Proceedings of CONF-SEML 2025 Symposium: Machine Learning Theory and Applications
DOI: 10.54254/2755-2721/154/2025.TJ23129

PromptCraft-RAG: Context-Based Prompt Enhancement of Refining Query for Retrieval Augmented Generation

Qinye Zhang

Department of Computing, Faculty of Engineering, Hong Kong Polytechnic University, Hong Kong, China
qinye.zhang@connect.polyu.hk

Abstract: Retrieval-augmented generation (RAG) has become a transformative framework in Natural Language Processing (NLP). It contributes to the generation process by retrieving relevant information from external knowledge bases, thus making the responses more accurate and contextualized. Recent developments in RAG have renewed interest in optimizing RAG frameworks, such as improving the efficiency of the retrieval module, query reconstruction, refinement, ranking mechanisms, and resolution of hallucinations. However, RAG still faces significant bottlenecks, especially when it comes to understanding knowledge sources and user prompts. Prompt enhancement is a critical but not explored aspect of the RAG field. Inspired by Refine Query for Retrieval Augmented Generation (RQ-RAG), this paper proposes prompt pruning and prompt enhancement as innovative solutions to improve prompts iteratively for better retrieval and generated results. It also attempts to expand the dataset to cover multiple tasks in various scenarios. Experimental comparisons between RQ-RAG and PromptCraft-RAG (PC-RAG) show competitive performance: RQ-RAG achieves 68.3 (single-hop Question Answering (QA)) and 49.7 (multi-hop QA), while PC-RAG scores 68.1 and 51.7, respectively.

Keywords: Retrieval- augmented Generation (RAG), Refine Query for Retrieval Augmented Generation (RQ-RAG), Prompt Enhancement.

1. Introduction

Retrieval-augmented generation (RAG) performs dynamic knowledge retrieval and generative processes, enabling systems to deal with complex and knowledge-intensive tasks [1]. Despite this, its overall performance can be severely impacted if the input prompts are of poor quality. These prompts might be ambiguous, too short, or lack sufficient context, which can lead to retrieval failures, hallucinations, or output results that do not match the user's intent [2]. These issues reflect a significant challenge facing current RAG systems: the lack of a mechanism for iteratively refining prompts and making the retrieval and generation process consistent.

Much work has been striving to improve retrieval strategies or generator architectures. For example, the foundational RAG frameworks set up the retriever-generator paradigm, and later efforts like Refine Query for Retrieval Augmented Generation (RQ-RAG) introduced query refinement techniques to improve retrieval relevance [3, 4]. However, these methods mainly use the prompts as static inputs but don’t consider the enhancement of the prompts. Slight changes to the prompts, such as disambiguation or decomposition, can substantially impact the retrieval results [5]. Despite this, in

DOI: 10.54254/2755-2721/154/2025.TJ23129

real-world applications, existing systems often cannot do refinement tasks, especially when there are multimodal inputs and prompt lengths are restricted.

This work addresses these challenges by introducing context-based prompt enhancement, which dynamically improves user queries to enhance retrieval and generation. Building on the principles of query refinement proposed by RQ-RAG, thesis propose PromptCraft-RAG (PC-RAG), which has three crucial innovations: expanding the dataset with queries from different scenarios, applying auto-merging techniques to produce more structured and semantically meaningful truncated text; and using relation-entity extraction techniques to reduce computational costs. This work aims to increase the consistency between user queries and system responses by focusing on these areas, making the RAG system more adaptable to various real-world tasks. The rest of the paper has been divided into four parts. Section 2 reviews the dataset constructed for this experiment, and Section 3 lays out the theoretical dimensions of the research, which is concerned with the methodology. Section 4 presents the experimental results and analysis. Finally, Section 5 gives a summary and a discussion of possible future directions.

