UniRAG: Unified Query Understanding Method for Retrieval Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) technology effectively addresses the issues of knowledge update lag and hallucinations in large language models (LLMs) by integrating internal and external knowledge. Existing query augmentation methods improve RAG's performance in handling complex queries but face two key challenges: (1) the separation of query augmentation and encoding tasks, which hinders information sharing and introduces cumulative errors, and (2) the difficulty of selecting the optimal augmentation strategy for different scenarios. In this work, we propose UniRAG, a unified framework for query understanding in RAG. UniRAG employs a decoder-only LLM to jointly perform query augmentation and encoding, eliminating task separation. To facilitate adaptive query augmentation, we categorize existing techniques into *query paraphrasing*, *query expansion*, and *query abstraction*. Our model learns to select the optimal augmentation strategy based on user queries, leveraging retrieval and generation outputs as feedback. Experimental results show that UniRAG significantly outperforms traditional query augmentation methods in five knowledge-intensive benchmark tasks in both closed and open domain question answering[1].

1 Introduction

In recent years, Large Language Models (LLMs) have become fundamental components of various Natural Language Processing (NLP) tasks due to their remarkable ability to comprehend and generate human-like text (Dubey et al., 2024; Brown et al., 2020; Hurst et al., 2024). Despite accumulating vast knowledge during pretraining, these models face inherent challenges such as outdated knowledge and the generation of inaccurate or misleading information. Recently, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Asai et al., 2024) has emerged as a standard approach to address these issues by integrating parametric and non-parametric knowledge through the retrieval of relevant passages.

Given the complexity of understanding user queries, many studies have explored query augmentation techniques using LLMs. For instance, (Ma et al., 2023) employs LLM feedback to train a query rewriter, while HyDE (Gao et al., 2023a) generates hypothetical documents to expand queries. Although these techniques have significantly improved RAG system performance, existing research still faces two key challenges: (1) **Separation of Query Augmentation and Encoding Tasks:** As shown on the left side of Figure 1, current approaches typically treat query augmentation and encoding as independent models or stages, limiting the effective sharing of information and potentially leading to cumulative errors. (2) **Lack of Adaptive Augmentation Strategy Selection:** Preliminary experiments in Section 7.1 indicate that different query augmentation techniques exhibit varying performance across different scenarios. Therefore, determining the optimal augmentation strategy for specific contexts or even specific user queries remains a significant challenge.

To address these challenges, this work introduces UniRAG. As shown on the right side of Figure 1, we achieve unified execution of query augmentation and query encoding by training a decoder-only LLM. To adaptively augment a given query, we categorize existing query augmentation techniques into three types: (1) *query paraphrasing*, (2) *query expansion*, and (3) *query abstraction*, and select a representative method for each of these three augmentation techniques. Using a diverse set of knowledge-intensive question-answering and retrieval datasets, we synthesize augmented query data by applying these augmentation techniques. Additionally, we leverage feedback scores from

[1]Our code is available at https://anonymous.4open.science/r/UniRAG-8260

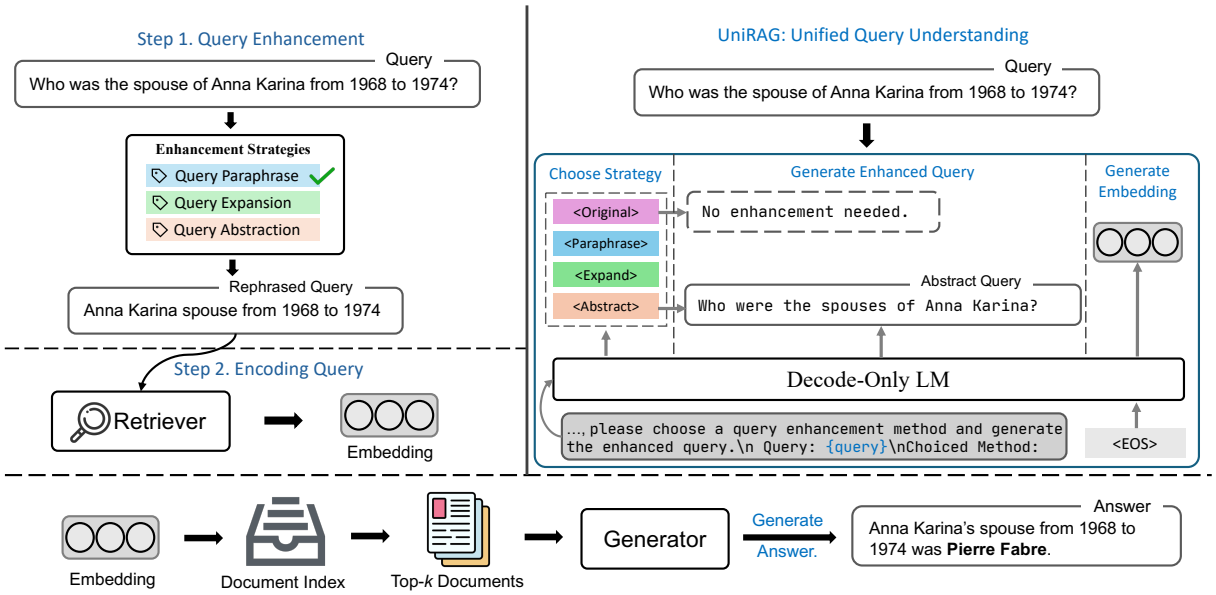


Figure 1: Overview of UniRAG. UniRAG unifies the query augmentation phase and the query encoding phase, and adaptively selects augmentation strategies for the given query during the augmentation phase.

both the retriever and generator as signals for selecting the appropriate augmentation strategy. To control the model's selection of different augmentation strategies, inspired by previous research (Asai et al., 2024; Schick et al., 2023), we supervise the model to generate action tokens and subsequently generate augmented queries corresponding to those actions, while also allowing the model not to perform query augmentation. To complete query encoding, we expand the dataset to fit the representation task, adding an extra `[EOS]` token to the input to obtain embeddings, and train the model's encoding ability using contrastive learning.

Compared to the previous query augmentation paradigm, **UniRAG** offers the following advantages: (1) **End-to-End Query Understanding:** A single model performs both query augmentation and encoding, reducing fragmentation in the process. (2) **Adaptive Strategy Selection:** UniRAG autonomously determines and applies the most appropriate query augmentation strategy based on the query. (3) **Plug-and-Play Compatibility:** UniRAG can be integrated with different LLMs, mitigating retrieval noise caused by insufficient query understanding. Furthermore, it supports customizable decoding algorithms, allowing for flexible query augmentation frequencies tailored to different downstream applications.

