Test-Time Strategies for More Efficient and Accurate Agentic RAG

Abhinav Sharma[1]*, Brian Zhang[1]*, Deepti Guntur[1]*, Zhiyang Zuo[1]*, Shreyas Chaudhari[1], Wenlong Zhao[1], Franck Dernoncourt[2], Puneet Mathur[2], Ryan A. Rossi[2], Nedim Lipka[2]

[1]University of Massachusetts Amherst [2]Adobe Research

{abhinavs, bszhang, dguntur, zzuo, schaudhari, wenlongzhao}@umass.edu

{dernonco, puneetm, ryrossi, lipka}@adobe.com

Abstract

Retrieval-Augmented Generation (RAG) systems face challenges with complex, multi-hop questions, and iterative agentic frameworks such as Search-R1 (Jin et al., 2025) have been proposed to address these complexities. However, such approaches can introduce inefficiencies, including repetitive retrieval of previously processed information and challenges in contextualizing retrieved results effectively within the current generation prompt. Such issues can lead to unnecessary retrieval turns, suboptimal reasoning, inaccurate answers, and increased token consumption. In this paper, we investigate test-time modifications to Search-R1's open-source Qwen2.5-7B pipeline to mitigate these identified shortcomings. Specifically, we explore the integration of two components and their combination: a contextualization module to better integrate relevant information from retrieved documents into reasoning, and a deduplication module that replaces previously retrieved documents with the next most relevant ones. We evaluate our approaches using the HotpotQA (Yang et al., 2018) and the Natural Questions (Kwiatkowski et al., 2019) datasets, reporting the exact match (EM) score, an LLM-as-a-Judge assessment of answer correctness, and the average number of turns. Our best-performing variant (contextualization) achieves a 5.6% increase in EM score and reduces the average number of turns by 10.5% compared to the Search-R1 baseline. While contextualization itself introduces additional LLM calls, our results demonstrate improved answer accuracy and reduced retrieval load.

1 Introduction

RAG systems have shown promising results in complex question answering (QA) tasks by combining external document retrieval with generative language models (Lewis et al., 2021). Despite this success, traditional RAG systems that rely on a single-step retrieval and generation process often struggle to handle complex or nuanced questions, especially those requiring deep contextual understanding and multi-hop retrieval. To address these complexities, recent research has proposed agentic RAG systems which utilize large language model (LLM) agents to orchestrate retrieval, refine search queries, and optimize responses (An et al., 2024; Chan et al., 2024; Singh et al., 2025). Another popular approach is to augment the reasoning loop of LLMs with a retrieval tool, enabling the model to autonomously use retrieval while performing multi-step reasoning (Li et al., 2025; Jin et al., 2025).

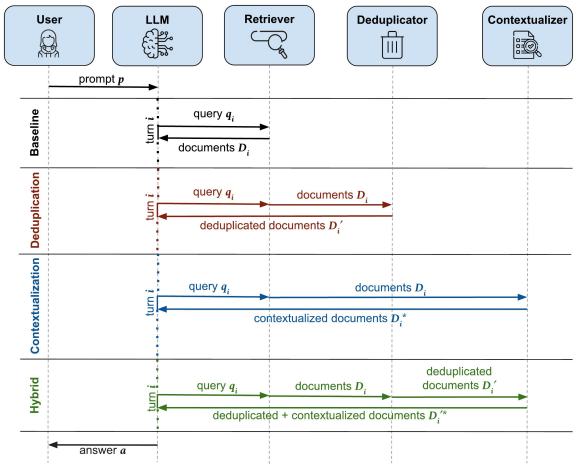

Figure 1: An illustration of the information flow for our proposed test-time strategies compared to the baseline during a single inference turn i. *Baseline:* This represents the standard Search-R1 framework, where the LLM sends a query (q_i) to the retriever and directly receives the retrieved documents (D_i) to continue its reasoning. *De-duplication:* This approach filters out previously seen content and returns only a set of novel documents (D_i') to the LLM. *Contextualization:* This approach parses the retrieved documents (D_i) and reformulates their content to improve integration into the LLM's reasoning process, returning an enhanced set of information (D_i^*). *Hybrid:* This approach combines both modules sequentially.