2. Dataset

2.1. Dataset used in RQ-RAG

The dataset for training and evaluating PC-RAG was designed to handle various query scenarios [6, 7]. It contains single-hop question answering (QA), multi-hop QA, ambiguous queries, and instruction-following tasks. For single-hop QA tasks, the dataset has three distinct components. The AI2 Reasoning Challenge (ARC-C) dataset includes 1,172 multiple-choice questions that assess commonsense reasoning. open-domain QA (PopQA) provides a long-tail subset of 1,399 instances, focusing on factual questions. OpenbookQA consists of 500 multiple-choice questions that require answers based on a predefined set of facts. These datasets evaluate the model's ability to manage straightforward queries.

The dataset also includes multi-hop QA tasks, assessing the model's ability to integrate information from various sources to provide coherent answers. The HotpotQA dataset has questions linked to ten passages, with each question depending on two relevant documents for accurate answers. Similarly, 2WikiMultiHopQA features complex queries linked to ten passages, two of which apply to the question. MuSiQue includes questions linked to twenty candidate documents and requires two to four hops to answer.

Another essential part of the dataset focuses on ambiguous queries. The Australian Skills Quality Authority (ASQA) dataset is designed to evaluate the model's ability to handle unclear or underspecified questions. It requires the model to clarify vague queries before generating responses. Additionally, it contains collections like LIMA, WizardLM, Open-Orca, OpenAssistant, and Chat Generative Pre-Trained Transformer 4 (ChatGPT4)-Alpaca. These tasks require the model to follow complex instructions and generate structured, contextually appropriate responses.

2.2. Data collection and annotation

In this study, thesis have carefully expanded the dataset. This extensive dataset consists of 42,810 instances that help the model to effectively cope with various scenarios and query types in the experiments, including single-hop question answering, multi-hop reasoning, vague querying, and instruction following tasks. Apart from that, thesis also automated the dataset annotation process using ChatGPT (gpt-3.5-turbo-0125), which extracts raw data into a structured and usable format. It generates refined queries, searches for relevant information, and produces responses based on retrieved context. This automated flow provides consistency in dataset annotation while reducing the need for manual effort for such large-scale tasks.

DOI: 10.54254/2755-2721/154/2025.TJ23129

2.3. Dataset construction pipeline

The dataset was constructed using a clear pipeline to ensure high quality and relevance. First, thesis categorized all tasks into specific classes, such as multi-turn dialogues, query decomposition, and query disambiguation. Next, ChatGPT is employed with predefined prompt templates to refine queries to meet the requirements of the retrieval tasks. Then, relevant contexts were retrieved using DuckDuckGo and other sources. Finally, thesis used ChatGPT again to generate renewed responses based on the refined queries and retrieved contexts. This systematic approach ensured that the dataset was rich, diverse, and suitable for training and evaluating the framework.

3. Results and discussions

The key idea is to refine prompts before retrieving information by explicitly performing query-rewriting, decomposition, and disambiguation. The implementation consisted of three core components: Dataset Construction, Generator Training, and Sample Strategies. Each element was designed to align with the system's goal of refining queries and generating contextually accurate responses. PC-RAG implements innovative ideas mainly in Dataset Construction and Generator Training sessions.

3.1. Dataset construction

In RQ-RAG, training data was constructed to mirror the inference-time workflow. Generally speaking, it is significant to tune the given trivial input and output the dataset into the format of input and generate the final output after several operations. The transformation process can be denoted as:

$$(X_{origin}, Y_{origin}) \rightarrow (X_{origin}, SPECIAL_{type}, Q_i, type, [D_{i1}, \cdots, D_{ik}], \cdots, Y_{new}) \quad (1)$$

In this workflow, the original input-output pairs are represented by (X_{origin}, Y_{origin}), and i indicates the iteration turn. $SPECIAL_{type}$ refers to the special toke, denoting the actions the model needs to perform, which are represented by Type: rewrite, decompose, or disambiguate. Qi, type means the refined query generated in iteration i. D_{i1} to D_{ik} is the top-k relevant documents retrieved, and $Y_{new}$is the final answer generated in the last iteration. In one dataset, the type of all input-output pairs is deliberately selected to be of the same type. To perform dataset construction, the system first begins processing the initial question X_{origin}. The datasets are classified into three types: rewrite, decompose, and disambiguate, and the system utilizes a pre-defined prompt corresponding to each type to generate a refined query for each type. This query is then leveraged to retrieve information from external data sources. Based on the query from the previous iteration, the type, and its retrieved content, Large Language Model (LLM) is prompted to generate a new response and iteratively execute the response generation process.

