Neurocomputing 695 (2026) 134047

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

ReflectRAG: Enhancing retrieval-augmented generation with GRPO-optimized iterative reflection

Xuanhe Chen [a], Yuchen Li [b,c,**], Youyi Bi [a,*], Shuaiqiang Wang [b], Linghe Kong [c], Dawei Yin [b]

[a] Global College, Shanghai Jiao Tong University, Shanghai, China
[b] Baidu Inc., Beijing, China
[c] School of Computer Science, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Communicated by D. Guo

Keywords:
Retrieval-augmented generation
Question answering
Reflection mechanism
Reinforcement learning

ABSTRACT

Retrieval-Augmented Generation (RAG) enhances question answering (QA) by grounding responses in external documents, yet it struggles with irrelevant retrieval and single-pass generation errors. To address these issues, we propose **ReflectRAG**, a framework that advances RAG through structured planning, iterative reflection, and reinforcement learning (RL). Specifically, ReflectRAG first retrieves and re-ranks documents and generates answers guided by fact-based plans. Then, ReflectRAG leverages a reflection module to iteratively critique and refine plans and answers, and optimizes the reflection module via RL. By integrating planning, reflection, and RL, ReflectRAG ensures precise, contextually aligned answers, offering a robust solution for open-domain QA. To demonstrate the effectiveness of our proposed model, we carry out extensive experiments on MS MARCO v2.1, TriviaQA (TQA), and Natural Questions (NQ). The comparative results on three open-domain datasets show that ReflectRAG achieves up to **+0.31 absolute F1 points** over the BASE RAG baseline across MS MARCO v2.1, TriviaQA, and NQ, establishing ReflectRAG as an effective solution for enhancing factual accuracy and coherence in QA systems. The source code is available at https://github.com/uncxlirh/ReflectRAG.

1. Introduction

Retrieval-Augmented Generation (RAG) has gained prominence in open-domain question answering by anchoring generated responses in externally retrieved documents [1]. However, several issues hamper the effectiveness of current RAG systems. Models frequently generate hallucinated or incoherent answers when confronted with irrelevant or redundant documents, a shortcoming that becomes particularly problematic when answers require precision and factual accuracy. The prevalent single-pass generation approach further limits the opportunities for error detection and correction, thereby compromising the reliability and completeness of responses.

Current methods endeavor to tackle these challenges with different strategies. Traditional retrieval techniques, such as BM25 [2], have been effectively used to select candidate documents based on term frequency and inverse document frequency [3]. However, while BM25 delivers a computationally efficient baseline, its reliance on surface-level matching often fails to capture deeper semantic nuances. In contrast, dense retrieval methods like DPR employ contextual embeddings for a more refined selection [4]. To enhance the quality of the retrieved candidates, re-ranking strategies have been incorporated using state-of-the-art large language models [5,6]. Beyond open-domain QA, multi-stage retriever–reranker pipelines have also been systematically studied in claim retrieval for fact-checking, highlighting the practical value of combining efficient retrieval with neural re-ranking [7]. Despite these advancements, most existing pipelines remain confined to a single-pass generation process, which provides no systematic mechanism for correcting initial errors or omissions.

At the same time, a more fundamental blind spot lies in the absence of a structured organization of the retrieved content before generation. Mainstream RAG pipelines typically concatenate the top-k passages and prompt the LLM to respond, neglecting the necessity of explicit planning. Without such structured planning, models struggle to distinguish key information from noise. Although some studies have introduced iterative self-refinement mechanisms or verification loops [8,9], these modules are generally appended as post-hoc patches and operate independently

* Corresponding author.
** Corresponding author at: Baidu Inc., Beijing, China.
Email addresses: cxhjaccount@sjtu.edu.cn (X. Chen), yuchenli@sjtu.edu.cn (Y. Li), youyi.bi@sjtu.edu.cn (Y. Bi), wangshuaiqiang@baidu.com (S. Wang), linghe.kong@sjtu.edu.cn (L. Kong), yindawei02@baidu.com (D. Yin).

https://doi.org/10.1016/j.neucom.2026.134047
Received 9 January 2026; Received in revised form 3 April 2026; Accepted 20 May 2026
Available online 21 May 2026
0925-2312/© 2026 Elsevier B.V. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

from earlier stages, failing to integrate planning, generation, and refinement into a unified and coherent pipeline.

