ROGRAG: A Robustly Optimized GraphRAG Framework

Zhefan Wang[1] **Huanjun Kong**[1†] **Jie Ying**[1] **Wanli Ouyang**[1,3] **Nanqing Dong**[1,2,*]
[1]Shanghai Artificial Intelligence Laboratory [2]Shanghai Innovation Institute
[3]Department of Information Engineering, Chinese University of Hong Kong

Abstract

Large language models (LLMs) commonly struggle with specialized or emerging topics which are rarely seen in the training corpus. Graph-based retrieval-augmented generation (GraphRAG) addresses this by structuring domain knowledge as a graph for dynamic retrieval. However, existing pipelines involve complex engineering workflows, making it difficult to isolate the impact of individual components. It is also challenging to evaluate the retrieval effectiveness due to the overlap between the pretraining and evaluation datasets. In this work, we introduce **ROGRAG**, a **R**obustly **O**ptimized **G**raph**RAG** framework. Specifically, we propose a multi-stage retrieval mechanism that integrates dual-level with logic form retrieval methods to improve retrieval robustness without increasing computational cost. To further refine the system, we incorporate various result verification methods and adopt an incremental database construction approach. Through extensive ablation experiments, we rigorously assess the effectiveness of each component. Our implementation includes comparative experiments on SeedBench, where Qwen2.5-7B-Instruct initially underperformed. ROGRAG significantly improves the score from 60.0% to 75.0% and outperforms mainstream methods. Experiments on domain-specific datasets reveal that dual-level retrieval enhances fuzzy matching, while logic form retrieval improves structured reasoning, highlighting the importance of multi-stage retrieval. ROGRAG is released as an open-source resource[1] and supports installation with `pip`.

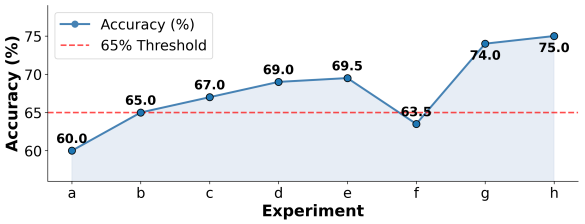

Figure 1: Performance improvements with each experiment – (a) Our initial system, (b) Remove abundant zero-shot example during retrieval, (c) Revert LLM *rope_scaling* default value, (d) Use 8k length for *nodes* and *edges*, 12k length for *chunks*, (e) Expand low-level keys, (f) Exact matching method, (g) Revert to dual-level method and optimize NER prompt, (h) Fuse logic form retrieval with pre-check.

1 Introduction

The rapid advancement of LLMs has significantly enhanced natural language processing (NLP) tasks (Min et al., 2023). However, their reliance on finite training data and static pre-trained knowledge limits their effectiveness in knowledge-intensive applications, such as question answering (QA) and complex reasoning (Roberts et al., 2020). Retrieval-augmented generation (RAG) addresses these limitations by integrating information retrieval with language generation, improving factual accuracy and adaptability (Lewis et al., 2020).

Traditional RAG methods typically rely on dense retrieval (Karpukhin et al., 2020) or keyword-based matching to obtain relevant information in response to user queries. While effective for many tasks, these approaches often struggle with complex reasoning tasks that require understanding relationships between entities or synthesizing multi-hop knowledge. To overcome these limitations, recent research has explored GraphRAG, which incorporates structured knowledge representations such as knowledge graphs to enhance both retrieval accuracy and reasoning capability (Guo et al., 2024; Liang et al., 2024). By explicitly modeling entities and their relations, GraphRAG improves retrieval precision and facilitates structured reasoning.

Despite its potential, the development and evaluation of GraphRAG systems present several chal-

*Corresponding author.
†Project lead.
[1]https://github.com/tpoisonooo/ROGRAG

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 604–613
July 27 - August 1, 2025

