GainRAG: Preference Alignment in Retrieval-Augmented Generation through Gain Signal Synthesis

Yi Jiang, Sendong Zhao*, Jianbo Li, Haochun Wang, Bing Qin
Research Center for Social Computing and Interactive Robotics,
Harbin Institute of Technology, China
{yjiang,sdzhao,jbli,hcwang,qinb}@ir.hit.edu.cn

Abstract

The Retrieval-Augmented Generation (RAG) framework introduces a retrieval module to dynamically inject retrieved information into the input context of large language models (LLMs), and has demonstrated significant success in various NLP tasks. However, the current study points out that there is a preference gap between retrievers and LLMs in the RAG framework, which limit the further improvement of system performance. Some highly relevant passages may interfere with LLM reasoning because they contain complex or contradictory information; while some indirectly related or even inaccurate content may help LLM generate more accurate answers by providing suggestive information or logical clues. To solve this, we propose **GainRAG**, a novel approach that aligns the retriever's and LLM's preferences by defining a new metric, "gain", which measure how well an input passage contributes to correct outputs. Specifically, we propose a method to estimate these gain signals and train a middleware that aligns the preferences of the retriever and the LLM using only limited data. In addition, we introduce a pseudo-passage strategy to mitigate degradation. The experimental results on 6 datasets verify the effectiveness of GainRAG[1].

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Touvron et al., 2023) perform well in processing natural language tasks, but their knowledge is fixed in model parameters and is difficult to update dynamically over time (Ji et al., 2023; He et al., 2022). To tackle this issue, the Retrieval Augmented Generation (RAG) framework adds a retrieval module that brings in relevant external knowledge and integrates it into the input context of the LLMs. This approach has shown impressive results across various natural language processing tasks (Gao et al., 2023; Lewis et al., 2020). Previous work is devoted to solving two problems, namely, *retrieving more relevant information* in retrieval and *effectively utilizing context to generate the correct answer* in generation. However, they fail to address the preference gap between the retriever and the LLMs. Recent studies have highlighted that while retrieved passages may be relevant, they are not necessarily preferred for generation. In other words, only passages that align with the LLM's preferences can provide meaningful gain and enhance generation performance.

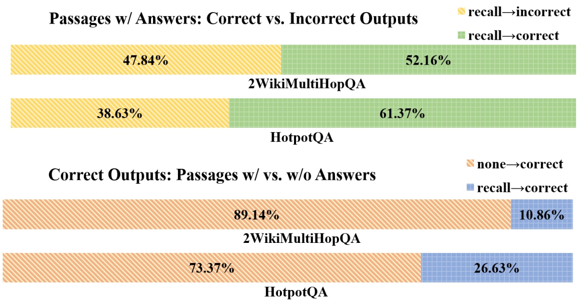

Figure 1: We analyze the preference gap between retrieved passages and LLMs on 2 datasets: HotpotQA and 2Wiki2MultiHopQA. The top shows the proportion of correct and incorrect generations when the retrieved passage contains the gold answer. The bottom shows the proportion of whether the passage used contains the golden answer when the LLM response is correct.

Specifically, existing retrievers are usually designed based on human-defined relevance criteria, such as whether a passage directly contains the answer to a question (Ke et al., 2024). However, this approach does not fully align with the way LLMs process information. Some highly relevant passages may actually disrupt LLMs reasoning by introducing complexity or contradictions, while some seemingly unrelated or even partially incorrect content can help by offering useful hints or

[*] Corresponding author
[1] The source code is publicly available at https://github.com/liunian-Jay/GainRAG

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10746–10757
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

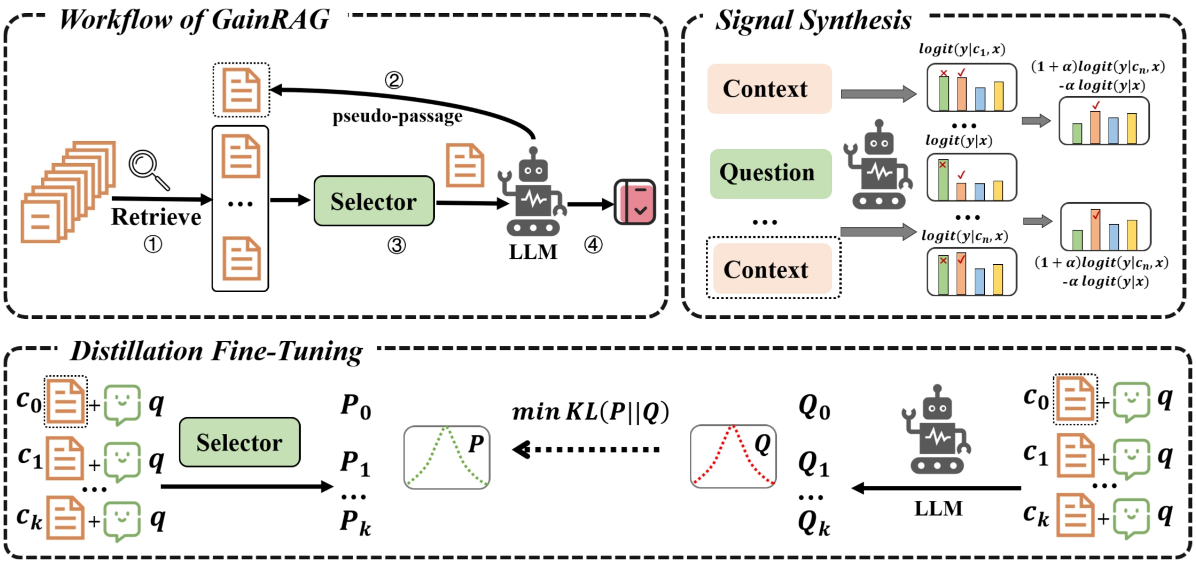

Figure 2: Illustration of the GainRAG framework. The GainRAG workflow, preference signal synthesis, and selector distillation fine-tuning are shown respectively.