*Equal contribution.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 463–469

A notable example of this approach is the Search-R1 framework, which uses reinforcement learning (RL) to train LLMs for interleaved reasoning and retrieval (Jin et al., 2025). At inference time, the Search-R1 model first performs reasoning on a given user prompt p to either produce an answer a or generate a search query q to retrieve supporting information D. More specifically, in the i-th turn, the query q_i is sent to a dense retriever E5 (Wang et al., 2024), which returns relevant passages D_i from a 2018 Wikipedia dump. D_i is directly incorporated into the reasoning trace and fed back into the LLM, which continues its reasoning and repeats these steps until the final answer a is produced, see Figure 1 (Baseline). The Search-R1 models are trained using RL methods, such as PPO and GRPO, optimizing the exact match (EM) score between the ground truth and the predicted answer.

While Search-R1 has achieved substantial improvement—up to 41% over its baseline—our analysis of the Qwen2.5-7B Search-R1 model during inference has revealed several shortcomings. First, the model often performs repetitive retrieval of previously processed information, which leads to unnecessary retrieval turns, and increased token consumption and latency. Secondly, the model often struggles to effectively contextualize retrieved passages, leading to suboptimal reasoning and inaccurate answers.

These observations raise two questions: whether a concise representation of retrieved information can help the LLM be more accurate while issuing fewer retrievals, and whether preventing redundant retrieval alone can encourage greater contextual diversity and improve answer quality. Our work builds upon and extends the Search-R1 framework (Jin et al., 2025) and investigates test-time approaches that target both questions. We address the identified limitations through three test-time modifications that process the retrieved results D: (1) a **contextualization** module, (2) a **de-duplication** module, and (3) a **hybrid** approach that combines both.

2 Related Work

A Memory Knowledge Reservoir (Shi et al., 2024) stores previously retrieved content as a title-document pair. This system first consults previously retrieved information before formulating a new query. This allows the system to produce more targeted queries. As a result, they can reduce the response time by 46% while preserving the accuracy of the response of their baseline. Rather than storing the entire document, we propose a contextualization module that prompts an external LLM to extract useful information, and incorporate this into the model's reasoning chain after each retrieval step and before the next query is generated.

Search-o1 (Li et al., 2025) addresses the issue of hallucination in reasoning models by incorporating an agentic RAG mechanism to extract information from documents before integrating it into a reasoning chain. We use a similar approach to Search-o1 by utilizing an external LLM to contextualize helpful information in a document. Unlike Search-o1 which only provides the extracted information, our pipeline retains previously contextualized information, which is passed along with newly retrieved documents at each reasoning step.

RAG-RL (Huang et al., 2025) introduces a reasoning language model specifically trained for RAG tasks using reinforcement learning and curriculum learning strategies. The authors demonstrate that stronger answer generation models can identify relevant contexts within larger sets of retrieved information, thus alleviating the burden on retrievers and enhancing overall performance. By benchmarking on HotpotQA and MuSiQue datasets, RAG-RL achieves performance that surpasses previous generative reader models. RAG-RL's rewards are all rule-based and determined by the final answer, output format, and the citations included. Our work only explores test-time approaches and does not involve any modifications to the model architecture or training process.

3 Approach

To understand the limitations of Search-R1, we conducted a qualitative analysis of Search-R1's reasoning chains. These chains contain the original user prompt, multiple turns of the model reasoning, search queries, and retrieved documents, as well as the final answer. We observed two primary limitations in the Search-R1 model. First, **Information Forgetting:** the model struggles to retain and utilize information from previous retrieval steps, often resulting in redundant or duplicate retrieval queries before arriving at a final answer. Second, **Ineffective Information Extraction:** the model often fails to effectively identify and extract the most relevant information from the retrieved documents, which hinders its reasoning and overall accuracy of

the answers.