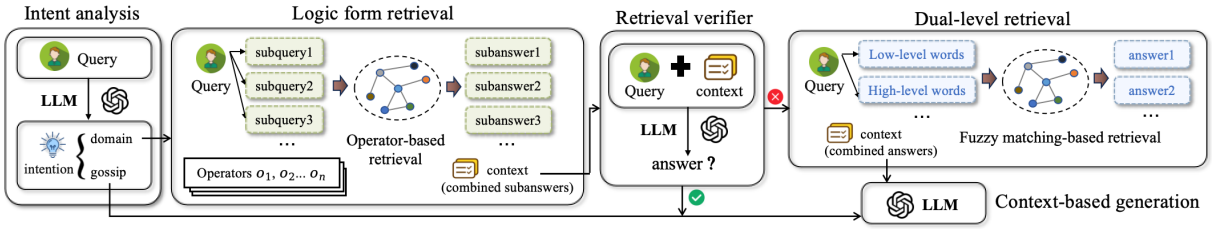

Figure 2: Multi-stage retrieval mechanism. User queries are first analyzed by an LLM to identify their intent and domain. The system then performs two retrieval strategies: logic form retrieval based on operator reasoning, and dual-level retrieval leveraging fuzzy matching. A verifier determines whether the retrieved context sufficiently answers the query. The final answer is generated by the LLM based on verified context.

Domain	LLM win rate	Length ratio
Argriculture	0.97	9.25
Art	0.91	8.20
Biography	0.82	7.69
Cooking	0.88	7.70
Computer Science	0.97	10.77
Fiction	0.65	6.63
Finance	0.85	9.33
Health	0.93	8.97

Table 1: Validation of the use of the RAG benchmark in training models. The questions from the UltraDomain (Qian et al., 2024) dataset are first executed using Qwen2.5-7B-Instruct (Qwen et al., 2025). The Kimi API[2] is then queried to compare the LLM's responses with the ground truth (GT) and determine its preference. Using these preferences, the LLM win rate is calculated, along with the average token length ratio between LLM's responses and the GT. The results show that the direct responses from the 7B model significantly outperform GT, highlighting the difficulty of validating the RAG system's effectiveness on such datasets.

lenges. First, these pipelines typically consist of multiple interdependent components—including entity extraction, knowledge graph construction, query decomposition, retrieval mechanisms, and response generation (Lewis et al., 2020)—making it difficult to assess the contribution of each individual module. Second, the widespread use of publicly available RAG benchmarks in LLM pretraining corpora complicates evaluation. As demonstrated in Table 1, we verify that high performance may result from model memorization rather than true retrieval capability (Lewis et al., 2021). Finally, many GraphRAG systems rely on heuristic-driven query decomposition, which may introduce errors if LLMs fail to generate accurate sub-queries, thereby degrading retrieval quality.

[2]See https://platform.moonshot.cn for Kimi API.

To systematically address these challenges, we introduce ROGRAG, an **R**obustly **O**ptimized **G**raph**RAG** framework designed for knowledge-intensive domains where the performance of *vanilla* LLM remains suboptimal. Our method yields a substantial performance improvement, increasing the score from 60.0% to 75.0%, as illustrated in Figure 1. We improve the accuracy of the GraphRAG system by integrating and refining four seminal GraphRAG methodologies: DB-GPT (Xue et al., 2023) (scalability), LightRAG (Guo et al., 2024) (simple implementation), KAG (Liang et al., 2024) (reasoning), and HuixiangDou (Kong et al., 2024) (robustness), into a unified system. This integration not only leverages the advantages from different GraphRAG implementations, but also enables a comprehensive ablation study on the contributions of different indexing, retrieval, and generation strategies. The system prioritizes query decomposition, and degrades to fuzzy matching if decomposition fails or verification is unsuccessful, as shown in Figure 2. This mechanism ensures robustness and enables continuous streaming responses. ROGRAG also retrains key components from the previous generation, including refusal-to-answer and intent slots. Additionally, we adopt domain-specific datasets where LLMs exhibit lower baseline scores to ensure that observed improvements reflect genuine retrieval enhancements rather than model memorization effects. Our main contributions are as follows:

- **Unified GraphRAG Framework:** We merge multiple leading GraphRAG implementations into a single, extensible pipeline for structured retrieval and reasoning. Additionally, we introduce incremental database construction to dynamically expand and refine knowledge graphs.
- **Enhanced Retrieval Mechanism:** We evaluate and refine retrieval techniques, including dual-

level query decomposition, logic form retrieval, and fuzzy matching, with various result verification to improve accuracy and adaptability.

- **Rigorous Empirical Evaluation:** Through comprehensive ablation studies on datasets where LLMs do not achieve trivial success, we provide insights into the key factors that contribute to performance improvements in GraphRAG-based QA systems. Additionally, the system is successfully deployed on the platform for user access.