logical cues (Dong et al., 2024; Cuconasu et al., 2024). Therefore, retrieval should shift its focus from traditional "relevance" to "gain"—prioritizing information that helps LLMs generate more accurate results.

To examine and validate this preference gap, we retrieve 100 relevant passages for each sample on two multi-hop question-answering datasets. Each passage is used to enhance the sample query for evaluation. As shown at the top of Fig. 1, we find that even when the retrieved passage contains the correct answer, nearly half of the samples still generate incorrect responses, indicating that while these passages are relevant, they are not particularly beneficial for generation. As shown at the bottom of Fig. 1, in most cases where the generation is correct, the passages used do not directly contain the answer. Instead, some passages that indirectly provide answers or clues may be less relevant but more beneficial, as they align better with the LLM's preferences.

Existing work aligns the retriever with the LLM's preferences mainly by fine-tuning the retriever or training both together. For example, Replug (Shi et al., 2023b) aligns the retriever with LLM preferences by training the retriever directly, while RA-DIT (Lin et al., 2023) uses dual training. However, this approach requires a large amount of high-quality data and is difficult to implement in real-world industrial settings. There are also some training middleware to align language model preferences, such as BGM (Ke et al., 2024) and DPA-RAG (Dong et al., 2024), but their perception and measurement of LLM preferences are coarse, capturing only basic patterns without a detailed understanding of nuanced differences.

To address the above challenges, we introduce a middleware between the retriever and the LLM to solve the problem of inconsistent preferences between the two, that is, the most profitable passages can be selected from the large number of passages retrieved by the retriever. Specifically, we introduce a method to quantify LLM's preference based on based on perplexity and contrastive decoding. This enables passage gain calculation and mitigates the LLM's overconfidence bias. Secondly, we use this to synthesize a small number of samples to distill the preference perception ability to the selector. Finally, we introduce a pseudo-passage strategy to prevent situations where all retrieved passages are not profitable. Together with the selector, it mitigates degradation and enables efficient integration of internal and external knowledge.

In general, our contributions can be summarized as follows:

- We analyze the preference gap between retrievers and LLMs, quantifying the LLM's preference for passages by defining "gain" and introducing GainRAG to address this gap.

- We provide a selector and introduce a pseudo-passage strategy that work together to not only avoid degeneration and triviality, but also achieve efficient integration of internal and external knowledge.

- We train with very few samples and validate on 6 datasets. The results show excellent performance and generalization of GainRAG.

2 Related Works

2.1 Retrieval-augmented Generation

In recent years, in order to solve the problems of outdated knowledge in the model and hallucination of large language models, retrieval-augmented generation has been introduced (Fan et al., 2024; Gao et al., 2023), and many efforts have been made in two aspects: "how to retrieve more relevant information" including retriever fine-tuning (Nian et al., 2024) and query optimization (Ma et al., 2023; Wang et al., 2023) and "how to better use the retrieved information to generate answers" including special fine-tuning (Wang et al., 2024; Zhang et al., 2024) and decoding strategies (Shi et al., 2023a).

2.2 Retriever-LLM Preference Alignment

The misalignment between the preferences of the retriever and LLM results in retrieved information being "relevant" but not always "useful". There are many existing works that have made efforts to solve this problem. Replug (Shi et al., 2023b) and Altas (Izacard et al., 2022) use LLM to supervise and guide the fine-tuning of the retriever. RA-DIT (Lin et al., 2023) uses dual training allows the language model to guide the training of the retriever and the language model to adapt to the retrieved information. However, these methods usually require a large amount of data, which makes the resources expensive and huge. DPA-RAG (Dong et al., 2024) synthesizes data and trains the reranker and LLM to align the preferences of the two. BGM (Ke et al., 2024) trains an intermediate to complete the rearrangement and selection of information from the coarsely ranked retriever. These methods simply do not clearly define and quantify this preference, which may lead to deviations in alignment. Different from them, we propose a method to quantify the preference, so that a small amount of data can be synthesized through LLM to fine-tune the selector and achieve alignment between the two.

2.3 Bridge Between Retriever and LLM

Many existing works achieve RAG optimization by providing a bridge between the retriever and LLM. Rerankers (Glass et al., 2022) such as BGE-reranker (Xiao et al., 2024) are the most common middleware, which can find more accurate passages from the information recalled by the retriever. Compressors such as RECOMP (Xu et al., 2024) and rewriters (Ma et al., 2023; Wang et al., 2023) are also commonly used bridges, which compress the retrieved passages to achieve denoising and solve the problem of context overload. Similar to them, GainRAG we proposed introduces a preference selector to select the optimal passage, which achieves score perception and avoids excessively long context.