The experimental results show that UniRAG consistently outperforms independent query augmentation methods across five different benchmark datasets in both closed and open domains. Furthermore, our findings validate the significant advantages of integrating query augmentation and encoding within a unified framework.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) (Guu et al., 2020a; Gao et al., 2023b; Borgeaud et al., 2022; Asai et al., 2020, 2023) has become a key paradigm for large language models (LLMs), alleviating the hallucination problem of current LLMs by incorporating external knowledge, and achieving advantages in several downstream tasks such as code generation (Parvez et al., 2021; Lu et al., 2022; Wang et al., 2024c) and knowledge-based question answering (Yu et al., 2022; He et al., 2024).

Recently, more and more research (Zhu et al., 2024; Wang et al., 2024b; Xu et al., 2023) has focused on enhancing the performance of RAG systems from various aspects, such as improving decoding efficiency (Kim et al., 2024; Wang et al., 2024b), exploring long-context retrieval (Luo et al., 2024; Li et al., 2024c), and compressing prompts (Jiang et al., 2023; Xu et al., 2023). Despite the advantages of these RAG systems, they inevitably face potential noise issues in the retrieval content due to imperfect retrievers.

To address this problem, researchers have primarily explored three categories of techniques for augmentation: (1) Adaptive Retrieval: Self-RAG (Asai et al., 2024) and ReAct (Yao et al., 2023) introduce reflection techniques to determine whether

the retrieval can assist in answering. (2) Document Filtering: Xu et al. (2024) proposed incorporating reranking to reorder the final retrieval results; Kong et al. (2024) used prompts for LLMs to judge the relevance of documents. (3) Query Augmentation: Techniques such as paraphrasing (Ma et al., 2023), query expansion (Gao et al., 2023a; Lavrenko and Croft, 2017; Mao et al., 2023), and query abstraction (Zheng et al., 2023) are used to enhance the original query to handle diverse user queries.

2.2 LLM-based Embedding Models

Recently, techniques that utilize only decoder LLMs as embedding models have begun to emerge, showing significant improvements in accuracy and generalization capability within the domain. These studies treat LLMs as query and information encoders and optimize through contrastive learning. During this process, e5-mistral (Wang et al., 2024a) and Gecko (Lee et al., 2024b) incorporate synthetic datasets, and Gecko attempts to distill a smaller bidirectional embedding model from a decoder-only LLM. LLM2Vec (BehnamGhader et al., 2024) seeks to construct embedding models from LLMs using only publicly available data. SFR-Embedding-Mistral (Rui Meng, 2024) further fine-tunes on a mix of non-retrieval and retrieval datasets to enhance accuracy for both tasks. Llama2Vec (Li et al., 2024a) adapts LLMs to retrieval tasks by proposing autoencoding and autoregressive pre-training tasks. GritLM (Muennighoff et al., 2024) employs instruction fine-tuning techniques to enable LLMs to handle both generation and retrieval tasks. NV-Embed (Lee et al., 2024a) enhances pooling embeddings by adding latent attention and improves performance on retrieval and non-retrieval tasks through a two-stage instruction fine-tuning technique. bge-en-icl (Li et al., 2024b) generates high-quality text embeddings by incorporating few-shot examples.

3 Preliminary

We aim to produce an accurate response r from a user query q and external documents in a corpus $\mathcal{D}$. The traditional Retrieval-Augmented Generation (RAG) pipeline has three components: an *augmenter* E, a *retriever* R, and a *generator* G.

Query Augmentation. The augmenter E transforms the original query q into $\tilde{q}$. This process may add context or synonymous rewrites to improve the likelihood of retrieving relevant documents.

$$\tilde{q} = E(q). \tag{1}$$

Document Retrieval. The retriever R encodes $\tilde{q}$ and each document $d \in \mathcal{D}$ into a shared representation space. A similarity score (e.g., dot product) ranks how relevant d is to $\tilde{q}$. We select the top-k documents with the highest similarity.

$$\{d_1, \ldots, d_k\} = \underset{d \in \mathcal{D}}{\text{top-}k}\, \text{sim}\big(\tilde{q}, d\big). \tag{2}$$

Response Generation. We concatenate $\tilde{q}$ and the retrieved documents $d_1, \ldots, d_k$ into $\mathcal{C}$, then pass it to the generator G to produce the response r:

$$r = G\big(\mathcal{C}\big). \tag{3}$$

4 UniRAG Training

4.1 Overview

UniRAG can automatically select appropriate augmentation strategies for the raw query given by the user and encode the augmented query to achieve unified modeling of augmentation and encoding. As shown in Figure 2, UniRAG consists of two training stages: (1) **Query augmentation training:** Train the model using synthetic augmented queries and feedback data from the augmentation strategies. (2) **Query encoding training:** Augmented queries are added to the original retrieval dataset, and use contrastive learning to train the model. We summarize the training process in Algorithm 1.

4.2 Query Augmentation Training

4.2.1 Data Collection

Seed Dataset Collection. To augment the model's capability in handling diverse user queries, we propose a novel data annotation process to generate data for instruction tuning. We sample from knowledge-intensive QA and retrieval datasets, including Natural Questions (Kwiatkowski et al., 2019), MS MARCO Passages (Nguyen et al., 2016), BoolQ (Clark et al., 2019), NarrativeQA (Kočiskỳ et al., 2018), Dolly15k (Conover et al., 2023), and SQuAD (Rajpurkar, 2016), to construct a seed dataset for data synthesis. Detailed statistics are provided in Appendix A.1.