However, a new technique called auto-merging has been implemented in PC-RAG for this transformation process. The first step is to truncate the whole context into smaller sections based on semantic meanings. As shown in Figure 1, this truncation process is performed recursively, from context level to paragraph level, then sentence level, and finally, word level. It should be noted that the structure built can be any random tree. Besides, the tree can be unbalanced since the number of children depends on how many sections can be divided according to semantic meanings. For any given query, the workflow starts from leaves and selects nodes with a relevancy score higher than the predefined threshold. There are no top-k restrictions, so an arbitrary number of nodes is acceptable. For each parent node, if more than half of the child nodes are regarded as relevant to the query, the system will select the parent instead of the number of child nodes. The system recursively performs the pruning process until no parent node has more than half of the "relevant" children's nodes.

DOI: 10.54254/2755-2721/154/2025.TJ23129

Eventually, $N_1, N_2, \ldots N_k$ are calculated, where N_i is the i-th node selected. N_i can be any node from root to leaf. The critical steps can be summarized into the formula:

$$(X_{origin}, Y_{origin}) \rightarrow (X_{origin}, SPECIAL_{type}, Q_i, type, [N_1, N_2 \cdots, N_k], \cdots, Y_{new}) \quad (2)$$

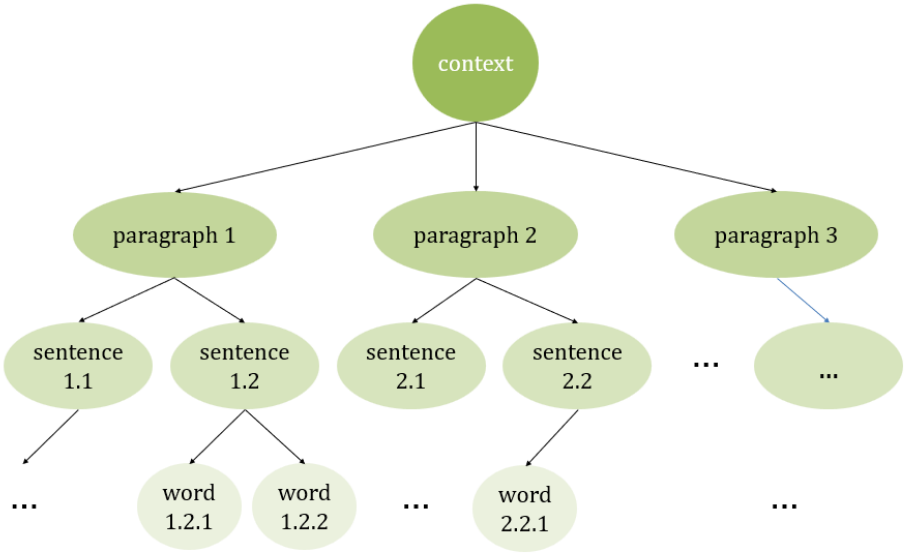

Figure 1: Auto merging example (picture credit: original)

The application of Auto-Merging techniques in Dataset Construction has many advantages. The truncated texts are more structured, making it easier for human readers to read and debug. Texts are getting more semantically meaningful and coherent, making it easier for LLM to generate answers. In the previous method utilized in RQ-RAG, sentences are naively cut in the middle, leaving behind many meaningless starting and endings. Despite the advantages over traditional methods, there also exist possible limitations. Auto-merging may add overhead to calculation; semantic division and merging children are all computational burdens. Additionally, the structure becomes more complex and, hence, more complicated to implement. However, in this case, thesis get sufficient computational resources, and the scale of the dataset is relatively small, so not much calculation is needed.