3 Methodology

In this section, we introduce GainRAG (Fig. 2), starting with the synthesis and motivation of the gain signal, followed by selector training, and concluding with the GainRAG workflow.

3.1 Preliminaries

Before discussing GainRAG, we first provide a formal definition of the RAG problem. Given a query q and a corpus of documents $\mathcal{D}$, RAG systems typically follow a retrieval-then-reading framework. In this approach, the retriever selects relevant passages $C = \{c_1, c_2, , c_k\} \subset \mathcal{D}$ from the entire corpus $\mathcal{D}$, and the generator model (LLM) utilizes these passages C to generate the answer $\hat{a}$. This process can be represented as:

$$\begin{aligned} C &= \mathcal{R}(q, \mathcal{D}, k), \\ \hat{a} &= \mathcal{G}(\mathcal{P}(q, C)), \end{aligned} \tag{1}$$

where $\mathcal{R}$ denotes the retrieval function that selects k relevant passages, $\mathcal{P}$ represents the prompt template that combines q and C, and $\mathcal{G}$ is the generator, i.e., the LLM, which generates the final answer $\hat{a}$.

3.2 Gain signal to quantify preference

To make the passage used meet the preferences of the LLM, we need to quantify the benefit of a passage for the LLM to answer a question. To achieve this goal, we introduce contrastive decoding (Li et al., 2022) and calculate the perplexity (Li et al., 2023) after contrastive decoding.

Perplexity In instruction tuning, the model is trained to maximize the likelihood of a response given the instruction. Thus, perplexity (PPL) serves as an indicator of difficulty (Li et al., 2023, 2024). Specifically, the PPL of a given sample (q, a) is defined as

$$\mathrm{PPL}(a \mid q) = \exp\left(-\frac{1}{N}\sum_{j=1}^{N}\log p(a_j \mid q, a_1, \ldots, a_{j-1})\right). \tag{2}$$

Similarly, we can also use perplexity to measure how difficult it is for the LLM to generate the correct answer a for question q given c in RAG.

$$\mathrm{PPL}(a \mid q, c) = \exp\Bigg(-\frac{1}{N}\sum_{j=1}^{N}\log p\big(a_j \mid q, c, a_1, \ldots, a_{j-1}\big)\Bigg). \tag{3}$$

Contrastive Perplexity The indicator $\mathrm{PPL}(a \mid q, c)$ indicates the difficulty of the LLM to generate answers through query q and context c, but it cannot distinguish whether the answer generation is driven by context c or the LLM's internal knowledge. Therefore, directly using PPL for passage screening is biased. Inspired by CAD (Shi et al., 2023a), we introduce contrastive perplexity, which calculates perplexity using contrastive decoded logits (Li et al., 2022) to measure the gain of context c for answering a. This mitigates bias from the model's internal knowledge, enabling a more accurate quantification of the gain from context c to query q. Given our modeling of the internal prior $p(a_t \mid q, a_{<t})$, the probability distribution of the LLM output is adjusted as:

$$\begin{aligned} a_t &\sim \tilde{p}(a_t \mid c, q, a_{<t}) \\ &\propto p(a_t \mid c, q, a_{<t}) \left(\frac{p(a_t \mid c, q, a_{<t})}{p(a_t \mid q, a_{<t})}\right)^{\alpha}, \end{aligned}$$

where p is the original probability distribution from the LLM, $\tilde{p}$ is the distribution adjusted via contrastive decoding, $a_{<t}$ is the generated sequence, and α is a hyperparameter controlling the degree of adjustment. After rearranging the formula,

$$\begin{aligned} a_t \sim \mathrm{softmax}\big[&(1+\alpha)\,\mathrm{logit}_\theta(a_t \mid c, q, a_{<t}) \\ &- \alpha\,\mathrm{logit}_\theta(a_t \mid q, a_{<t})\big]. \end{aligned} \tag{4}$$

Therefore, the gain of passage c on the correct answer a generated by the LLM for question q can be quantified as $\mathcal{M}(c, a \mid q)$, formally,

$$\mathcal{M}(c, a \mid q) = \exp\left(-\frac{1}{N}\sum_{j=1}^{N}\log \tilde{p}\big(a_j \mid q, c, a_1, \ldots, a_{j-1}\big)\right). \tag{5}$$

3.3 Selector between LLM and Retriever

To effectively align the retriever's output with with the preferences of the LLM, we introduce a middleware, the selector, to identify the most beneficial passages. This selector leverages the context-gain-aware approach to refine passage selection, ensuring that the information fed into the LLM maximally enhances its performance.

The selector is formulated as a gain estimation problem, where gain estimates are distilled into a learnable function $f(q, c; \theta) \rightarrow \hat{v}$, where θ is the trainable parameter of the model. We employ the BGE pre-trained model as the foundation for the selector, defining its function mapping as:

$$\hat{\boldsymbol{V}} = [f(q, c_0), \ldots, f(q, c_k)], \tag{6}$$

where $\hat{V}$ denotes the predicted gain values for the retrieved passages C with respect to query q. This formulation enables the selector to effectively rank passages, ensuring that the most relevant and gainful content is utilized by the LLM.