Data Synthesis. To obtain augmented queries, we select a representative method from each category of augmentation techniques. Specifically, for *query paraphrasing*, we adopt (Ma et al., 2023) to

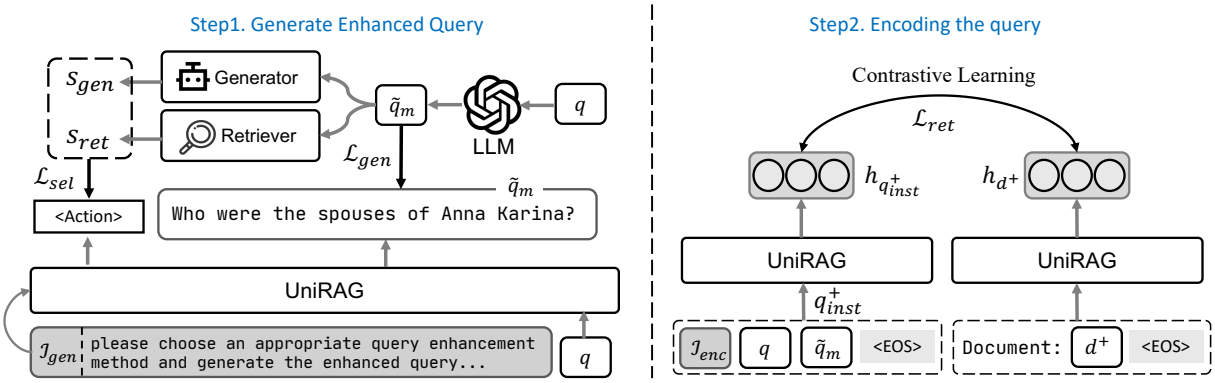


Figure 2: The two-stage training process of UniRAG: first, query augmentation training is conducted based on the augmented queries and feedback signals, followed by query encoding training using contrastive learning.

rephrase queries by resolving ambiguity and vagueness; for *query expansion*, we employ HyDE (Gao et al., 2023a) to generate hypothetical documents; and for *query abstraction*, we utilize Step-Back Prompting (Zheng et al., 2023) to abstract the original query. We then apply these augmentation techniques to the seed dataset using the GPT-4o-mini (Hurst et al., 2024). For a given query q and a selected augmentation strategy m from the set of all strategies $\mathcal{M}$, we generate the corresponding augmented query $\tilde{q}_m = E_m(q)$, where E_m denotes the augmenter instantiated with strategy m.

Feedback Signal Collection. To evaluate the quality of an augmented query $\tilde{q}_m$, we collect two types of feedback signals:

- **Retriever Feedback:** Retriever feedback captures how well a query x (either q or $\tilde{q}_m$) retrieves relevant documents. We calculate the reciprocal ranking as the retrieval feedback score: $s_{ret}(x) = \frac{1}{rank_x}$, where $rank_x$ is the position of the first relevant document in the ranking list retrieved by the retriever R for x.

- **Generator Feedback:** For a given query x, we retrieve candidate documents $\{d_1, \ldots, d_k\}$, concatenate them into the input context and feed them into generator G to generate an answer r:

$$\begin{aligned} \mathcal{C}(x) &= [x, d_1, \ldots, d_k], \\ p_G(r \mid \mathcal{C}(x)) &= \prod_{t=1}^{T} p_G(r_t \mid r_{1:t-1}, \mathcal{C}(x)). \end{aligned} \tag{4}$$

  where r_t is the token at step t in the generated answer, and T is the number of tokens in r. The generator score is: $s_{gen}(x) = \log p_G(r \mid \mathcal{C}(x))$.

For each original query q, we obtain feedback scores for q as well as for each augmented query $\tilde{q}_m$ produced using an augmentation strategy m. We then aggregate these scores into two collections:

$$\boldsymbol{s}_{ret} = \Big\{ s_{ret}(x) \,\Big|\, x \in \{q\} \cup \{\tilde{q}_m : m \in \mathcal{M}\} \Big\},$$

$$\boldsymbol{s}_{gen} = \Big\{ s_{gen}(x) \,\Big|\, x \in \{q\} \cup \{\tilde{q}_m : m \in \mathcal{M}\} \Big\}.$$

Data Filtering. Not all queries are suitable for every augmentation technique. To ensure reasonable application of augmentation methods and minimize noise in the synthetic dataset, we perform data filtering using the following criteria:

- We use Retriever R to retrieve documents for query q and discard any instances where the true document is not among the top k' results.

- If any feedback signal score for an augmented query $\tilde{q}$ is lower than that of the original query q, we discard the augmented instance.

To address the case where no augmentation is needed, we add the original query to the dataset when the original query has the highest feedback score, at which point $\tilde{q}_m$ is the empty string. The final dataset consists of instances of the format $(q, m, \tilde{q}_m, \boldsymbol{s}_{ret}, \boldsymbol{s}_{gen})$. We ended up collecting 263K data points for instruction tuning, including 105K valid feedback signals. See the Appendix A.1 for more details and analysis.

4.2.2 Training Objective

To train the model to both select the appropriate augmentation strategy m for a given query q and generate a corresponding augmented query $\tilde{q}_m$, we design a joint training objective. For each training instance, we first construct the input by applying an instruction template $\mathcal{I}_{gen}$ (details in Appendix D) to q and then define the expected output based on the augmentation strategy m and the corresponding augmented query $\tilde{q}_m$.

- **Augmentation Strategy Selection Loss $\mathcal{L}_{sel}$**: To align the model selection strategy probabilities with feedback scores, we map $\boldsymbol{s}_{ret}$ and $\boldsymbol{s}_{gen}$ to a vocabulary $\mathcal{V}$ of action tokens and apply a softmax function with a temperature of 0.1. This yields $p^*_{ret}(v)$ and $p^*_{gen}(v)$. At the designated action token position, the model predicts a probability distribution $\hat{p}(v \mid q)$, and we compute the KL-divergence between this predicted distribution and feedback score probability:

$$\mathcal{L}_{sel} = \mathrm{KL}\Big(p^*_{ret}(v) \,\Big\|\, \hat{p}(v \mid q)\Big) + \mathrm{KL}\Big(p^*_{gen}(v) \,\Big\|\, \hat{p}(v \mid q)\Big). \tag{5}$$

- **Augmented Query Generation Loss $\mathcal{L}_{gen}$**: Once the augmentation strategy is selected, the model generates the corresponding augmented query $\tilde{q}_m$. At each time step t, the model predicts the next token in $\tilde{q}_m$. The generation process follows the standard next token loss:

$$\mathcal{L}_{gen} = -\sum_{t=1}^{T} \log p_G\big(\tilde{q}_{m,t} \mid \tilde{q}_{m,1:t-1}, q, m\big), \tag{6}$$

where T is the total number of tokens in $\tilde{q}_m$. We do not supervise the predictions of action tokens.

The overall training objective combines the two losses: $\mathcal{L}_{enh} = \mathcal{L}_{sel} + \mathcal{L}_{gen}$.