2 Related Work

2.1 Retrieval-Augmented Generation

RAG enhances LLMs by integrating external knowledge retrieval to improve factual accuracy and contextual relevance (Gao et al., 2023; Fan et al., 2024). The standard RAG pipeline (Lewis et al., 2020) consists of three key components: indexing, retrieval, and generation. Retrieval methods typically rely on semantic similarity (Khattab and Zaharia, 2020) to identify relevant knowledge from external sources. Recent advancements in RAG have focused on mitigating hallucinations and improving generation quality. For instance, RETRO (Borgeaud et al., 2022) employs large-scale retrieval during both training and inference, while Lift-RAG (Cheng et al., 2024) introduces self-memory mechanisms to leverage generated content for subsequent retrieval. However, traditional RAG models struggle with structured data, as they primarily rely on single-document retrieval and fail to capture complex multi-hop relationships.

2.2 Graph Retrieval-Augmented Generation

To address the limitations of conventional RAG, GraphRAG integrates graph-based structures, including knowledge graphs like Freebase (Bollacker et al., 2008) and Wikidata (Vrandečić and Krötzsch, 2014), into the retrieval process. By leveraging entity-relationship graphs, GraphRAG provides richer contextual information to enhance both retrieval and generation tasks. Since its introduction (Edge et al., 2024), researchers have explored how integrating graph-structured data improves the model's ability to capture complex dependencies (Han et al., 2024). Meanwhile, a systematic analysis of the application of GraphRAG in customizing LLMs has also been conducted (Zhang et al., 2025).

Despite these significant advancements, existing GraphRAG models still face challenges in balancing retrieval accuracy, computational efficiency, and adaptability to diverse query structures (Peng et al., 2024). Our work builds upon these foundations by refining GraphRAG techniques to improve retrieval precision and response coherence while maintaining efficient knowledge integration.

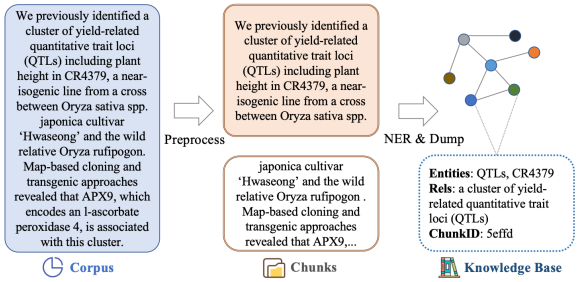

Figure 3: Architecture of GraphRAG indexing. The raw corpus is first cleaned and segmented into manageable chunks. Entities, relationships, keywords and descriptions are then extracted from each chunk. Subsequently, graph nodes and edges are constructed and linked back to their corresponding text chunks.

3 Methodology

In this section, we describe the components of GraphRAG, including indexing, retrieval, and generation, as well as the key methodological choices. A comprehensive algorithmic overview can be found in Algorithm 1 in Appendix A.1.

3.1 Graph-Based Indexing

The indexing process, represented by f in Algorithm 1, consists of the preprocess, named entity recognition (NER), and dump stages. An overview of this indexing workflow is shown in Figure 3.

Preprocess. Corpus files from heterogeneous sources are standardized and segmented into discrete text chunks to ensure structural uniformity.

NER. $\langle \text{entity}_s, \text{relation}, \text{entity}_o \rangle$ triplets are initially extracted from segmented text, along with corresponding keywords, description and weight (*e.g.*, $\langle$Marie Curie, discovered, radium$\rangle$; scientist, discovery; Marie Curie discovered radium in 1898; 4.5). Subsequently, a graph is constructed by establishing node-edge relationships to capture complex multi-hop dependencies across the corpus.

Dump. The extracted entities, relations, and their corresponding embeddings are stored in a structured database for efficient retrieval and analysis.

3.2 Graph-Guided Retrieval

The retrieval phase is governed by both g and the iterative selection in Algorithm 1. There are

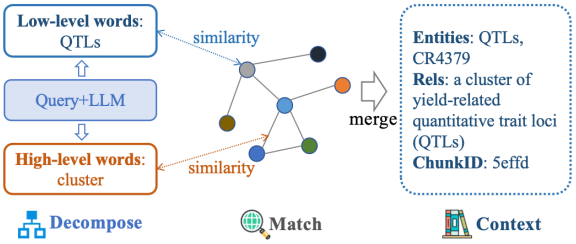

Figure 4: Dual-level retrieval method. User query would be decomposed into low-level and high-level keywords, then match with the knowledge graph.

two main methods: dual-level and logic form.