3.4 Pseudo-passage Strategy

While increasing the number of retrieved passages raises the likelihood of finding useful information, passage selection can still become degenerate, that is, all retrieved passages do not provide any useful information, or augmenting with each retrieved passage may be worse than LLM direct response.

To address this challenge, we introduce the pseudo-passage strategy. Concretely, before selecting any external passages, we generate a pseudo-passage c_0 by prompting the LLM with the query. Formally, we define:

$$c_0 = \mathcal{G}(\mathcal{P}_0(q)), \tag{7}$$

where $\mathcal{P}_0$ refers to the prompt template used to generate the pseudo-passage. This pseudo-passage c_0 is then added to the selector's candidate list alongside the truly retrieved passages.

This strategy reduces over-reliance on potentially unhelpful retrieved passages and ensures that the selection of passages is always gain-oriented. Consequently, the pseudo-passage strategy not only mitigates degradation but also promotes the collaborative and efficient integration of internal and external knowledge of the LLM, ultimately leading to more robust performance.

3.5 Training of the LLM Preference Selector

As shown in Fig. 2, we train a middleware, i.e., selector, to align the preferences of the LLM by selecting the most beneficial passage.

Data Construction Starting with a set of QA pairs, for each $\{q, a\}$ pair, we first retrieve k relevant passages $C = \{c_1, \ldots, c_k\}$, and then calculate the gain to construct the training set $\{(q_i, a_i, c_i^j, v_i^j) \mid v_i^j = M(c_i^j, a_i \mid q_i)\}$. Additionally, to mitigate degradation, we generate and add a pseudo-passage c_0 to the set. Algorithm 1 summarizes this process.

Training Loss To make the selector aware of the relative gains of different c on the same q, we refer to Shi et al. (2023b); Lin et al. (2023) and use distillation loss. Note that due to the long-tail distribution of V and the large values at the tail, the label V we actually use is a simple transformation, that is, $v = -\log(v+1)$. The distillation loss is calculated as follows,

$$\begin{aligned} \boldsymbol{P} &= \operatorname{softmax}(V), \quad \boldsymbol{V} = [v_1, \ldots, v_k], \\ \boldsymbol{Q} &= \operatorname{softmax}(\hat{V}), \quad \hat{\boldsymbol{V}} = [\hat{v}_1, \ldots, \hat{v}_k], \\ \mathcal{L} &= \mathrm{KL}(\boldsymbol{P} \parallel \boldsymbol{Q}) = \sum_i \boldsymbol{P}_i \log \frac{\boldsymbol{P}_i}{\boldsymbol{Q}_i}. \end{aligned} \tag{8}$$

3.6 GainRAG Inference Workflow

After obtaining the selector, it acts as a middleware to align preferences between the retriever and the LLM.

Specifically, when a query q comes, we first prompt the LLM to generate the internal information about this query and use the retriever to retrieve several relevant passages. Formally,

$$c_0 = \mathcal{G}(q), \tag{9}$$

$$[c_1, \ldots, c_k] = \mathcal{R}(q, \mathcal{D}, k), \tag{10}$$

After getting all the passages, we use the selector to predict the gain of these passages for q and select the passage with the highest gain. Formally,

$$c^* = \arg \max_{c \in \{c_0, c_1, \ldots, c_k\}} f(q, c) \tag{11}$$

Finally, we use the selected optimal passage for enhanced generation to obtain the predicted answer. In terms of formula,

$$\hat{a} = \mathcal{G}(q, c^*). \tag{12}$$

Algorithm 2 summarizes this process.

Algorithm 1 Gain Signal Construction

```
Input: Original Dataset $D_o = \{(q, a), \ldots\}$, Corpus $\mathcal{D}$
Output: Enriched dataset $D_v = \{(q, c, v), \ldots\}$
Initialize $D_v \leftarrow \emptyset$                    ▷ Initialize
for $(q, a) \in D_o$ do
    $c_0 \leftarrow \mathcal{G}(\mathcal{P}_0(q))$          ▷ Generate pseudo-passage
    $[c_1, \ldots, c_k] \leftarrow \mathcal{R}(q, \mathcal{D}, k)$   ▷ Retrieve relevant passages
    for $c \in \{c_i \mid i = 0 \ldots k\}$ do
        Compute $v = \mathcal{M}(c, a \mid q)$
        Add $(q, c, v)$ to $D_v$
    end for
end for
return $D_v$                 ▷ Return the enriched dataset
```

Algorithm 2 GainRAG Inference Workflow

```
Input: Query $q$, Corpus $\mathcal{D}$
Output: Answer $\hat{a}$
$c_0 \leftarrow \mathcal{G}(\mathcal{P}_0(q))$        ▷ Generate pseudo-passage
$[c_1, \ldots, c_k] \leftarrow \mathcal{R}(q, \mathcal{D}, k)$
$\hat{V} \leftarrow []$                          ▷ Initialize score list
for $i \leftarrow 0$ to $k$ do             ▷ Iterate through all passages
    $\hat{V}[i] \leftarrow f(q, c_i)$           ▷ Compute score for $c_i$
end for
$\hat{i}^* \leftarrow \arg\max \hat{V}$
$c^* \leftarrow c_{\hat{i}^*}$                    ▷ Select the best passage $c^*$
$\hat{a} \leftarrow \mathcal{G}(q, c^*)$                 ▷ Predict the answer
return $\hat{a}$                      ▷ Return the final answer
```

4 Experiments

In this section, we report our experimental details and results, and provide an experimental analysis of GainRAG.