4.3 Query Encoding Training

4.3.1 Data Collection

During the query augmentation stage, our collected dataset includes relevant document annotations, making it suitable for training the encoder. Specifically, we combine the original query q, augmentation strategy m, and the augmented query $\tilde{q}_m$ with a preset instruction template $\mathcal{I}_{enc}$ (details in Appendix D) to generate a new query q_{inst}, which is paired with annotated positive documents.

To achieve scalable expansion of the retrieval dataset, we further include other data instances from the original dataset that have not been augmented into the retrieval dataset. Since these additional data instances do not contain annotations for the augmented query, we uniformly add an `<Original>` token to them and apply the instruction template to generate the corresponding q_{inst}. Additionally, we mined hard negative samples for each query to improve retrieval performance. For detailed information on the collection of the retrieval dataset, see Appendix A.2.

4.3.2 Training Objective

Given a relevant query-document pair (q^+_{inst}, d^+), we append an `<EOS>` token to both the query and the document. These are then fed into the model to extract their respective embeddings from the final layer `<EOS>` vector, denoted as $h_{q^+_{inst}}$ and h_{d^+}.

To optimize the embedding model, we employ the InfoNCE loss $\mathcal{L}_{ret}$, which ensures that positive query-document pairs are assigned higher similarity scores compared to negative samples:

$$\mathcal{L}_{ret} = -\log \frac{\phi(q^+_{inst}, d^+)}{\phi(q^+_{inst}, d^+) + \sum\limits_{d^- \in \mathbb{N}} \phi(q^+_{inst}, d^-)}, \tag{7}$$

where $\mathbb{N}$ represents the set of all negative samples. The function $\phi(q, d) = \exp\left(\mathrm{sim}(h_q, h_d)/\tau\right)$, where τ is the temperature hyperparameter.

5 UniRAG Inference

In the query augmentation phase, UniRAG defaults to using constraint decoding to generate action tokens. After generating the action token, UniRAG applies the corresponding augmentation strategy and employs a *greedy search* to produce the augmented query. To adapt different task scenarios and flexibly control the augmentation frequency, we additionally designed two decoding strategies and summarized the inference process in Algorithm 2.

Threshold-based Decoding. This strategy dynamically determines whether to adopt an augmentation method by calculating the ratio of the generation probability of the `<Original>` token to the maximum generation probability among the action tokens corresponding to various augmentation methods. Formally, let $P(\texttt{<Original>})$ and $\max_{m \in \mathcal{M}} P(m)$ denote the generation probability of the `<Original>` token and the maximum generation probability of the action tokens, respectively. We define a threshold γ. When the following conditions are met, UniRAG will select the action token with the highest generation probability and generate the corresponding augmented query:

$$\frac{P(\texttt{<Original>})}{P(\texttt{<Original>}) + \max_{m \in \mathcal{M}} P(m)} < \gamma, \tag{8}$$

This decoding strategy achieves a balance between augmented accuracy and computational cost by controlling the threshold γ.

Tree-based Decoding. For scenarios with a higher inference budget, multiple query augmentation methods can be combined to further improve performance. In this strategy, UniRAG generates all action tokens whose generation probabilities are no lower than that of the `<Original>` token. Then, during the augmented query generation, beam search is applied with a global beam size B to explore all possible combinations of augmentation methods. In this way, we can ultimately generate a set of B augmented queries $\mathcal{Q}_B = \{\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_B\}$ and their corresponding embedding. In the retrieval phase, we apply Reciprocal Rank Fusion (RRF) (Cormack et al., 2009) to merge the retrieval results obtained from the different query embeddings, thereby improving the accuracy and recall of query matching.

6 Experimental Setups

6.1 Datasets

We comprehensively validated the effectiveness of UniRAG on **closed-set tasks** and **open-domain question answering tasks**.

Closed-set QA Tasks. The closed-set QA tasks include: (1) PubHealth (Zhang et al., 2023): a fact verification dataset for the public health domain. (2) ARC-Challenge (Clark et al., 2018): a multiple-choice reasoning dataset constructed from science exam questions.

Open-domain QA Tasks. The open-domain QA tasks cover the following datasets: PopQA (Mallen et al., 2022), TriviaQA-unfiltered (Joshi et al., 2017), and TimeQA (Chen et al., 2021). In the PopQA, the system needs to answer open questions involving factual knowledge. We employed the long-tail subset for evaluation containing queries involving rare entities. For TriviaQA-unfiltered, we followed the validation and test set division method of previous research (Asai et al., 2024; Guu et al., 2020b) to evaluate model performance. In the TimeQA, the model needs to answer queries involving complex time-sensitive knowledge.

We followed the experimental setup of previous research (Asai et al., 2024; Zheng et al., 2023), using accuracy as the evaluation metric. Our evaluation approach does not rely on strict text matching, but rather is based on whether the model-generated results contain the gold standard answers. Statistical details of these datasets are provided in the Appendix E.

6.2 Baselines

To comprehensively evaluate UniRAG's performance, we compare it with existing query augmentation methods. Specifically, we examine the following augmentation methods: (1) Paraphrase, which involves *abbreviation replacement* and *ambiguity elimination* of the original query, referencing the specific implementation in (Ma et al., 2023); (2) HyDE (Gao et al., 2023a), which generates hypothetical documents based on the query to expand the original query; (3) Step-back Prompting (Zheng et al., 2023), by abstracting higher-level concepts from the original query, generates abstract query.

To ensure fair comparisons, we uniformly use Llama-3-8B-Instruct (Dubey et al., 2024) as the base model and select Contriever-MS MARCO (Izacard et al., 2021) as the dense retriever to retrieve relevant passages. Furthermore, we report two reference baselines: (1) zero-shot prompting without retrieval as the no-retrieval baseline; (2) using the original query for retrieval as the standard retrieval baseline, in order to more clearly measure the improvements of UniRAG under different settings.

To verify the model independence of UniRAG, we used Llama-3-8B-Instruct, Llama-3-70B-Instruct and GPT-4o-mini (Hurst et al., 2024) as generators to evaluate its cross-model generalization ability. For the implementation details of the experiment, including the training and inference processes, please see Appendix C.