To address these limitations, we introduce ReflectRAG. Unlike existing corrective or self-reflective RAG methods that append verification as a post-hoc step, ReflectRAG integrates three tightly coupled components into a unified pipeline. First, a *planning* stage organizes retrieved evidence into structured, fact-based representations before generation begins—addressing the noise-sensitivity problem at its source rather than after an answer has already been committed. Second, an *iterative reflection* module critiques draft answers against the retrieved passages and proposes targeted revisions, providing the systematic error-correction mechanism that single-pass pipelines lack. Third, a *reinforcement learning* stage optimizes the reflection model via Group Relative Policy Optimization (GRPO) [10], using evaluation metrics such as F1, ROUGE-L, and BERTScore [11] as training-time reward signals to ensure that learned critiques translate into measurable answer improvements. We conduct extensive experiments to demonstrate the effectiveness of ReflectRAG. Experimental results on MS MARCO v2.1, TriviaQA, and Natural Questions show significant gains, including consistent absolute F1 improvements, alongside improved ROUGE-L and BERTScore. ReflectRAG thus provides a practical enhancement to neural QA pipelines used in information-centric applications such as search and conversational systems.

The main contribution of this work can be summarized as follows:

- We study the problem of hallucinated and incoherent outputs in RAG and propose a unified framework, ReflectRAG, which utilizes fact-based planning to organize retrieved content, addressing the limitations of unstructured single-pass generation.
- We design an iterative reflection module that critically evaluates and refines outputs, and we optimize this module using Grouped Relative Policy Optimization (GRPO) to align the reflection process with downstream evaluation metrics.
- We conduct extensive experiments to demonstrate the effectiveness of ReflectRAG. Experimental results on MS MARCO v2.1, TriviaQA, and Natural Questions show consistent gains—up to $+0.31$ absolute F1—alongside improved ROUGE-L and BERTScore.

2. Related work

In this section, we review the literature pertinent to our proposed framework, focusing on three key areas: Retrieval-Augmented Generation (RAG) with structured reasoning, semantic document re-ranking, and reflection mechanisms optimized via reinforcement learning.

2.1. Retrieval-augmented generation and structured planning

Retrieval-Augmented Generation (RAG) has established itself as a cornerstone for addressing hallucinations in Large Language Models (LLMs) [1,12]. While early architectures like DPR [4] and FiD [13] focused on the mechanics of retrieving and fusing passages, recent advancements have shifted towards *active* and *corrective* strategies. For instance, Self-RAG [14] trains the generator to output critique tokens to assess retrieval quality, while CRAG (Corrective RAG) [15] introduces a lightweight evaluator to trigger web searches. Similarly, methods like Active RAG (FLARE) [16] dynamically decide when to retrieve based on generation confidence.

Despite these innovations, most RAG systems still employ a "retrieve-then-generate" paradigm. Inspired by Chain-of-Thought (CoT) [17] and Plan-and-Solve prompting strategies [18], our work introduces an explicit *planning* stage into the RAG pipeline. This aligns with recent trends in approaches that interleave retrieval with multi-step reasoning [19], where structured intermediate representations serve to filter noise and anchor the generation process. Recent neurocomputing work such as RECoT selects relation-consistent CoT sentences to drive next-hop retrieval and improve multi-hop QA performance [20]. Additionally, work on declarative optimization of LLM-based pipelines such as DSPy [21] motivates our modular design, where planning and reflection are treated as improvable components within the RAG workflow.

2.2. Semantic document re-ranking

Retrieval precision is a bottleneck for RAG performance. While sparse methods like BM25 remain an efficient baseline, they lack semantic understanding. Neural re-rankers, such as monoBERT [3], mark a significant leap by modeling query-document interactions.

With the emergence of general-purpose LLMs, generative re-ranking has become the state-of-the-art. RankGPT [5] leverages the zero-shot reasoning capabilities of LLMs to sort passages, and LLM-RankFusion [6] further improves robustness by integrating multiple ranking perspectives, similar to generative fusion strategies like LLM-Blender [22]. However, deploying LLMs for full-scale re-ranking is computationally prohibitive. In ReflectRAG, we adopt a pragmatic hybrid strategy: utilizing efficient BM25 retrieval for high recall followed by LLM-based re-ranking on a constrained candidate pool.

2.3. Iterative reflection and reinforcement learning

Self-correction is essential for mitigating errors in open-ended generation. Techniques like Self-Refine [8] and process-supervision methods like *Let's Verify* [9] demonstrate that LLMs can improve their outputs through iterative feedback and step-by-step verification. Recent advances like Reflexion [23] employ verbal reinforcement learning to verify outputs, sharing similarities with verification approaches such as Chain-of-Verification (CoVe) [24] and CRITIC [25].

However, relying solely on prompting is often unstable. To address this, Reinforcement Learning (RL) has been widely adopted to align models with task objectives. In Neurocomputing, RL-based training has also been explored to adapt text generation under low-resource settings [26]. Rooted in the foundational RLHF frameworks like InstructGPT [27], recent methods like Direct Preference Optimization (DPO) [28] have simplified alignment by optimizing a classification loss. Recently, Group Relative Policy Optimization (GRPO) [10] has emerged as a more efficient online alternative. ReflectRAG leverages GRPO to optimize a lightweight reflection module, creating a cost-effective feedback loop that maximizes downstream metrics.