4.1 Implementation Details

Training Data We randomly selecte 20k samples from the HotpotQA training set and about 4k from the WebQuestions training set. For each sample, we gathere 21 relevant passages: 20 retrieved using the most common retriever Contriever (Izacard et al., 2021) and 1 generated internally. We then applie Algorithm 1 to add relevant passages and filter out samples where the passage with the highest gain is incorrectly generated, resulting in about 10k samples. The decoding α is set to 0.5 according to CAD (Shi et al., 2023a).

Training Details We use LLama3-8b to generate preference values and BGE-reranker-base (Xiao et al., 2024) for the selector's initial weights, training for 2 epochs. All experiments are conducted on a single A100 with 80G memory.

Inference Details During inference, we use Contriever (Izacard et al., 2021) as the retriever and

Method	HotpotQA			2WikiMultiHopQA			WebQuestions		
	EM	F1	Avg	EM	F1	Avg	EM	F1	Avg
w/o retrieval									
Naive	22.40	22.44	22.42	26.80	20.44	23.62	44.39	35.90	40.14
GenRead	31.00	30.50	30.75	30.60	25.24	27.92	**47.69**	31.42	39.55
w/ retrieval									
Standard RAG	31.80	33.23	32.51	23.40	21.81	22.61	35.04	33.26	34.15
Self-RAG	30.60	18.83	24.71	**34.00**	17.33	25.67	42.18	23.14	32.66
Rerank	35.80	37.45	36.62	24.20	22.94	23.57	37.50	35.55	36.52
GainRAG	**39.60**	**41.99**	**40.79**	31.40	**28.92**	**30.16**	42.51	**39.17**	**40.84**

Table 1: EM/F1/Avg(EM,F1) of different methods experimented on datasets HotpotQA, 2WikiMultiHopQA, WebQuestions. The best and second best scores are highlighted in **bold** and underlined, respectively.

Method	SQuAD			NaturalQA			TriviaQA		
	EM	F1	Avg	EM	F1	Avg	EM	F1	Avg
w/o retrieval									
Naive	18.50	21.57	20.03	31.25	29.02	30.13	60.20	59.96	60.08
GenRead	21.13	20.90	21.01	38.48	32.77	35.62	64.15	58.94	61.55
w/ retrieval									
Standard RAG	29.53	32.46	30.99	38.14	36.82	37.48	62.16	61.87	62.02
Self-RAG	27.69	14.78	21.23	35.60	39.78	37.69	61.65	35.21	48.43
Rerank	29.36	31.84	30.60	30.86	30.60	30.73	65.55	65.09	65.32
GainRAG	**34.65**	**37.55**	**36.10**	**41.97**	**41.27**	**41.62**	**67.29**	**66.73**	**67.01**

Table 2: EM/F1/Avg(EM,F1) of different methods experimented on datasets SQuAD, NaturalQA, TriviaQA. The best and second best scores are highlighted in **bold** and underlined, respectively.

LLama3-8B as the generator. We set the initial retrieval setting k to 100 because 100 has a high coverage, as shown in Appendix B analysis experiment. For all datasets, we use 21M English Wikipedia (Karpukhin et al., 2020) dump as the source passages for the retrieval. Prompts for the experiments can be found in Appendix D

Task Type	Datasets	# Samples
Multi-HopQA	2WikiMultiHopQA	500
	HotpotQA	500
OpenQA	WebQuestions	2032
	NaturalQA	3610
	SQuAD	10570
	TriviaQA	11313

Table 3: Description of tasks and evaluation datasets.

4.2 Datasets and Evaluation Metrics

Eval Datasets To verify the effectiveness and generalization of GainRAG, we use the open domain question answering datasets WebQuestion (Berant et al., 2013), NaturalQA (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017) and SQuAD (Rajpurkar, 2016), as well as the complex multi-hop question answering datasets HotpotQA (Ho et al., 2020a) and 2WikiMultiHopQA (Ho et al., 2020b). The statistics are shown in Table 3. Its detailed description can be found in Appendix A.

Evaluation Metrics We calculate exact match (EM) and F1 scores. Following Asai et al. (2023); Mallen et al. (2022), we apply a non-strict **EM** metric, which considers a model's generation correct if it includes the golden answer, rather than requiring an exact match. F1 measures the overlap between the predicted and golden answers. Note that in our study, longer responses tend to increase **EM** scores due to higher matching probabilities, but often lower **F1** scores due to irrelevant content. Therefore, the average of both metrics may be a more balanced evaluation.