7 Results and Analysis

7.1 Main Results

Comparison of Individual Query Augmentation Methods. Table 1 presents the overall experimental results, showing that the performance impact of different query augmentation techniques varies across benchmark datasets. In most cases, applying query paraphrasing leads to consistent performance improvements. Additionally, using the HyDE method to generate hypothetical documents for query expansion yields the best results on TriviaQA and PubHealth. It is worth mentioning that in the PopQA, using the original query without any augmentations achieves the highest performance. Additionally, we observe that this variation in performance remains consistent across different generator. This finding underscores the importance of selecting the most suitable augmentation method for each query, as different datasets

Table 1: The overall performance of UniRAG and the baseline model in five knowledge-intensive tests was evaluated, and we conducted performance tests on three LLMs. The results of the optimal augmentation method are highlighted in ***bold***, and the best results obtained by using the augmentation method alone are *underlined*.

Model	Method	PopQA	TriviaQA	PubHealth	ARC-Challenge	TimeQA
Llama-3-8B-Instruct	Zero-shot Prompting	22.8	68.5	70.5	85.1	50.7
	Original Query	44.2	69.4	70.6	85.3	55.8
	Query Paraphrase	43.5	69.7	71.3	86.1	56.7
	HyDE	40.5	73.0	74.5	85.8	55.7
	Step-Back Prompting	42.0	63.7	70.2	85.2	56.9
	UniRAG	**49.2**	**76.8**	**77.5**	**90.4**	**60.5**
Llama-3-70B-Instruct	Zero-shot Prompting	28.9	69.2	60.5	86.3	51.4
	Original Query	40.8	71.1	58.2	89.7	56.1
	Query Paraphrase	39.5	71.2	58.3	90.1	56.3
	HyDE	32.1	74.4	59.0	89.7	55.1
	Step-Back Prompting	37.7	67.1	55.5	89.6	57.0
	UniRAG	**47.5**	**77.1**	**62.4**	**92.3**	**61.2**
GPT-4o-mini	Zero-shot Prompting	14.3	69.9	72.3	86.4	46.6
	Original Query	41.0	71.1	59.2	87.8	50.1
	Query Paraphrase	39.1	71.1	58.8	89.0	50.4
	HyDE	38.0	73.5	60.9	88.7	50.3
	Step-Back Prompting	40.8	68.8	58.4	90.3	51.9
	UniRAG	**47.6**	**78.0**	**63.6**	**92.5**	**58.3**

benefit from different augmentation strategies.

Comparison Between UniRAG and Baselines. Across all tasks, our proposed UniRAG consistently outperforms individual query augmentation methods. Moreover, UniRAG demonstrates robust improvements across all three generation models used in our experiments. These results highlight the advantages of integrating query augmentation and encoding into a unified framework while adaptively applying augmentation strategies. Notably, when individual query augmentation methods yield substantial performance gains, UniRAG exhibits even greater improvements. This observation further reinforces the effectiveness of our approach in dynamically selecting the most appropriate strategy, leading to superior overall performance.

7.2 Ablation Study

In order to identify which factors play a key role, we conducted a series of ablation experiments on our framework during the query augmentation phase and the query encoding respectively. We used Llama-3-8B-Instruct as the generator and presented the ablation results in Table 2.

7.2.1 Ablation Study on Query Augmentation

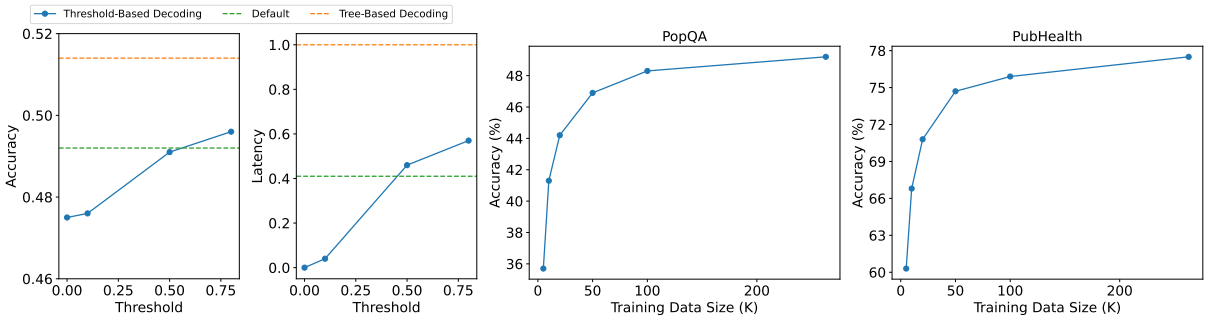
During the query generation phase, we try the following three model variants: (1) Using only retriever feedback (2) Using only generator feedback (3) Using a rejection sampling strategy to train the model to select the augmentation strategy with the highest retriever score. We see that all components play an important role, and the feedback from the generator is more important than the feedback from the retriever for the model's query augmentation strategy selection. Using the rejection sampling strategy resulted in the worst performance, indicating that it is crucial to design a refined loss function based on the feedback from the retriever and generator for selecting augmentation strategies.

Table 2: Results of ablation study on different modules.

	PopQA	Pub	ARC
UniRAG	49.2	77.5	90.4
Query Augmentation Phase			
Retrieval Feedback	47.9	74.8	88.5
Generator Feedback	48.5	75.5	89.2
Rejection Sampling	46.3	73.2	85.7
Query Encoding Phase			
Contriever	46.8	75.2	87.9
w/o $\mathcal{L}_{enh}$	47.9	76.3	89.4
– w/o Extra Data	47.3	75.9	88.7

7.2.2 Ablation Study on Query Encoding

During the query encoding phase, we tried different model variants: (1) Using Contriever-MSMARCO for retrieval (2) Skipping the training of the query augmentation phase (denoted as "w/o $\mathcal{L}_{enh}$") (3) Based on 2, only using the original retrieval data without adding augmented query data. We observed that using Contriever as the retriever resulted in the worst performance, as the decoder-only LLM has better generalization capabilities. Furthermore, the retriever trained with the Llama3-



(a) Decoding strategy (b) The impact of data size

Figure 3: **Analysis on :** (a) Performance and latency (*normalized*) analysis of different decoding strategies. (b) Impact of different amounts of synthetic data on model performance.

8b-base model still lags behind the UniRAG model trained in two stages, which proves that unifying the query augmentation stage and the query encoding stage can improve the model's performance.

7.3 Analysis

7.3.1 Accurate of the Selected Strategy

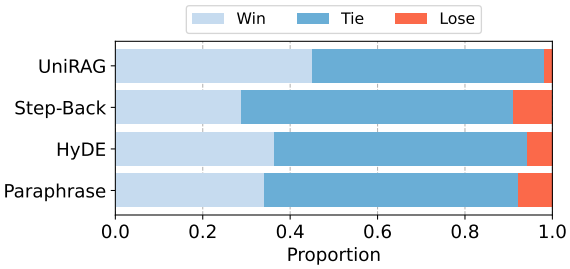


Figure 4: Comparison of win rates of different augmentation methods.