Dual-Level Method. As illustrated in Figure 4, the query is decomposed into two components: (i) low-level keywords representing entities and (ii) high-level relational descriptions. Entities are identified and matched to corresponding nodes in the graph, often using fuzzy matching, and their associated edges are subsequently retrieved. Similarly, relational keywords are mapped to edges to retrieve connected nodes. The retrieved results are then merged, with redundant edges, nodes, and chunk references systematically removed to refine the final retrieval context. This approach leverages multi-granularity features for layered fuzzy matching, improving retrieval coverage on ill-formed or complex queries and enhancing robustness.

Logic Form Method. Inspired by knowledge-aware reasoning frameworks, this approach utilizes a predefined set of operators (*e.g.*, filtering, aggregation) to decompose complex queries. An LLM is employed to transform natural language queries into a structured sequence of retrieval operations, which are iteratively refined to enhance the retrieval context. The pseudocode for this approach is presented in Algorithm 2 in Appendix A.2.

3.3 Graph-Enhanced Generation

During the generation stage, most RAG architectures leverage LLMs to (i) format and present the retrieved context as part of a prompt, (ii) produce the final response, and (iii) evaluate or verify whether the generated output correctly addresses the query. Techniques such as input formatting, prompt engineering, and self-consistency checks are often employed to optimize these generations.

4 Experiment

In this section, we describe the experimental setup and overall result of our GraphRAG system.

Method	SeedBench Subsets			
	QA-1 (Accuracy)	QA-2 (F1)	QA-3 (Rouge)	QA-4 (Rouge)
vanilla (**w/o RAG**)	0.57	0.71	0.16	0.35
LangChain	0.68	0.68	0.15	0.04
BM25	0.65	0.69	0.23	0.03
RQ-RAG	0.59	0.62	0.17	0.33
ROGRAG (Ours)	**0.75**	**0.79**	**0.36**	**0.38**

Table 2: A comparison of the test scores between several mainstream RAG systems and our proposed system. The experiment is conducted using Qwen2.5-7B-Instruct on SeedBench, while the BM25 (Jin et al., 2024) and RQ-RAG (Chan et al., 2024) methods are implemented based on FlashRAG (Jin et al., 2024). The GraphRAG framework, ROGRAG, which we proposed, outperforms all other methods across all four subsets of SeedBench, demonstrating the effectiveness of the subsequent optimizations.

4.1 Experimental Setup

Methods and Models. We adopt techniques from HuixiangDou (Kong et al., 2024) regarding the refusal-to-answer task. To switch between knowledge bases, we use TuGraph (TuGraph, 2023) for knowledge graph storage and BCEmbedding (NetEase Youdao, 2023) to extract features for entities and relations due to its effectiveness in generating high-quality embeddings. The extracted features are subsequently indexed in Faiss (Douze et al., 2024). To ensure that our experiments require minimal computational resources, we conduct tests on Qwen2.5-7B-Instruct, a relatively lightweight model that offers a good balance of efficiency and capability. Appendix B provides further details.

Data and Evaluation. We adopt the SeedBench (Ying et al., 2025) dataset, a domain-specific benchmark curated by human experts to evaluate systems. In subsequent ablation studies, we applied multiple-choice questions from QA-1 subset of SeedBench, with accuracy as evaluation metric.

4.2 Overall Result

Table 2 compares the test scores of several mainstream RAG systems and our proposed system. Initially, we evaluate the large model's direct response without RAG (*vanilla*) as baselines and then combine the four foundational GraphRAG methodologies to construct our initial system. Next, we conduct detailed ablation experiments on different components of the system, comparing various methods and parameters to improve each component, thereby progressively enhancing the test out-

Version	Nodes	Edges	Accuracy
Trial	20,739	19,857	0.61
Base	21,838	26,847	0.69
Optimize Prompt	29,086	35,750	0.74

Table 3: Number of nodes and edges generated by different Loop NER Strategies and the methods' impact on accuracy. Increasing the quantity can improve accuracy.