3. Methodology

ReflectRAG is a novel framework that enhances RAG for QA by incorporating *structured plan-based generation*, an *iterative reflection* mechanism, and *RL* to refine answers. This section describes the overall process, organized into three core components: (1) *Document Retrieval & Re-ranking*, (2) *Answer Generation Planning*, and (3) *Reflection & Refinement Learning (RL)-based Fine-tuning*.

3.1. Overall framework of ReflectRAG

As illustrated in Fig. 1, ReflectRAG consists of three steps: (1) *Document Retrieval & Re-ranking*, (2) *Answer Generation Planning*, and (3) *Reflection & RL-based Fine-tuning*. Specifically, given an input query q, in Step (1), ReflectRAG first leverages a retriever to identify a set of candidate passages from a large corpus, and then re-orders retrieved passages using a re-ranker to ensure semantic relevance to the input query. This selection process yields a concise and focused context for the generator, improving both precision and efficiency in the downstream workflow. Rather than generating answers directly from the selected passages, in Step (2), ReflectRAG first constructs a structured plan that extracts key facts and outlines intent, and then passes the plan to a generator, which produces an initial draft of the answer. The separation

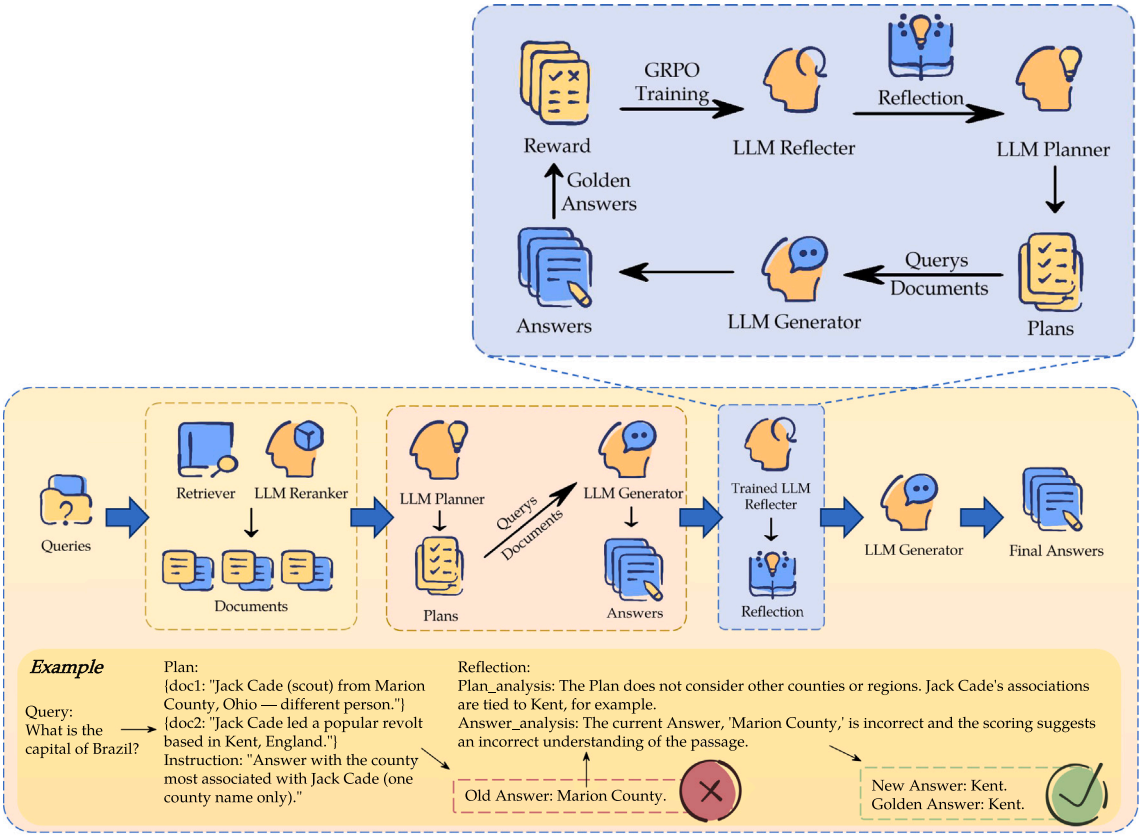

Fig. 1. The pipeline of the ReflectRAG consists of three steps: (1) *Document Retrieval & Re-ranking*, (2) *Answer Generation Planning*, and (3) *Reflection & Refinement Learning (RL)-based Fine-tuning*. GRPO training is performed offline to optimize the reflection module.

between content planning and realization improves controllability and helps mitigate irrelevant or unsupported content. Finally, to further refine the draft, ReflectRAG employs an iterative module to examine the query, plan, and answer, and proposes targeted revisions in Step (3). These are used to update the input to the generator over multiple cycles. The behavior of the iterative module is further improved through reinforcement learning, which encourages suggestions that lead to more accurate and complete answers. In addition, several lightweight filtering and optimization mechanisms are incorporated into the reflection module to improve stability and factual consistency. In this way, ReflectRAG can mitigate common RAG pitfalls (i.e., irrelevant retrieval and single-pass hallucinations) by systematically refining both the retrieval stage and the final answer.