4.3 Baselines

We selected several of the most common methods for comparison. 1) **StandardRAG**, which is the most classic "retrieve-then-read" paradigm. 2) **GenRead** (Yu et al., 2022): Its retriever can be seen as itself since it uses self-generated context to answer questions. It has almost no preference for misalignment, but there may be insufficient information. 3) **Self-RAG** (Asai et al., 2023): Through adaptive retrieval and self-criticism, it alleviates

Method	HotpotQA			2WikiMultiHopQA			WebQuestions			NaturalQA		
	EM	F1	Avg	EM	F1	Avg	EM	F1	Avg	EM	F1	Avg
Standard RAG	31.80	33.23	32.51	23.40	21.81	22.61	35.04	33.26	34.15	38.14	36.82	37.48
w/o all	35.80	37.45	36.62	24.20	22.94	23.57	37.50	35.55	36.52	30.86	30.60	30.73
w/o pseudo	37.80	40.65	39.23	27.20	24.88	26.04	41.24	38.97	40.11	41.25	40.94	41.09
w/o distillation	34.20	35.85	35.02	29.60	26.68	28.14	**43.21**	36.66	39.94	32.41	31.39	31.90
GainRAG	**39.60**	**41.99**	**40.79**	**31.40**	**28.92**	**30.16**	42.51	**39.17**	**40.84**	**41.97**	**41.27**	**41.62**

Table 4: Ablation studies, including: w/o all (removing all modules i.e., the ordinary reranker), w/o pseudo (removing the strategy for generating pseudo-passage, w/o distillation (removing the distillation fine-tuning.)

the preference misalignment problem to a certain extent. 4) **Rerank** (Glass et al., 2022; Xiao et al., 2024): It is a supplement to the classic RAG. It adds middleware between the retriever and the LLM. Following the "retrieve-rerank-read", we use the BGE-Reranker-base model. For fairness, Rerank has the same settings as ours. StandardRAG and Self-RAG also only use top-1. The rest of the settings follow the settings of their original papers.

4.4 Main Results

Experimental results are presented in Table 1 and Table 2, and we can get the following analysis:

1) Our method achieves state-of-the-art performance on almost all datasets. Despite using only a small subset of HotpotQA and WebQuestions for data synthesis, it generalizes well across datasets, demonstrating the robustness of GainRAG.

2) In WebQuestions, RAG methods generally underperform compared to those without external knowledge, suggesting that retrieval is not always beneficial. In cases of preference misalignment, retrieved passages can even be harmful. However, our approach still achieves the best average performance.

3) The reranking method outperforms other baselines, proving that middleware integration is both simple and effective. By leveraging a small amount of data to mitigate preference misalignment, our method significantly surpasses standard reranker.

4.5 Ablation Study

In order to verify the effectiveness of each module, we conducted ablation experiments on several datasets. The results, shown in Table 4, confirm that every module plays a crucial and irreplaceable role. The key findings are:

1) Without distillation fine-tuning, performance drops significantly, highlighting the importance of preference alignment. However, due to the existence of the pseudo-passage strategy, the performance can still be significantly improved over the common methods.

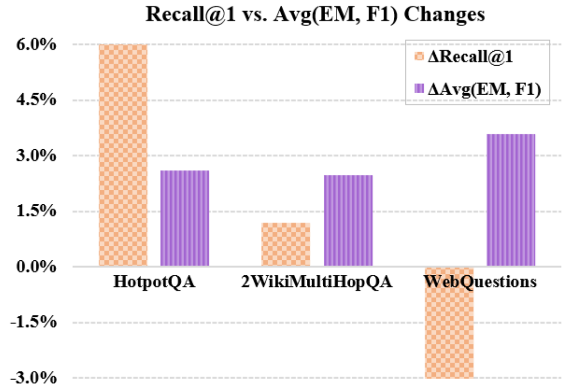

Figure 3: Illustration of gain. Changes in recall of the gold answer and downstream performance after using GainRAG.

2) When pseudo passage strategy is absent, performance drops on some datasets significantly but not all. It shows that it plays a significant role when there is a degenerate solution, that is, when the correct preferred passage cannot be retrieved.

3) Pseudo passages and preference perception fine-tuning are complementary and essential. Together, they prevent degenerate solutions and improve passage selection, aligning preferences between the retriever and the LLM.

4.6 Effect of Preference Selection

In order to explore why GainRAG improves the response performance of downstream LLMs, we removed the pseudo-passage strategy and calculated the changes in Recall@1 and downstream generation metrics.

As shown in Fig. 3, we find that there are three general cases. 1) Our selector sometimes significantly improves the Recall@1, which further improves downstream performance. 2) Our selector does not significantly improve the Recall@1, but the downstream performance are significantly improved. 3) Our selector reduces the Recall@1, but the downstream performance are significantly improved.

Case 1 is intuitive, while Cases 2 and 3 demonstrate that our selector's impact goes beyond just relevance, highlighting the benefits of selection based on gain. This further confirms the effectiveness of our approach.

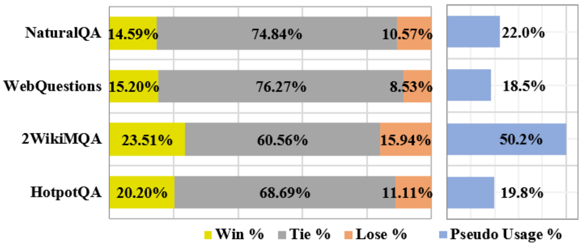

Figure 4: Illustration of the pseudo-passages generated for each dataset to avoid degenerate solutions.

4.7 What is the Effect of Pseudo-passage?

To examine the role of pseudo-passages in mitigating performance degradation, we analyze their usage across different datasets. Specifically, we count the overall use of pseudo-passages, as shown in the right part of Fig. 4. In addition, we replace these cases with non-pseudo-passage with the largest gain and performed a Win-Tie-Lose comparison, as shown in the left part of Fig. 4.