Max Length	Accuracy
32k	0.67
64k	0.65

Table 4: Impact of different maximum context lengths of LLM on accuracy. When the model's maximum length is sufficient, a smaller $rope\_scaling$ is preferred.

comes. Finally, we introduce ROGRAG and compare it with three mainstream RAG systems, including inverted indexing, similarity-based retrieval, and multi-round answering techniques.

Our experiments show that ROGRAG outperforms all other systems and the baseline across all four subsets of SeedBench. Additionally, it also proves that Qwen2.5-7B-Instruct is initially unsuitable for these tasks and subsequent optimizations are effective. Interestingly, on the QA-2 dataset, the non-RAG approach even outperforms mainstream methods. This suggests that RAG responses could be misled by irrelevant context and applying RAG does not always lead to higher accuracy, particularly in domains where LLMs lack familiarity. The poor performance of LangChain (Chase, 2022) and BM25 (Jin et al., 2024) methods on the QA-4 generation task may be due to parameter settings that resulted in minimal relevant content being retrieved, limiting the model's ability to generate accurate answers.

5 Indexing Analysis

In this section, we begin by introducing different NER strategies and then proceed to validate the parameters associated with indexing.

5.1 Loop NER Strategies

To maximize the recall of the NER, GraphRAG tends to iteratively call the LLM in order to extract as many entities as possible. The common stopping condition is based on the LLM's judgment of whether any entities have been missed. We compare two implementations of iterative NER, as shown in Algorithm 3 in Appendix A.3. The trial version (trial) follows a standard procedural logic: it performs NER first, then queries the LLM whether further extraction is needed, and proceeds only if the response is affirmative. In contrast, Baseline version (base) deviates from this flow. After performing NER, it informs the LLM that more entities may exist and requests further extraction before entering the if-condition.

As shown in Table 3, the larger nodes and edges in the knowledge graph is positively correlated with higher precision, and the erroneous entities generated by LLM have minimal impact on the overall graph structure and final accuracy. Therefore, we can optimize the prompt by specifying entity types and splitting examples to improve accuracy.

5.2 Max LLM Context Length

While LLMs typically perform well in needle-in-a-haystack experiments, RAG systems often need to focus more on subtle and implicit expressions within the corpus. To extend the maximum input length in YaRN (Peng et al., 2023), we modify the parameter $rope\_scaling$, which modifies the rotary position embedding (RoPE) scaling strategy in order to accommodate longer contexts. Our empirical results show that increasing the length from 32k to 64k leads to a noticeable 2% drop in precision. Therefore, a smaller $rope\_scaling$ setting is generally preferable, as shown in Table 4. Similarly, reducing *chunk size* may lead to better results, as it allows for more accurate information extraction.

6 Retrieval & Generation Analysis

In this section, we first validate the parameters that impact performance, followed by a comparison of different retrieval and verification methods.

6.1 Representation and Matching Granularity

Based on the previous conclusions, we hypothesize that the essence of the dual-level method lies in approximate matching. The richer the representations derived from the corpus and query, the higher the overall precision. To validate this, we conducted experiments in two opposite directions.

Extended Queries and Low-Level Keys. Since LLMs often struggle with domain-specific data (*e.g.*, mistaking entities for relationships), we expand the maximum length of the query representation (R_q in Algorithm 1) and increase the number

Method	Accuracy
Dual-level	0.650
+28k context length	0.690
+expand low-level keys	0.695
Exact matching method	0.635

Table 5: Validations of dual-level retrieval method in two opposite directions. It is hypothesized that increasing the length of R_q and R_c will lead to improved accuracy, and the results support this hypothesis.

Method	Avg length	Accuracy
Optimized dual-level	9863	0.74
Logic form	1699	0.55

Table 6: Average output context length of dual-level and logic form and methods' impact on accuracy. Although the logic form retrieval method shows suboptimal precision, it provides higher information density.

of low-level keys. This approach aims to allow the query to incorporate more detailed information, thereby enhancing the accuracy of matching.

Exact Matching. Instead of concatenating the entity list, we independently store the features of each entity during the indexing phase and match individual entities during query time. This method aims to improve precision by avoiding the noise typically introduced by approximate matching.