Based on the data collection process in Section 4.2.1, we additionally constructed a validation set of 2.1K entries and compared the predictive performance of the augmented strategy applied by UniRAG with that of using the original query for each question. We counted the wins and losses according to the ranking of the actual documents and added the comparative results of other augmentation methods, as shown in Figure 4. We found that UniRAG's augmentation strategy improves performance in over 90% of cases and has a better win rate compared to other augmentation methods.

7.3.2 Comparison of Inference Methods

We conducted an analysis of UniRAG's performance and latency under different inference methods. Specifically, we used Llama-3-8B-Instruct as the generator on the PopQA dataset, and the experimental results are shown in Figure 3a. We found that tree-based methods perform best in terms of performance, but their computational latency is also the highest. In addition, threshold-based inference methods show a trend of gradually improving performance with increasing threshold γ, while latency also increases correspondingly. These results indicate that the various decoding strategies we provide can adapt to different application needs.

7.3.3 Effects of Synthetic Data Size

We analyzed the impact of synthetic data scale on model performance. Specifically, we randomly sampled 5k, 10k, 20k, 50k, and 100k instances from the original 263k training instances and performed two-phase fine-tuning on these subsets. Subsequently, we compared the performance of these models on the PopQA and PubHealth benchmark datasets with that of models trained on the complete training dataset. The experimental results are shown in Figure 3b. Based on the results, an increase in training data typically accompanies an improvement in model performance. This trend suggests that further expanding the training dataset may yield additional performance gains.

8 Conclusion

In this work, we propose UniRAG, a unified query understanding framework for retrieval-augmented generation. By integrating query augmentation and encoding within a single model, UniRAG improves information sharing and reduces cumulative errors. Through adaptive selection of augmentation strategies and contrastive learning, our approach significantly boosts retrieval performance. Experimental results demonstrate UniRAG's effectiveness in improving both query understanding and retrieval accuracy, making it a robust and adaptable solution for RAG-based applications. Future research can combine UniRAG with broader augmentation strategies and explore improving query understanding capabilities by scaling testing time.

Limitations

Despite its advantages, UniRAG still has certain limitations. First, our approach relies on predefined augmentation strategies, which may not generalize well across all domains or user intents. Future work could explore more dynamic and broader scenarios for augmentation strategies. Secondly, although our model enhances retrieval accuracy, its performance is still affected by the quality of retrieved documents. In low-resource or highly specialized fields, this dependency can introduce noise that negatively impacts overall effectiveness.

References

Akari Asai, Sewon Min, Zexuan Zhong, and Danqi Chen. 2023. Retrieval-based language models and applications. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 6: Tutorial Abstracts)*, pages 41–46.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hanna Hajishirzi, and Wen tau Yih. 2020. Reliable, adaptable, and attributable language models with retrieval. 2024. url https://api. semanticscholar. org/corpusid: 268248911. oprea, and colin raffel. extracting training data from large language models. In *USENIX Security Symposium (USENIX)*.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *arXiv preprint arXiv:2404.05961*.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. *ArXiv*, abs/2203.05765.

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

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020b. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. 2024. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *arXiv preprint arXiv:2402.07630*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2024. Speculative decoding with big little decoder. *Advances in Neural Information Processing Systems*, 36.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Xiang Kong, Tom Gunter, and Ruoming Pang. 2024. Large language model-guided document selection. *arXiv preprint arXiv:2406.04638*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Victor Lavrenko and W. Bruce Croft. 2017. Relevance-based language models. *SIGIR Forum*, 51(2):260–267.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024a. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024b. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*.

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.

Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024a. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3490–3500.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024b. Making text embedders few-shot learners. *arXiv preprint arXiv:2409.15700*.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024c. Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 881–893, Miami, Florida, US. Association for Computational Linguistics.

Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seungwon Hwang, and Alexey Svyatkovskiy. 2022. Reacc: A retrieval-augmented code completion framework. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6227–6240.

Kun Luo, Zheng Liu, Shitao Xiao, Tong Zhou, Yubo Chen, Jun Zhao, and Kang Liu. 2024. Landmark embedding: A chunking-free embedding method for retrieval augmented long-context large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3268–3281, Bangkok, Thailand. Association for Computational Linguistics.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Kelong Mao, Zhicheng Dou, Fengran Mo, Jiewen Hou, Haonan Chen, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. *arXiv preprint arXiv:2303.06573*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human-generated machine reading comprehension dataset.

Md Rizwan Parvez, Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2021. Retrieval augmented code generation and summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2719–2734.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

P Rajpurkar. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Shafiq Rayhan Joty Caiming Xiong Yingbo Zhou Semih Yavuz Rui Meng, Ye Liu. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. 2024b. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Zora Zhiruo Wang, Akari Asai, Xinyan Velocity Yu, Frank F Xu, Yiqing Xie, Graham Neubig, and Daniel Fried. 2024c. Coderag-bench: Can retrieval augment code generation? *arXiv preprint arXiv:2406.14497*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Shicheng Xu, Liang Pang, Jun Xu, Huawei Shen, and Xueqi Cheng. 2024. List-aware reranking-truncation joint model for search and retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2024*, pages 1330–1340.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Wenhao Yu, Chenguang Zhu, Zhihan Zhang, Shuohang Wang, Zhuosheng Zhang, Yuwei Fang, and Meng Jiang. 2022. Retrieval augmentation for commonsense reasoning: A unified approach. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4364–4377.

Tianhua Zhang, Hongyin Luo, Yung-Sung Chuang, Wei Fang, Luc Gaitskell, Thomas Hartvigsen, Xixin Wu, Danny Fox, Helen Meng, and James Glass. 2023. Interpretable unified language checking. *arXiv preprint arXiv:2304.03728*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. In *The Twelfth International Conference on Learning Representations*.

Kun Zhu, Xiaocheng Feng, Xiyuan Du, Yuxuan Gu, Weijiang Yu, Haotian Wang, Qianglong Chen, Zheng Chu, Jingchang Chen, and Bing Qin. 2024. An information bottleneck perspective for effective noise filtering on retrieval-augmented generation. *arXiv preprint arXiv:2406.01549*.