In many cases, internally generated passages are selected, with internal knowledge used in 50% of the cases on 2WikiMultiHopQA. This is consistent with the comparative experiment, which shows that GenRead significantly improves performance on 2WikiMultiHopQA, highlighting the value of LLM-generated passages for this dataset. Additionally, the Win-Tie-Lose comparison reveals that the number of winning cases after replacing pseudo-passages with the highest-gain passages far outweighs the losing cases, further demonstrating the effectiveness of pseudo-passages in alleviating degradation.

4.8 Synthetic Signal Analysis

To explore the effect of contrastive decoding debiasing, we use the ordinary PPL synthetic signal to fine-tune the selector under the same settings. Its performance and changes are shown in Table 5.

We find that contrastive debiasing is crucial, as removing it weakens preference perception for individual passages. This is because this strategy enhances the model's perception of gain rather than just relevance, thereby alleviating over-reliance on LLM internal knowledge.

Datasets	EM / F1 / Avg(EM,F1)
HotpotQA	38.2 (↓ 1.40) / 41.38 (↓ 0.61) / 39.79 (↓ 1.00)
2WikiMQA	29.4 (↓ 2.00) / 27.12 (↓ 1.80) / 28.26 (↓ 1.90)

Table 5: Performance degradation after removing contrastive decoding

4.9 In-Depth Comparison with the Reranker

To explore the impact of the number of selector choices on performance and compare it with the reranker, we set the selector to the interval [1,5] and observe the performance changes, as shown in Fig. 5. The results reveal the following:

1) Increasing K significantly boosts the recall rate, as expected, since more passages increase the likelihood of including the gold answer.

2) For our selector, selecting the top passage is usually sufficient. Even for general rerankers, increasing K does not improve downstream generation performance, and longer contexts even add overhead and may be harmful.

3) While the recall rate increases, downstream generation performance remains largely unchanged, highlighting the scientificity and rationality of selection. This further supports the observation that simply including the gold answer does not guarantee correct generation.

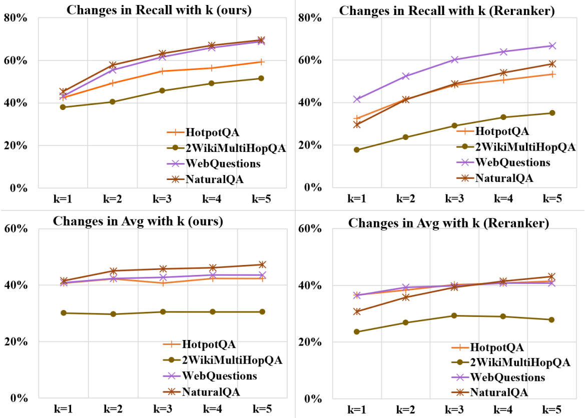

Figure 5: As the number of passages increases, the changes in recall and downstream generation performance. The left part is the change of our selector, the right side is the BGE-reranker, and the upper and lower parts are recall and Avg(EM, F1) respectively.

5 Conclusion

This work analyzes the preference gap between retrievers and LLMs and proposes GainRAG to address this misalignment. We define and quantify preferences, then fine-tune a selector with signals from a small number of samples. By adding a selector and using a pseudo-passage strategy to prevent degradation, GainRAG effectively integrates internal and external knowledge of LLMs, achieving superior performance.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China [62206079]; and the Key R&D Program of Heilongjiang Province [2023ZX01A11]. We also appreciate the support from China Mobile Group Heilongjiang Co., Ltd. @ on our research, the research is jointly completed by both parties.

Limitations

GainRAG selects passages with gain by calculating the gain score. However, this selection may not be the optimal solution. Whether there are some combinations of passages that make the gain stronger remains to be verified. And we only used a very small amount of training data to show the effect. In the future, large-scale data training experiments are still needed to verify whether it will get better performance. In addition, for the signal generation of large-scale data, whether a small model can be used as a generator when generating signals to accelerate the experiment is also a need for further experimental verification.

References

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

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Zhicheng Dou, and Ji-Rong Wen. 2024. Understand what llm needs: Dual preference alignment for retrieval-augmented generation. *arXiv preprint arXiv:2406.18676*.

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*.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Hangfeng He, Hongming Zhang, and Dan Roth. 2022. Rethinking with retrieval: Faithful large language model inference. *arXiv preprint arXiv:2301.00303*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020a. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020b. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

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*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 1(2):4.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea

Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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.

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*.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and llms. *arXiv preprint arXiv:2401.06954*.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang-Woo Lee, Minjoon Seo, Jung-Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. *arXiv preprint arXiv:2404.13081*.

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.

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.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *arXiv preprint arXiv:2402.00530*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *arXiv preprint arXiv:2308.12032*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. 2023. Ra-dit: Retrieval-augmented dual instruction tuning. *arXiv preprint arXiv:2310.01352*.

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, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7.

Jinming Nian, Zhiyuan Peng, Qifan Wang, and Yi Fang. 2024. W-rag: Weakly supervised dense retrieval in rag for open-domain question answering. *arXiv preprint arXiv:2408.08444*.

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

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Scott Wen-tau Yih. 2023a. Trusting your evidence: Hallucinate less with context-aware decoding. *arXiv preprint arXiv:2305.14739*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023b. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423.