Our hypotheses are validated in Table 5. Increasing the maximum length of R_q to 28k results in a 4% improvement in precision compared to baseline. Expanding the number of low-level keys helps fix some rare bad cases. In contrast, switching to exact matching leads to a decrease in precision. This suggests that fuzzy matching is essential for capturing the nuances of the query, as exact matching fails to account for the implicit keywords derived from the query. This aligns with common sense, as exact matching is too rigid for complex queries.

6.2 Dual-Level vs. Logic Form

We compare the results of dual-level and logic form methods in Table 6. Although the dual-level method achieves higher precision, it fails to provide convincing answers to questions that require calculations (*e.g.*, "How much taller is Zhefu 802 than its parent?"). In our real-world scenario evaluations, responses generated by the logic form method are more concise and exhibit a clearer logical progression, preferred by domain experts.

Method	Accuracy
Argument Checking	0.75
Result Checking	0.72

Table 7: Performance comparison of checking strategies on logic form retrieval method. The results indicate that the argument checking yields better performance.

6.3 Argument Checking vs. Result Checking

RAG systems often employ LLM to verify the correctness of results. We compare two approaches: argument checking and result checking. The argument checking verifies whether the provided context can answer the question before generating the final response, while the result checking examines the question, context, and response together for overall coherence. The pseudocode is shown in Algorithm 4 in Appendix A.4. The results are summarized in Table 7, which shows that argument checking is preferred. From the aspect of inference, the response diverts part of the LLM's attention, thereby reducing its ability to focus on the core question. From the aspect of model, Qwen-2.5-7B-Instruct is a causal model, where correct reasoning within the context typically leads to correct results. Therefore, result checking is redundant.

7 Conclusion

In this work, we introduce ROGRAG, a robustly optimized GraphRAG framework that addresses the limitations of existing retrieval-augmented generation pipelines in handling specialized and domain-specific queries. By combining dual-level and logic form methods in a multistage retrieval process, ROGRAG enhances retrieval robustness. The system is further enhanced with result verification methods and an incremental knowledge graph construction strategy. The empirical studies show that the logic form method, with its step-by-step approach, is more acceptable by domain experts. This method aligns well with human reasoning and provides clear and logical answers that are easy to interpret and validate. Lastly, future work shall focus on building a high-accuracy verifier, refining LLM decomposition steps, and exploring further enhancements to improve overall performance. For a more detailed discussion, see Appendix C.

Limitations

Due to the large workload, we have only used Qwen2.5-7B-Instruct to conduct experiments on a single domain-specific dataset for the time being. In the future, we will explore the application of ROGRAG to more general models, test its performance on a broader range of domain-specific datasets, and enhance its robustness. However, developing the GraphRAG system presents several challenges, such as the difficulty of constructing an effective high-accuracy verifier, which is essential for further improving precision. On the other hand, due to the inherent limitations of large models and the noisy or incomplete corpus provided, errors in entity extraction, query decomposition, or knowledge retrieval within the GraphRAG method may propagate throughout the system, exacerbating inaccuracies in response generation. In addition, heuristic-driven query decomposition remains a potential bottleneck, as errors in subquery formation may degrade retrieval performance.

Acknowledgments

This work was supported by Shanghai Artificial Intelligence Laboratory. The authors would like to thank Chenyang Wang from Huawei Technologies Co., Ltd and Zhe Ma from Shanghai Artificial Intelligence Laboratory for academic support.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240. PMLR.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Harrison Chase. 2022. LangChain.

Xin Cheng, Di Luo, Xiuying Chen, Lemao Liu, Dongyan Zhao, and Rui Yan. 2024. Lift yourself up: Retrieval-augmented text generation with self-memory. *Advances in Neural Information Processing Systems*, 36.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

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*, pages 6491–6501.

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

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation.

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*.

Jiajie Jin, Yutao Zhu, Xinyu Yang, Chenghao Zhang, and Zhicheng Dou. 2024. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. *CoRR*, abs/2405.13576.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Huanjun Kong, Songyang Zhang, Jiaying Li, Min Xiao, Jun Xu, and Kai Chen. 2024. Huixiangdou: Overcoming group chat scenarios with llm-based technical assistance.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation

for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115.