Based on these findings, we propose three test-time modifications to the Search-R1 pipeline aimed at addressing the challenges of information forgetting and ineffective information extraction. For each approach, we evaluate performance using Exact Match, LLM Match score (an LLM-as-a-judge metric for semantic equivalence; defined in Section 4.1.5), and the average number of retrieval steps.

3.1 Contextualization

To assess the importance of extracting and retaining relevant information across retrieval steps, we introduce the Contextualization module, shown in Figure 1 (Contextualization). This is an additional component in the pipeline that leverages an external language model to extract relevant information from retrieved documents and maintain a persistent memory cache based on previously contextualized information. After each retrieval step, the external LLM identifies concise, useful content and updates the cache accordingly. At each reasoning step, the model accesses both the most recently retrieved documents and the accumulated cache, allowing it to reason over both new and previously retained information. We use a separate, more capable LLM rather than the Search-R1 backbone itself for contextualization in order to isolate the contextualization mechanism from the backbone's own summarization quality. Using the Search-R1 model as its own contextualizer is a natural ablation that we leave to future work.

We provide the LLM with a structured prompt that instructs it to extract only the information relevant to answering the user prompt p from the newly retrieved documents D_i during each retrieval turn i. This extracted content D_i^* is then appended to a persistent memory cache that accumulates across retrieval steps. The external language model is constrained to preserve all previously stored information and may only add new, relevant content. If no new helpful information is identified, the model returns the existing cache; if no cache is available, it explicitly indicates that no useful content was found. The inputs to this process include the user prompt p, the newly retrieved documents D_i, and the accumulated memory cache.

The use of an information cache mitigates information forgetting by retaining relevant content across retrieval steps, enabling more coherent multi-hop reasoning. Meanwhile, explicitly extracting key information from retrieved documents addresses ineffective information selection, helping the model focus on information that is most useful for answering the question.

This approach allows us to assess whether providing a concise representation of retrieved information, combined with a cache of previously relevant context can improve answer quality and reduce redundant retrievals, without modifying the underlying model. By decoupling information extraction from reasoning, it introduces an agentic component that enables more structured, context-aware inference and better utilization of retrieved knowledge across steps.

3.2 De-duplication of retrieved documents

To investigate the causes of duplicate search query generation and evaluate the effect of retrieval redundancy on model performance, we introduce a De-duplication module that filters out documents retrieved in previous steps. This approach tests the hypothesis that the model generates repeated queries because it deems the initially retrieved information insufficient for the task. By preventing repeated access to the same content, this module encourages the model to incorporate a broader set of documents throughout its reasoning process. Specifically, when a retrieved document is discarded as a duplicate, the system is forced to consider the next-highest-ranked passage from the retriever's full ranked list. This effectively allows the model to continue to explore parts of the document collection that did not appear in the top-k results of previous turns, thereby increasing the diversity of the information it considers.

At the retrieval step of each turn, $k = 3$ documents D_i are returned. However, these documents might already have been seen. In this approach, Figure 1 (De-duplication), we maintain a set of unique document IDs for all passages seen during the reasoning process in previous turns for a given user prompt p. Any new retrieval that returns a document whose ID is already in this set is discarded and replaced by the next-highest-ranking, unseen document. The result is a set of unseen documents D_i'.

Ultimately, this allows us to examine whether reducing retrieval overlap leads to improved answer accuracy and fewer redundant search queries. If information forgetting is the cause of repeated retrievals, we expect the De-duplication pipeline to result in a drop in answer accuracy, as it means

the LLM no longer has access to information to answer the question correctly.

3.3 Hybrid

The hybrid approach combines the Contextualization with the De-duplication approaches to evaluate whether retaining extracted relevant information while enforcing retrieval diversity can jointly enhance reasoning performance. By integrating the contextualization module with non-redundant retrieval, this setup allows us to test whether the limitations of one component (e.g., information forgetting or redundancy) can be mitigated by the other, leading to improved answer accuracy and more efficient use of retrieved content.