A Training Dataset Collection

We collected a training dataset from multiple knowledge-intensive question-and-answer and retrieval datasets for query augmentation and query encoding training. We will introduce the collection process of this dataset and the statistical results of the final collected dataset.

A.1 Query Augmentation Dataset collection.

We create a seed dataset for data synthesis by sampling from various knowledge-intensive QA and retrieval datasets, including Natural Questions (Kwiatkowski et al., 2019), MS MARCO Passages (Nguyen et al., 2016), BoolQ (Clark et al., 2019), NarrativeQA (Kočiskỳ et al., 2018), Dolly15k (Conover et al., 2023), and SQuAD (Rajpurkar, 2016). In order to obtain a diverse seed dataset, we extracted 32K pieces of data from the Natural Questions and MS MARCO datasets, and 16K from other datasets. For datasets with fewer than 32K pieces of data, we included the complete dataset. For the MS Marco dataset, we performed stratified sampling based on the given `query_type` field, sampling 6.4K entries for each of the five types (`person`, `entity`, `location`, `numeric`, `description`); for Dolly15k, we only selected data where the `category` is `closed_qa`. Next, we conducted data synthesis and gathered feedback signals as well as data filtering based on the method outlined in section 1. Ultimately, we collected 263,185 data points for instruction tuning, including 105,096 valid feedback signals. Detailed statistical results are shown in Table 3. Additionally, we present the statistics of the data related to different augmentation methods in Table 4.

Table 3: Statistics of query augmentation data obtained from different seed datasets.

Dataset	Train	Feedback
Natural Questions	78,071	31,788
MS MARCO Passages	85,350	32,000
BoolQ	24,013	9,397
NarrativeQA	32,484	14,375
Dolly15k	4,931	1,759
SQuAD	38,336	15,777

A.2 Query Encoding Dataset Collection.

We have detailed the process of creating the retrieval dataset in Section 4.3.1, and we will explain in detail how to perform hard negative sample mining for each dataset. For MS MARCO Passages and Natural Questions, we used 30 difficult negative samples pre-mined from the Tevatron (Gao et al., 2022). For other datasets, we retrieved the top 30 passages from Wikipedia passages as difficult negative samples using the BM25 (Robertson and Walker, 1994). The statistical results of the final dataset are shown in Table 5.

Table 4: Statistics on the number of data for different augmentation methods.

	Ori	Expansion	Abstract	Para
Nums	5,123	82,052	87,247	88,763

Table 5: Statistics of query encoding data obtained from different seed datasets.

Dataset	Training Samples
Natural Questions	154,615
MS MARCO Passages	585,108
BoolQ	37,708
NarrativeQA	79,941
Dolly15k	7,064
SQuAD	135,340

B Training and Inference Algorithm

The training pipeline of UniRAG is elaborated in Section 4 with its formalized procedure summarized in Algorithm 1, while the inference mechanism is systematically described in Section 5 and algorithmically instantiated in Algorithm 2.

C Implementation Details

C.1 Training Details.

During the data collection process, we utilize the Contriever-MSMARCO retriever to gather retrieval feedback, while the Llama-3-8B-Instruct model is employed to collect generator feedback.

For training, we adopt the Llama-3.1-8B model as the base model and perform instruction tuning and contrastive learning. The training is conducted using four NVIDIA A100 GPUs, each with 80GB of memory. In the instruction tuning phase, the model is trained for 3 epochs with a batch size of 256, a peak learning rate of 2e-5, and a linear decay schedule following a 3% warmup step. The maximum token length is set to 512. We leverage DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) to enable

Algorithm 1 UniRAG Training

Input Generator G, Retriever R, Augmentation Strategies $\mathcal{M}$
Initialize UniRAG with pre-trained language model θ_0
Construct seed dataset $\mathcal{D}_{seed}$ from QA & retrieval corpora ▷ **Data Synthesis (Section 4.2.1)**
for $q \in \mathcal{D}_{seed}$ **do**
    for $m \in \mathcal{M}$ **do** ▷ Apply augmentation strategies
        Generate augmented query $\tilde{q}_m = E_m(q)$ via GPT-4o-mini
    end for
    Retrieve docs $\{d_i\}$ for q and $\{\tilde{q}_m\}$ using R ▷ Feedback collection
    Compute $\boldsymbol{s}_{ret}$, $\boldsymbol{s}_{gen}$ using R and G
    if $\exists\, m$ where $s_{ret}(\tilde{q}_m) < s_{ret}(q)$ or $s_{gen}(\tilde{q}_m) < s_{gen}(q)$ **then** ▷ Data filtering
        Discard underperforming $\tilde{q}_m$
    end if
    Store $(q, m, \tilde{q}_m, \boldsymbol{s}_{ret}, \boldsymbol{s}_{gen})$ in $\mathcal{D}_{enh}$
end for
for $(q, \tilde{q}_m, \boldsymbol{s}) \in \mathcal{D}_{enh}$ **do** ▷ **Query Augmentation Training (Section 4.2)**
    Map $\boldsymbol{s}_{ret}$, $\boldsymbol{s}_{gen}$ to the corresponding action token positions in $\mathcal{V}$ ▷ Strategy selection
    Compute $\mathcal{L}_{sel}$ via KL-divergence (Eq. 5)
    Compute $\mathcal{L}_{gen}$ via next token prediction (Eq. 6) ▷ Query generation
    Update θ using $\mathcal{L}_{enh} = \mathcal{L}_{sel} + \mathcal{L}_{gen}$
end for
for $(q^+_{\text{inst}}, d^+) \in \mathcal{D}_{retrieval}$ **do** ▷ **Query Encoding Training (Section 4.3)**
    Extract embeddings $h_{q^+_{\text{inst}}}$, h_{d^+}, and compute InfoNCE loss $\mathcal{L}_{ret}$ (Eq. 7) ▷ Contrastive learning
    Update θ using $\mathcal{L}_{ret}$
end for