Lei Liang, Mengshu Sun, Zhengke Gui, Zhongshu Zhu, Zhouyu Jiang, Ling Zhong, Yuan Qu, Peilong Zhao, Zhongpu Bo, Jin Yang, Huaidong Xiong, Lin Yuan, Jun Xu, Zaoyang Wang, Zhiqiang Zhang, Wen Zhang, Huajun Chen, Wenguang Chen, and Jun Zhou. 2024. Kag: Boosting llms in professional domains via knowledge augmented generation.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

Inc. NetEase Youdao. 2023. Bcembedding: Bilingual and crosslingual embedding for rag. `https://github.com/netease-youdao/BCEmbedding`.

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*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

TuGraph. 2023. Tugraph: A high performance graph database.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Siqiao Xue, Caigao Jiang, Wenhui Shi, Fangyin Cheng, Keting Chen, Hongjun Yang, Zhiping Zhang, Jianshan He, Hongyang Zhang, Ganglin Wei, et al. 2023. Db-gpt: Empowering database interactions with private large language models. *arXiv preprint arXiv:2312.17449*.

Jie Ying, Zihong Chen, Zhefan Wang, Wanli Jiang, Chenyang Wang, Zhonghang Yuan, Haoyang Su, Huanjun Kong, Fan Yang, and Nanqing Dong. 2025. Seedbench: A multi-task benchmark for evaluating large language models in seed science. In *Proceedings of the 63nd Annual Meeting of the Association for Computational Linguistics*, Vienna, Austria. Association for Computational Linguistics.

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*.

A Additional Details on Methodology

A.1 GraphRAG

Let the corpus be denoted by C and the user query by Q. The GraphRAG framework, defined as $\mathsf{graphrag} = (f, g, d)$, consists of three primary functions:

- **Indexing function** f, which extracts structured representations from C, yielding R_c.
- **Retrieval function** g, which derives representations from Q, producing R_q.
- **Graph-based augmentation function** d, which iteratively constructs paths linking R_c and R_q, refining the retrieval context.

Algorithm 1 GraphRAG

Compute corpus representations: $R_c = f(C)$
Compute query representations: $R_q = g(Q)$
Initialize the retrieval set: $S_0 \subseteq R_c$
for $k = 1, 2, \ldots$ **do**
    Select the most relevant element: $e_k^+ = \arg\min_{e \in R_c \setminus S_{k-1}} \text{dist}(S_{k-1} \cup \{e\}, R_q)$
    Remove the least relevant element: $e_k^- = \arg\min_{e \in S_{k-1}} \text{dist}(S_{k-1} \setminus \{e\}, R_q)$
    Update retrieval set: $S_k = S_{k-1} \cup \{e_k^+\} \setminus \{e_k^-\}$
    if $\text{dist}(S_k, R_q)$ does not improve **then**
        Terminate retrieval process
    end if
end for

The indexing and retrieval processes are formalized in Algorithm 1. At each iteration, an element e_k^+ is selected for inclusion if it minimizes the retrieval distance $\text{dist}(S_{k-1}, R_q)$, while an element e_k^- is removed if it similarly optimizes the retrieved set. The stopping criterion ensures termination when no further improvements can be achieved.

A.2 Logic Form Retrieval

Logic Form method. Inspired by knowledge-aware reasoning frameworks, this approach employs a predefined set of operators (e.g., filtering, aggregation) to decompose complex queries.

A.3 Loop NER

Two implementations of iterative NER, including a trial version and a base version.

Algorithm 2 Logic Form Retrieval

Input: Operator set $O = \{o_1, o_2, \ldots, o_n\}$, where each $o_i = (\text{operator}_i, \text{function}_i)$
Input: User query Q
Output: History
Decompose query Q into a list of subqueries L using LLM
$L \leftarrow \{(q_1, a_1), (q_2, a_2), \ldots, (q_m, a_m)\}$, where each $a_j \in O$
for $(q_j, a_j) \in L$ **do**
    Identify the corresponding operator o_j for a_j
    Execute $a_j \leftarrow o_j(q_j)$
end for
Concatenate all sub-queries and sub-answers a_j into history
return History

Algorithm 3 Loop NER

Initialize input text T
Initialize maximum attempts MAX
Initialize history $H \leftarrow NER\_init(T)$

function TRIAL
    for $i = 0$ **to** MAX **do**
        $continue \leftarrow LLM\_judge(H)$
        if $continue ==$ "no" **then**
            break
        end if
        $result \leftarrow NER\_continue(T, H)$
        $H \leftarrow H \cup result$
    end for
end function

function BASE
    for $i = 0$ **to** MAX **do**
        $result \leftarrow NER\_continue(T, H)$
        $H \leftarrow H \cup result$
        $continue \leftarrow LLM\_judge(H)$
        if $continue ==$ "no" **then**
            break
        end if
    end for
end function

A.4 Retrieval Verifier

Two implementations of iterative retrieval verifier, including two methods: pre-check and post-check.