4 Experiments

4.1 Setup

4.1.1 Data source

To enable direct comparison with Search-R1 (Jin et al., 2025), we evaluate on the same two benchmarks: HotpotQA (Yang et al., 2018) and Natural Questions (NQ) (Kwiatkowski et al., 2019). Since labeled test sets for these two datasets are not publicly available, we follow prior work and use the validation sets. Retrieval is performed on the 2018 Wikipedia dump with the E5 retriever.

4.1.2 Data splits

To reduce the cost associated with querying external LLMs, we created a smaller subset of question-answer pairs for evaluation. Specifically, we randomly sample 500 question-answer pairs from the HotpotQA (Yang et al., 2018) and NQ (Kwiatkowski et al., 2019) validation sets. This subset is solely used for evaluation; no hyperparameter tuning or training is performed using this subset. All reported metrics are based on this validation set.

4.1.3 Baselines

We utilize the already trained Qwen2.5-7B Search-R1-base (PPO) as our main baseline for comparison. While running inference with both Qwen2.5-3B Search-R1-base (PPO) and Qwen2.5-3B Search-R1-instruct (GRPO), the latter model exhibits difficulties in adhering to the structured output format specified by the Search-R1 framework. Inference outputs show frequent failures to generate required output tags such as <*think*> and <*search*> within the iterative reasoning loop. Additionally, the model often generates retrieved information under <*information*> tags by itself after the search query, and also occasionally fails to produce a final answer at the end of the reasoning chain. These behaviors indicate limitations in the ability to reliably follow instruction-guided formatting. Therefore, we only enhance the superior Qwen2.5-7B Search-R1-base (PPO) model to test our modules and report the corresponding results in Table 1. For all approaches, we run inference on our validation dataset of 500 questions and compute exact match, LLM Match, and the average number of turns.

4.1.4 Implementation details

We built on top of the publicly available Search-R1 source code on GitHub, where we make modifications to the model prompt to optimize the model's behavior. For inference on the trained model, we use the HuggingFace transformer library to perform forward pass, then run the Wikipedia article chunk E5 dense retriever provided in the Search-R1 GitHub repository. Contextualization and LLM Match evaluation are computed by GPT-4.1-mini via the OpenAI API. All Search-R1 model inference is run locally on a single NVIDIA A100 (80GB) GPU. Our code is publicly available at `https://github.com/a1bhinav/test-time-agentic-rag`.

4.1.5 Evaluation Metrics

The overall performance of our model is reported as the exact match (EM), just as in Search-R1. An analysis of the Search-R1 baseline reveals several false negatives where the predicted answer string does not exactly match the golden answer, despite referring to the same underlying entity, a discrepancy that is easily recognizable to human evaluators (Examples, "2" and "Two", "950 Pesos" and "P950"). In order to scale up these judgments, we prompt an external LLM model (OpenAI GPT-4.1-mini) to evaluate whether the predicted answer matches the golden answer. We call this evaluation metric LLM Match, and we present it in addition to the Exact Match metric in the results section.

For LLM Match, the model is given both the predicted answer and a set of ground truth answers, and is instructed to determine whether the predicted answer is semantically equivalent to any of the gold answers. Minor differences in phrasing are permitted as long as the predicted answer conveys the same meaning.

The prompt directs the model to assign a binary score:

- **1** if the predicted answer is semantically equivalent to the ground truth, and
- **0** if it is incomplete or diverges in meaning.

The evaluation explicitly focuses on semantic similarity, independent of factual correctness. This setup enables scalable, consistent semantic evaluation without human annotators. In addition, since we are focused on retrieval efficiency, we report the average number of retrievals. It is important to consider this metric in the context of the EM score as the model can drive the number of retrieval iterations to 0 by always hallucinating an answer and performing no retrieval.

4.2 Results

Table 1: Performance comparison of our proposed modules against the Search-R1 baselines on our 500-question evaluation set. We report Exact Match, LLM Match, and the average number of turns. Our modules are applied as test-time enhancements to the Qwen2.5-7B base (PPO) model. Cont: Contextualization (Section 3.1). De-Dup: De-duplication (Section 3.2). Hybrid (Section 3.3). Best results for our approaches are in bold.