Yuhao Wang, Ruiyang Ren, Junyi Li, Wayne Xin Zhao, Jing Liu, and Ji-Rong Wen. 2024. Rear: A relevance-aware retrieval-augmented framework for open-domain question answering. *arXiv preprint arXiv:2402.17497*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2022. Generate rather than retrieve: Large language models are strong context generators. *arXiv preprint arXiv:2209.10063*.

Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*.

A Dataset

Here, we introduce in detail the datasets we used, which are seven datasets on four tasks.

2WikiMultiHopQA (Ho et al., 2020b) and **HotpotQA** (Ho et al., 2020a): Both datasets are multi-hop question answering datasets based on Wikipedia. Considering the limitation of experimental cost, we used the sub-sampling set published by Trivedi et al. (2022); Kim et al. (2024), which is obtained by extracting 500 questions from the validation set of each dataset.

WebQuestions (Berant et al., 2013): Constructed from questions posed by the Google Suggest API, where the answers are specific entities listed in Freebase.

NaturalQA (Kwiatkowski et al., 2019): A dataset designed to support comprehensive QA systems. It consists of questions from real Google search queries. The corresponding answers are text spans from Wikipedia articles, carefully identified by human annotators.

SQuAD (Rajpurkar, 2016): It is a dataset for evaluating reading comprehension, created by annotators who generate questions based on the documents they read. It is widely used for training and testing open-domain QA systems.

TriviaQA (Joshi et al., 2017): A compilation of trivia questions paired with answers, both originally pulled from online sources.

B Coverage Study

To analyze the performance of the original retriever, we conducted experiments on four datasets. Specifically, we used the retriever to retrieve [1, 5, 10, 20, 50, 100] paragraphs respectively and calculated the recall, EM coverage, and F1 coverage. For EM coverage, we used each paragraph to enhance the query separately, and as long as there is one correct response, it is considered to be covered. For EM coverage, we used each paragraph to enhance the query separately, took the response with the largest F1 value, and calculated the average of the overall dataset.

As shown in Fig. 6,Fig. 7 and Fig. 8, as K increases, the recall and coverage will steadily increase. When retrieving 100, the coverage is large enough and far exceeds that of the current state-of-the-art RAG method.

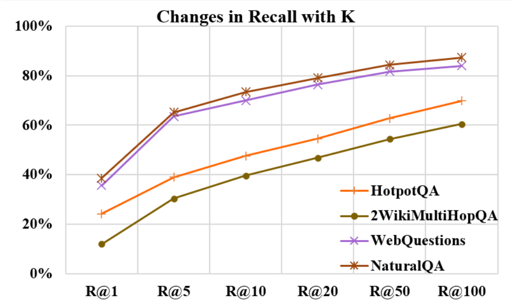

Figure 6: Illustration of the change in recall as the number of retrievals K increases

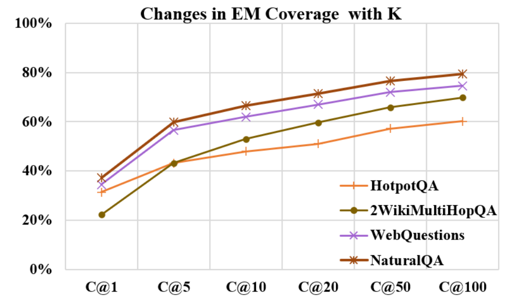

Figure 7: Illustration of the change in EM coverage as the number of retrievals K increases

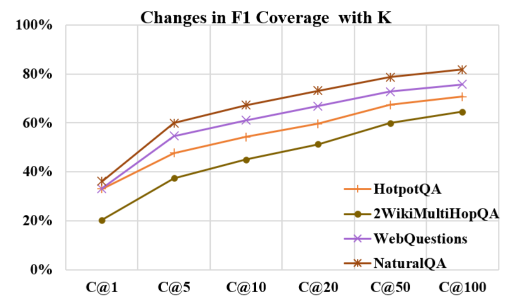

Figure 8: Illustration of the change in F1 coverage as the number of retrievals K increases

C Training Details

We use LLama3-8b as our generator for generating preference values and BGE-reranker-base (Xiao et al., 2024) for initializing the selector. We train selection for 2 epochs. During fine-tuning, we set train-group-size to 16 and batch-size to 8. In addition, during training, we randomly select 16 out of 21 passages to ensure generalization. The rest of the settings follow the official fine-tuning script (Xiao et al., 2024). Regarding the training data, we randomly selected 20,000 samples from the Hotpot training set and all 3,778 samples from the WebQuestion training set. After filtering, we finally obtained 14,084 training data. All experiments are conducted on a single A100 with 80G memory.

D Prompt Templates

All the prompt templates used by our proposed GainRAG are shown in Table 6.

Task	**Task Instruction**
Generation	$\{passage\}$ \n ### Instruction: \n Answer the question below concisely in a few words. \n\n ### Input: \n $\{query\}$
Pseudo-Passage	Please provide background for the question below in 100 words. Do not respond with anything other than background. If you do not know or are unsure, please generate “N/A” directly. Question: $\{query\}$

Table 6: Full list of instructions used during zero-shot evaluations and pseudo-passage generation. Where $query$ and $passage$ are the paragraph to be used and the question to be answered.