augmented Generation Algorithm With Cuckoo Filter. *arXiv preprint arXiv:2501.15098* (2025).

[26] Zhixun Li, Liang Wang, Xin Sun, Yifan Luo, Yanqiao Zhu, Dingshuo Chen, Yingtao Luo, Xiangxin Zhou, Qiang Liu, Shu Wu, et al. 2023. Gslb: The graph structure learning benchmark. *Advances in Neural Information Processing Systems* 36 (2023), 30306–30318.

[27] Jingyu Liu, Jiaen Lin, and Yong Liu. 2024. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338* (2024).

[28] Mohammad Norouzi, David J Fleet, and Russ R Salakhutdinov. 2012. Hamming distance metric learning. *Advances in neural information processing systems* 25 (2012).

[29] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921* (2024).

[30] Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. MemoRAG: Moving Towards Next-Gen RAG via Memory-Inspired Knowledge Discovery. *arXiv preprint arXiv:2409.05591* (2024).

[31] Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. Raptor: Recursive Abstractive Processing for Tree-Organized Retrieval. In *The Twelfth International Conference on Learning Representations (ICLR)*.

[32] Yixuan Tang and Yi Yang. 2024. Multihop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-hop Queries. *arXiv preprint arXiv:2401.15391* (2024).

[33] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[34] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[35] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[36] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *APSIPA transactions on signal and information processing* 8 (2019), e19.

[37] Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. 2025. ArchRAG: Attributed Community-based Hierarchical Retrieval-Augmented Generation.

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

[39] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[40] Yahe Yang and Chengyue Huang. 2025. Tree-based RAG-Agent Recommendation System: A Case Study in Medical Test Data. *arXiv preprint arXiv:2501.02727* (2025).

[41] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *arXiv preprint arXiv:1809.09600* (2018).

[42] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Mu-Nan Ning, Yu-Yang Liu, and Li Yuan. 2023. Llm lies: Hallucinations are not bugs, but features as adversarial examples. *arXiv preprint arXiv:2310.01469* (2023).

[43] Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. 2024. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193* (2024).

[44] Fangyuan Zhang, Zhengjun Huang, Yingli Zhou, Qintian Guo, Zhixun Li, Wensheng Luo, Di Jiang, Yixiang Fang, and Xiaofang Zhou. 2025. EraRAG: Efficient and Incremental Retrieval Augmented Generation for Growing Corpora. *arXiv preprint arXiv:2506.20963* (2025).

[45] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. 2024. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. *arXiv preprint arXiv:2410.02506* (2024).

[46] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Junnan Dong, Hao Chen, Yi Chang, and Xiao Huang. 2025. A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models. *arXiv preprint arXiv:2501.13958* (2025).

[47] Yang Zhang, Hanlei Jin, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901* (2024).

[48] Shengming Zhao, Yuheng Huang, Jiayang Song, Zhijie Wang, Chengcheng Wan, and Lei Ma. 2024. Towards understanding retrieval accuracy and prompt quality in rag systems. *arXiv preprint arXiv:2411.19463* (2024).

[49] Siyun Zhao, Yuqing Yang, Zilong Wang, Zhiyuan He, Luna K Qiu, and Lili Qiu. 2024. Retrieval augmented generation (rag) and beyond: A comprehensive

survey on how to make your llms use external data more wisely. *arXiv preprint arXiv:2409.14924* (2024).

[50] XUJIANG ZHAO, JIAYING LU, CHENGYUAN DENG, C ZHENG, JUNXIANG WANG, TANMOY CHOWDHURY, L YUN, HEJIE CUI, ZHANG XUCHAO, TIANJIAO ZHAO, et al. 2023. Beyond One-Model-Fits-All: A Survey of Domain Specialization for Large Language Models. *arXiv preprint arXiv* 2305 (2023).

[51] Yujia Zhou, Yan Liu, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Zheng Liu, Chaozhuo Li, Zhicheng Dou, Tsung-Yi Ho, and Philip S Yu. 2024. Trustworthiness in retrieval-augmented generation systems: A survey. *arXiv preprint arXiv:2409.10102* (2024).

[52] Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, et al. 2025. In-depth Analysis of Graph-based RAG in a Unified Framework. *arXiv preprint arXiv:2503.04338* (2025).