3.2. Generator training

This step aims to train the model to decide the appropriate special operations to perform and when to finish the decision tree loop, enhancing the performance in generating final answers. The model was trained auto-regressively to maximize the likelihood of generating correct responses given inputs, refined queries, and retrieved documents. In the implementation of RQ-RAG, the training objective was defined as:

$$\mathcal{L} = max\, E_{(x,y)\sim D}\left[log\, p_M\left(y|q_1, d_1, \cdots, q_i, d_i, x\right)\right] \quad (3)$$

In this formula, L represents the likelihood that thesis aim to maximize. M denotes the model parameters. The expectation $E_{(x,y)\sim D}$ averages over the dataset D, while the term $p_M(y \mid q_1, d_1, \ldots, q_i, d_i, x)$ indicates the probability of the model M generating the response y given the input x and the refined query q_i with the retrieved document di at step i. In the previous trials of RQ-RAG, the model is trained on 8 NVIDIA H800 GPUs with 80GB memory, using a learning rate of 2×10^{-5} with 3% warmup steps. The maximum input length is set to 4096 tokens to accommodate the extended context length of the dataset. During the experiment, however, although the typical

DOI: 10.54254/2755-2721/154/2025.TJ23129

required vRAM size to train Llama2-7B is well below 80 GB, the model overwhelmed the server by signaling CUDA OOM, which was allocated 100 GB vRAM and GPU clusters.

PC-RAG employs the Relation-Entity Extraction technique to tackle the issue of putting everything into training, which is relatively computationally expensive. An entity is a distinct object or concept in a given domain, such as a person, place, organization, or object, represented by a unique name or identifier (e.g., "Einstein," "Paris," "Cat"). A relation defines the connection or association between two or more entities, describing how they relate (e.g., "Einstein was born in Ulm"). Figure 2 presents a network of entities and the relations between them, aiming to give an intuitive understanding of the two concepts. To implement Relation-Entity Extraction in this system, the first procedure is to extract entities respectively from $q_1, d_1, q_2, d_2 \ldots, q_n, d_n$. The next step is to de-duplicate the entities in each pair of (q_i, d_i), reducing the entity-relation graph to its basic form. After the de-duplication, the graph will be transformed into the context C_i which is recognizable by the training model. During this process, every pair of (q_i, d_i) is transferred into C_i. After completing the entire workflow, the model training process can start. The training objective can now be denoted as:

$$\mathcal{L} = max\, E_{(x,y)\sim D}\left[log\, p_M\left(y|c_1, c_2, \cdots, c_i, x\right)\right] \tag{4}$$

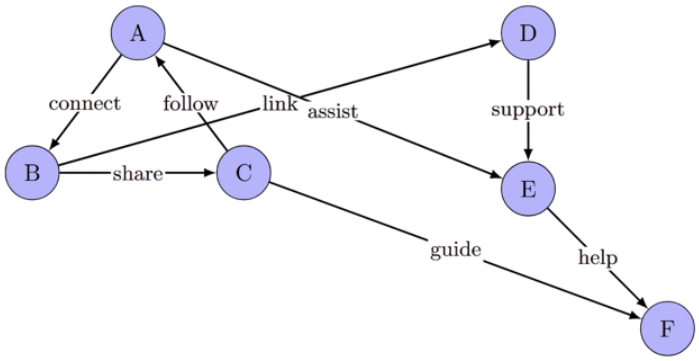

Figure 2: Relation-entity extraction (picture credit: original)

Relation-entity extraction reduces the computation burden significantly. By extracting entity-relation graphs and converting them to interpretable context, this strategy performs data enhancement to the retrieved contexts, together with query enhancement, bi-directionally fine-tuning the model. Nevertheless, the process of de-duplicating, graph construction, and graph-to-text conversion can still cause complex overheads.

4. Experimental results

This section reports the experimental results of the proposed PC-RAG model, which builds upon the RQ-RAG framework with specific optimizations. To provide context, thesis first summarize the performance of RQ-RAG as reported in its original paper. Then, thesis present the training progress and evaluation of PC-RAG on both single-hop and multi-hop QA tasks.