Variant	Exact Match	LLM Match	avg. # searches
Qwen2.5-3B Search-R1			
base (PPO)	0.292	0.356	1.410
instruct (GRPO)	0.310	0.396	2.054
Qwen2.5-7B Search-R1			
base (PPO)	0.464	0.538	2.392
base (PPO) w/ Cont **(Ours)**	**0.490**	**0.574**	**2.142**
base (PPO) w/ De-Dup **(Ours)**	0.478	0.560	2.498
base (PPO) w/ Hybrid **(Ours)**	0.480	0.568	2.154

In terms of answer accuracy, the Contextualization approach achieves a 5.6% increase in EM and a 6.7% increase in LLM Match score compared with the Search-R1 baseline. It also issues the fewest retrievals, reducing the average from 2.392 in the baseline to 2.142. While the De-duplication and Hybrid approaches have similar gains in EM and LLM Match over the baseline, only the Hybrid approach reduces the average number of retrievals, similar to the Contextualization. The De-duplication pipeline in fact issues more retrievals than the baseline (2.498 vs. 2.392 on average). Overall, the Contextualization approach still achieves the highest EM, LLM Match, and lowest average number of retrievals. All metrics along with the baseline are shown in Table 1. We note that this measure of efficiency reflects retriever load (number of dense retrieval calls) rather than end-to-end inference cost: Contextualization itself issues an additional LLM call per turn, so fewer retrievals do not directly imply lower runtime or token consumption.

5 Conclusion

In this work, we implemented and evaluated three inference-time enhancements to the Search-R1 pipeline: (1) a Contextualization module, (2) a De-duplication module for retrieved documents, and (3) a Hybrid approach combining both. All three approaches improve the answer accuracy of the Search-R1 framework that serves as our baseline. Our Contextualization module also reduces the average number of retrieval turns, while the De-duplication module increases it. The Hybrid approach reduces retrieval count as well, though less than Contextualization alone.

Limitations

This work has several limitations. We evaluate on a 500-question validation subset from HotpotQA and Natural Questions, both of which predate current LLMs and may be affected by training-data contamination. Smaller Search-R1 variants did not reliably follow the required inference format, limiting us to a single backbone, so the observed gains may not generalize to other datasets, domains, retrievers, retrieval depths, or agentic RAG systems. The contextualization module and LLM Match metric rely on GPT-4.1-mini, and our efficiency analysis measures retrieval turns rather than end-to-end latency, monetary cost, or total token usage. We did not ablate the choice of contextualizer or LLM judge, both of which are natural directions for future work. Extending our test-time modules to other agentic RAG systems (e.g., Search-o1, RAG-RL) and to training-time interventions is also left for future work.

References

Zhiyu An, Xianzhong Ding, Yen-Chun Fu, Cheng-Chung Chu, Yan Li, and Wan Du. 2024. Golden-retriever: High-fidelity agentic retrieval augmented generation for industrial knowledge base. *Preprint*, arXiv:2408.00798.

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

Jerry Huang, Siddarth Madala, Risham Sidhu, Cheng Niu, Julia Hockenmaier, and Tong Zhang. 2025. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *Preprint*, arXiv:2503.12759.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025. Search-o1: Agentic search-enhanced large reasoning models. *Preprint*, arXiv:2501.05366.

Yunxiao Shi, Xing Zi, Zijing Shi, Haimin Zhang, Qiang Wu, and Min Xu. 2024. Enhancing retrieval and managing retrieval: A four-module synergy for improved quality and efficiency in rag systems. *Preprint*, arXiv:2407.10670.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *Preprint*, arXiv:2501.09136.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

A Additional Analysis

We examine the outputs of the baseline Search-R1 and the De-duplication approach to determine the source of this increase in retrieval count. We observed that the Search-R1 baseline is more likely to stop pursuing the same search objective when its searches return duplicate documents, as these documents offer no new information. In contrast, the De-duplication approach only returns new documents, causing the model to continue generating similar search queries in an effort to gather more context for the current search objective. This behavior leads to an increased average number of queries in the De-duplication approach. However, the additional context is rarely helpful, as the necessary information is often already present in the initial retrieval but fails to be extracted by the model, resulting in only a small improvement in answer accuracy.