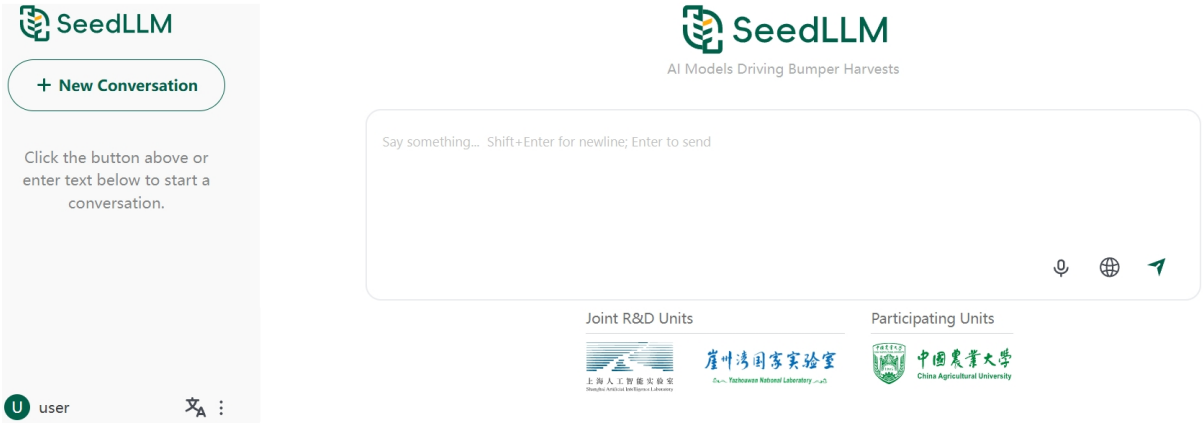

Figure 5: User interface of the deployment platform. The system enables natural language interaction for agricultural knowledge retrieval and question answering.

Algorithm 4 Retrieval Verifier

```
Initialize:
context ← logic_form_retrieve()

function ARGUMENT CHECKING
    if LLM_judge(query, context)
      == "support" then
        return LLM(query, context)
    end if
end function

function RESULT CHECKING
    reply ← LLM(query, context)
    if LLM_judge(query, context, reply)
      == "support" then
        return reply
    end if
end function
```

B Detailed Experimental Setup

We adopt techniques from (Kong et al., 2024) regarding the refusal-to-answer task. Specifically, for text splitting, we employ `ChineseRecursiveTextSplitter`[3] for Chinese text, which takes into account both maximum length and punctuation positions. For English text, we use `RecursiveCharacterTextSplitter`[5] with a chunk overlap of 32. In both cases, the default chunk size was set to 768 tokens.

C Additional Discussion on Experiments

Our experiments highlight several key factors that affect the performance of the GraphRAG system. Our analysis of iterative NER methods shows that increasing the number of extracted entities can improve accuracy, as erroneous entities will become isolated nodes and will not significantly affect retrieval accuracy. When evaluating the LLM context length, a smaller $rope\_scaling$ yields better performance when the maximum length of the model is sufficient. A larger context length (64k) leads to a slight decrease in accuracy, possibly because the increased volume of information makes it harder to extract meaningful entities and relations. In retrieval and generation analysis, we find that expanding the query representation and incorporating more low-level keys improves accuracy while switching to exact matching leads to performance degradation. This result suggests that fuzzy matching is critical to capturing subtle query semantics, as strict exact matching cannot account for implicit query variations. Comparing retrieval methods, we observe that while the dual-level method achieves higher accuracy, it lacks the ability to provide convincing reasoning on real queries. In contrast, the logic form method provides higher information density and is more concise and clear. Finally, in the validator design, we find that argument checking is more effective than result checking.

D Application

ROGRAG is deployed on an online research platform SeedLLM[4], as illustrated in Figure 5.

[3] https://github.com/chatchat-space/Langchain-Chatchat

[4] https://seedllm.org.cn