4.1. Performance of RQ-RAG

The experimental results in the paper "RQ-RAG: Learning to Refine Queries for Retrieval Augmented Generation" are divided into several key areas, including model training results and final outputs across various tasks [8, 9]. In single-hop QA, it was tested on datasets such as ARC-C, PopQA, and OpenbookQA. RQ-RAG surpassed LLama2-7B (Zero Shot) by an average of 33.5% in retrieval settings. It also outperformed models trained on task-specific or curated datasets without the search-

DOI: 10.54254/2755-2721/154/2025.TJ23129

augmented intermediate step. Compared to SAIL-7B, the state-of-the-art model, RQ-RAG achieved an average performance increase of 20.3% across the three datasets. Additionally, it surpassed Self-RAG, the previous state-of-the-art, by 1.9% on average, despite using significantly fewer training samples (40,000 compared to 150,000).

In multi-hop QA tasks, RQ-RAG was evaluated on datasets including HotpotQA, 2WikiMultiHopQA, and Musique. It demonstrated substantial improvements over zero-shot models and those trained on individual datasets without the search-augmented step. The model achieved an average performance improvement of 22.6% across these datasets. Furthermore, RQ-RAG outperformed robust baselines like Chain-of-Thought and Chain-of-Note, which use ChatGPT as their backbone. This achievement is noteworthy because RQ-RAG relies on a smaller backbone model compared to ChatGPT.

4.2. Model training and progress

PC-RAG was fine-tuned using the Llama2-7b-hf architecture. During training, the learning rate was gradually reduced in a linear schedule, as shown in Figure 3. The initial learning rate was set to 1.9375e-5 and progressively decreased to 0 by the final training step. This approach facilitated stable and effective parameter updates throughout the training process. The training loss, visualized in Figure 4, consistently decreased as the training progressed. Initially, the loss was 8.77 and gradually reduced to 3.01 by the final step. This steady decline demonstrates that the model successfully learned from the data and effectively minimized the error during fine-tuning.

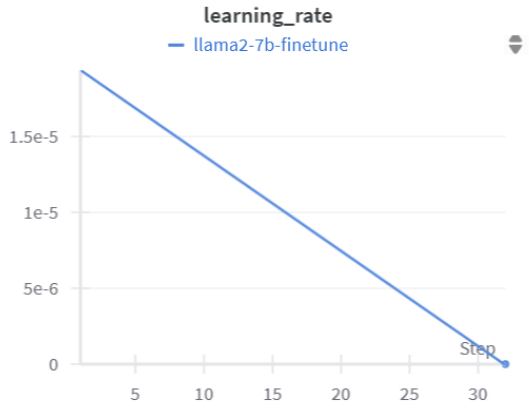

Figure 3: Changes in learning rate during training (picture credit: original)

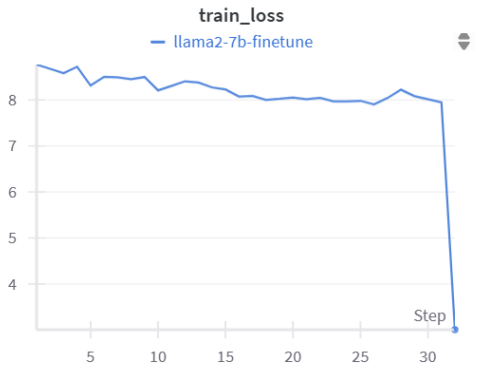

Figure 4: Losses at each step of training (picture credit: original)

DOI: 10.54254/2755-2721/154/2025.TJ23129

4.3. Performance on single-hop and multi-hop QA tasks

The performance of PC-RAG was evaluated on three single-hop QA datasets: ARC-C, POPQA, and Open Book QA (OBQA). The results are summarized Table 1. PC-RAG achieved slightly improved performance on POPQA (+1.2) compared to RQ-RAG. However, it showed a slight decrease on OBQA (-2.2). The overall average performance of PC-RAG remained close to that of RQ-RAG, indicating that the optimizations did not lead to significant changes in single-hop QA performance. However, the evaluation of multi-hop QA datasets revealed more pronounced improvements with PC-RAG. Based on the results showcased in Table 2, PC-RAG outperformed RQ-RAG across all multi-hop QA datasets. The improvements were most notable on ARC-C (+2.2) and OBQA (+2.8). The average performance of PC-RAG (51.7) exceeded that of RQ-RAG (49.7), highlighting the effectiveness of the proposed optimizations in enhancing reasoning capabilities for multi-hop questions.