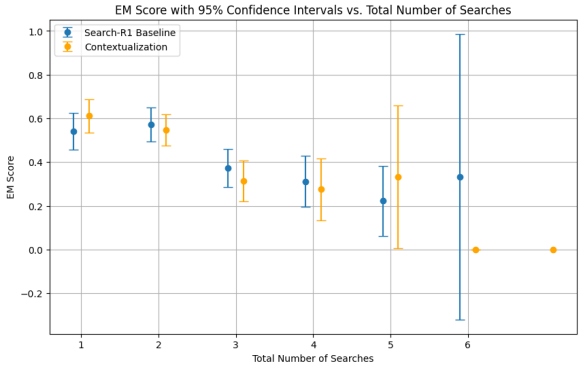

Figure 2: Exact Match (EM) score for the Search-R1 baseline and the Contextualization module conditioned on the total number of searches performed. Questions for which the model performs more searches tend to have lower EM, suggesting that higher retrieval count is associated with more difficult or less confidently answered cases. The overlapping 95% confidence intervals indicate that differences within individual search-count buckets should be interpreted cautiously.

Figure 2 shows a 95% confidence interval around exact match score of the Search-R1 baseline and the Contextualization pipeline, conditioned on the total number of searches performed. While the difference between the two never appears statistically significant, we observe a downward trend in

both. This suggests that exact match is negatively correlated with the number of retrievals.

For the Search-R1 baselines and the three test-time approaches, the LLM match is 16 to 18% greater than the exact match score. Investigating the outputs where the LLM determines the golden and predicted answers match but exact match fails, we observe two common patterns: numerical answers and shortened names or abbreviations.

B Prompts

This appendix lists the prompts used for the contextualization module (Section 3.1) and the LLM Match metric (Section 4.1.5).

B.1 Contextualization Prompt

After each retrieval step, the contextualization LLM is given the system prompt below. The query, accumulated cache, and newly retrieved documents are then appended as the user message.

```
You are a helpful information
extractor. Given a query inside
<query> and </query>, a previous
cache inside <previous_cache> and
</previous_cache>, and retrieved
snippets inside <information> and
</information>, preserve all useful
information already in the previous
cache and add any new information
helpful for answering the query.
Output the updated information inside
<cache> and </cache>.

If the previous cache contains useful
information but no new relevant
information is found, output the
previous cache unchanged inside
<cache> and </cache>. If no helpful
information is found and the previous
cache is empty, output:
<cache>
No helpful information found
</cache>.

Only output <cache> and </cache>
with helpful information inside and
nothing else.
```

The user message uses the template:

```
<query>
{question}
</query>
<previous_cache>
{cache}
</previous_cache>
<information>
{information}
</information>
```

B.2 LLM Match Prompt

The LLM-as-a-Judge used for the LLM Match metric is given the system prompt below, followed by the predicted answer and the ground-truth answer(s) as the user message.

```
You are an expert evaluator to judge if
the model's predicted answer correctly
matches any of the ground truth answer.
Carefully consider the meaning and
correctness. Minor variations in wording
are acceptable if the predicted answer
conveys the same information as the
ground truth.

Given:
The predicted answer.
The ground truth answer.

Task:
Score 0 or 1 based on the semantic
similarity of predicted and ground truth
answer.

Notes:
Evaluate only semantic similarity, not
factual correctness.

Scoring:
0 - The predicted answer is not equivalent
    to ground truth, incomplete in a
    meaningful way, or diverges in
    meaning from the ground truth.
1 - The predicted answer is semantically
    similar or conveys the same meaning
    as ground truth.

Output:
Score: X
Justification: [Concise 1-2 sentence
justification]
```

The user message follows the template:

```
Predicted Answer: {prediction}
Ground Truth Answers: {gold_answers}
```