Algorithm 2 UniRAG Inference

Input Original query q, UniRAG θ, Retriever R, decoding strategy ξ, threshold γ, beam size B
Generate action token probabilities $P(v|q)$ using θ
if ξ = Default **then** ▷ Default decoding
    Select $m^* = \arg\max_{m \in \mathcal{M}} P(m|q)$
    Generate augmented query $\tilde{q}_{m^*}$ via greedy search
else if ξ = Threshold **then** ▷ Threshold-based decoding
    Compute ratio $\rho = \frac{P(\texttt{<Original>})}{\max_{m \in \mathcal{M}} P(m)}$
    if $\rho < \gamma$ **then**
        $m^* \leftarrow \arg\max_{m \in \mathcal{M}} P(m|q)$
        Generate $\tilde{q}_{m^*}$ via greedy search
    else
        Set $\tilde{q}_{m^*}$ to an empty string
    end if
else ▷ Tree-based decoding
    Collect valid actions $\mathcal{M}' = \{m | P(m) \geq P(\texttt{<Original>})\}$
    Initialize beam candidates $\mathcal{C} \leftarrow \{q\}$
    for $m \in \mathcal{M}'$ **do** ▷ Multi-strategy exploration
        Generate augmented queries $\tilde{q}_m$ with beam search (size B)
        Add to candidates $\mathcal{C} \leftarrow \mathcal{C} \cup \{\tilde{q}_m\}$
    end for
end if
return augmented result ($\tilde{q}_{m^*}$ or $\mathcal{C}$)

multi-GPU distributed training with BFloat16 precision. Additionally, FlashAttention2 (Dao, 2024) is integrated to enhance the efficiency of long-context training.

During the contrastive learning phase, the model undergoes 1 epoch of fine-tuning, with the maximum token length for queries and documents set to 256, and the temperature parameter τ set to 0.01. To optimize GPU memory usage, we employ LoRA (Hu et al., 2022) with a rank of 16, gradient checkpointing, and DeepSpeed ZeRO-3, with BFloat16 precision enabled.

C.2 Inference Details.

In the action token generation process, constrained decoding is applied, while greedy search is used when generating augmented queries. In the retrieval step, following Asai et al. (2024), we use Wikipedia data as the external retrieval corpus for the five benchmarks used in this study, where each document is an independent text block extracted from Wikipedia articles containing up to 100 words. For each query, we retrieve the top 5 paragraphs from this corpus. During the generation process, we use vLLM (Kwon et al., 2023) for accelerated inference of the model. When using the tree-based decoding algorithm, we set the beam width to 4. In the baseline generation step, we call GPT-4o-mini via the official OpenAI API. The generator uses the default temperature and sampling algorithm.

D Prompt Details

During the training phase, we prompt GPT-4o-mini to generate augmented queries for a given query to synthesize data. We provide the prompts used for generating augmented queries with different augmentation strategies in Table 8. In the training process of the query augmentation phase, the input and output prompt templates used for constructing instruction fine-tuning data can be referred to in Table 6; in the training of the query encoding phase, the query templates used for constructing contrastive learning data can be referred to in Table 7. For the document, we directly add the prefix `"Document:"` at the front.

In the experiment, to ensure a fair comparison, we used the same prompt as in the synthetic data phase for the baseline of the independent use of query augmentation method, as shown in Table 8.

Task Instruction

For the given user query, please choose an appropriate query augmentation method (generating hypothetical documents: <Expand>, query paraphrase: <Paraphrase>, step back to generate abstract queries: <Abstract> or do not augment: <Original>), and provide the augmented query afterwards.

Prompt Template for Input ($\mathcal{I}_{gen}$)

Instruction: {task_instruction}
Input:
Query: {query}

Output Template (No Augmentation)

Response:
Choiced method: <Original>, No augmentation needed.

Output Template (Apply Augmentation)

Response:
Choiced method: {augmentation_strategy}
Augmented query: {augmented_query}

Table 6: Input and output template used for constructing query augmentation instruction data.

E Benchmark Dataset Details

We comprehensively validated the effectiveness of UniRAG on **closed-set tasks** and **open-domain question answering tasks**, covering five knowledge-intensive benchmark datasets.

The **closed-set tasks** include two datasets: (1) PubHealth (Zhang et al., 2023): a fact verification dataset for the public health domain. (2) ARC-Challenge (Clark et al., 2018): a multiple-choice reasoning dataset constructed from science exam questions. We followed the experimental setup of previous research, using accuracy as the evaluation metric and reporting on the test set.

The **open-domain question answering tasks** cover three datasets: PopQA (Mallen et al., 2022), TriviaQA-unfiltered (Joshi et al., 2017), and TimeQA (Chen et al., 2021). In the PopQA task, the system needs to answer open questions involving factual knowledge. We employed the long-tail subset for evaluation, which contains 1,399 queries

Query Template (No Augmentation)

Instruction: {task_instruction}
Input:
Query: {query}
Response:
Choiced method: <Original>, No augmentation needed.

Query Template (Apply Augmentation)

Instruction: {task_instruction}
Input:
Query: {query}
Response:
Choiced method: {augmentation_strategy}
Augmented query: {augmented_query}

Table 7: Query template ($\mathcal{I}_{ret}$) used for constructing contrastive learning data.

involving rare entities, all of which have fewer than 100 monthly views on Wikipedia. For TriviaQA-unfiltered, since its open test set is not publicly available, we followed the validation and test set division method of previous research (Asai et al., 2024; Guu et al., 2020b) to evaluate model performance on 11,313 test queries. Notably, our evaluation method does not require strict text matching, but rather is based on whether the model-generated results contain the gold standard answers. In the TimeQA task, the system needs to answer challenging queries involving complex time-sensitive knowledge. We evaluated on 5,226 test queries to assess the model's ability to handle time-dependent issues. We summarized the types of benchmark datasets used in the experiment and the statistics on the number of test questions in Table 9.

Prompts for Query Paraphrase

Please optimize the search engine query by removing irrelevant words and vague expressions, replacing abbreviations, and retaining important concepts to ensure accurate and smooth responses.
Query: {query}
Rewrite Query:

Prompts for the HyDE

Please write a passage to answer the question.
Question: {query}
Passage:

Prompts for the Step-Back Prompting

Your task is to step back and paraphrase a question to a more generic step-back question, which is easier to answer.
Here are a few examples:
Query: Who was the spouse of Anna Karina from 1968 to 1974?
Step Back Query: Who were the spouses of Anna Karina?
Query: Estella Leopold went to which school between Aug 1954 and Nov 1954?
Step Back Query: What was Estella Leopold's education history?
Query: {query}
Step Back Query:

Table 8: We refer to the prompt templates from the official implementation for various query augmentation methods.

Table 9: Benchmark Dataset Statistics.

Dataset	Type	Questions
PopQA	Open-domain	1,399
TriviaQA	Open-domain	11,313
PubHealth	Closed-set	3,610
ARC-Challenge	Closed-set	948
TimeQA	Open-domain	12,576