Table 1: Single-hop task performance comparison

Model	ARC_C	POPQA	OBQA	AVG.
RQ-RAG	68.3	57.1	79.4	68.3
PC-RAG	68.7	58.3	77.2	68.1

Table 2: Multi-hop task performance comparison

Model	ARC_C	POPQA	OBQA	AVG.
RQ-RAG	62.6	44.8	41.7	49.7
PC-RAG	64.8	45.7	44.5	51.7

Based on the experimental results, the training progress and evaluation results indicate that PC-RAG effectively learned during fine-tuning, as evidenced by the consistent reduction in training loss. The optimizations introduced to the model had a stronger impact on multi-hop QA tasks, where reasoning complexity is higher. Single-hop QA performance remained stable, with minor variations across datasets. These results suggest that the modifications in PC-RAG enhanced its ability to tackle more challenging reasoning tasks while maintaining competitive performance on simpler ones.

5. Conclusions

This paper presents an enhanced implementation of RQ-RAG, a framework that improves LLMs by refining queries through rewriting, decomposing, and disambiguating. Three key contributions are introduced: (1) Dataset Enlargement, where the scope is expanded and annotations are automated using ChatGPT to handle diverse query scenarios. It includes single-hop QA, multi-hop reasoning, and ambiguous queries; (2) Auto-Merging, a technique that segments large contexts into semantically meaningful units and selectively recombines them to preserve critical information while maintaining structured inputs; (3) Relation-Entity Extraction, which identifies and structures entities and their relationships to enhance contextual relevance. Experimental comparisons between RQ-RAG and PC-RAG (an optimized variant) show competitive performance: RQ-RAG achieves 68.3 (single-hop QA) and 49.7 (multi-hop QA), while PC-RAG scores 68.1 and 51.7, respectively. Future work will focus on refining query trajectory selection, improving context reranking, and addressing scalability challenges for real-world applications. In addition, enhancing intent understanding and adopting broader evaluation metrics will strengthen the framework's robustness and usability.

DOI: 10.54254/2755-2721/154/2025.TJ23129

References

[1] *Rikkat, D. R., Sagar, M., & Morris, M. R. (2024). IntellBot: Retrieval augmented LLM chatbot for cyber threat knowledge delivery. arXiv print, 5442.*

[2] *Roth, K., Gupta, R., Halle, S., & Liu, B. (2024). Pairing analogy-augmented generation with procedural memory for procedural Q&A. arXiv print, 1344.*

[3] *Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in neural information processing systems, 33, 9459-9474.*

[4] *Chan, C. M., Xu, C., Yuan, R., Luo, H., Xue, W., Guo, Y., & Fu, J. (2024). Rq-rag: Learning to refine queries for retrieval augmented generation. arXiv print, 610.*

[5] *Anjum, S., Zhang, H., Zhou, W., Paek, E. J., Zhao, X., & Feng, Y. (2024). HALO: Hallucination analysis and learning optimization to empower LLMs with retrieval-augmented context for guided clinical decision making. arXiv print, 10011.*

[6] *Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2024). LightRAG: Simple and fast retrieval-augmented generation. arXiv print, 5779.*

[7] *Jin, Q., Dhingra, B., Liu, Z., Cohen, W. W., & Lu, X. (2019). Pubmedqa: A dataset for biomedical research question answering. arXiv print, 6146.*

[8] *Lin, C. Y., & Och, F. J. (2004). Looking for a few good metrics: ROUGE and its evaluation. Ntcir workshop, 1-8.*

[9] *Walker, S. M. (2023). What is the ROUGE score (Recall-Oriented Understudy for Gisting Evaluation). Klu.ai. Retried from https://klu.ai/glossary/rouge-score.*