3.2. Document retrieval and re-ranking

Retrieval. In the first step, ReflectRAG retrieves candidate passages with a retriever. Concretely, given a query q, ReflectRAG first indexes documents in the corpus and retrieves the top-k documents. ReflectRAG assigns a score to each candidate document as

$$\text{Score}(q,d) = \sum_{t \in q} \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5}\right) \cdot \frac{(k_1+1) f(t,d)}{k_1\left(1 - b + b\frac{|d|}{\text{avgdl}}\right) + f(t,d)}, \quad (1)$$

where N is the corpus size, and $n(t)$ is the document frequency of term t. $f(t,d)$ is the term frequency, and $avgdl$ represents the average document length. k_1 and b are parameters.

Re-ranking. To address potential term-matching gaps, ReflectRAG re-ranks the top-k retrieved passages using a reranker via a JSON-based prompt containing *(query + passages)*. Then, ReflectRAG outputs a list of indices, which indicate passage importance. Empirically, re-ranking selects more semantically relevant passages, e.g., focusing on the exact entity or date required by the question. We utilize a JSON-based prompt containing *(query + passages)* to instruct the model. An example of the prompt structure is shown below:

Example Re-ranking Prompt (JSON)

```
{ "query": "Who is the CEO of Open AI?", "passages": [ {"id": 0, "content": "Sam Altman is the CEO..."}, {"id": 1, "content": "OpenAI was founded in..."} ], "instruction": "Output a list of passage IDs in descending order of relevance." }
```

3.3. Answer generation and planning

ReflectRAG supports two generation modes: (1) *Direct Generation*. ReflectRAG answers the query directly from the top-k re-ranked passages. (2) *Plan-based Generation*. ReflectRAG first composes a *Plan* summarizing the salient facts or key points from each passage, then uses these structured facts to guide the final answer construction. For instance, given a query "What is the capital of Brazil?", ReflectRAG composes a plan as follows:

Example Plan (JSON)

```
{
    "query": "What is the capital of Brazil?",
    "plan": [
    {"doc_id": 1, "fact": "Brasília is the capital."},
    {"doc_id": 2, "fact": "It replaced Rio in 1960."}
    ],
    "instruction": "Mention the capital and date."
}
```

In this way, ReflectRAG fosters improved traceability and reduces extraneous information. Then, ReflectRAG generates the final answer according to the instructions in the plan.

3.4. Reflection & RL-based fine-tuning of the reflection model

Reflection mechanism. Although plan-based generation mitigates some hallucinations, single-pass outputs may still miss vital details. Therefore, ReflectRAG proposes an iterative reflection stage using a reflection model. During offline GRPO training, the reflection model is conditioned on: (1) *query*, (2) *current plan*, (3) *current answer*, (4) *gold answer(s)*, and (5) *evaluation score*—the latter two serving exclusively as training-time supervision signals. At inference time, the trained reflection model operates without access to gold answers or gold-derived scores; it relies solely on the query, the current plan, the current answer, and the retrieved passages to generate passage-grounded critiques and suggestions. For instance, given a query "Missing the city name.", ReflectRAG produces a JSON reflection as follows:

Example Reflection (JSON)

```
{
    "plan_analysis": "Missing the city name.",
    "answer_analysis": "Only mentions 'South America'.",
    "suggestions": [
    "Add 'Brasília'",
    "Incorporate date of capital relocation"
    ],
    "instruction": "Mention the capital and date."
}
```

Then, ReflectRAG feeds these suggestions back into the generator to refine both the plan and the answer. In this work, ReflectRAG limits reflection loops to three iterations or until no further improvements are observed. Empirically, by repeatedly spotting factual gaps, ReflectRAG can yield higher precision.

RL fine-tuning. Static reflection helps but may yield generic or unhelpful critiques. We therefore *fine-tune the reflection model* with a *Group Relative Policy Optimization (GRPO)* objective that directly prefers reflections leading to better downstream answers.

Let a *group* $\mathcal{G}_q = \{(x_k, y_k, r_k)\}_{k=1}^{n_q}$ collect multiple candidate reflections for the same query q (*same* prompt, different reflection outputs). We compute a scalar reward as the improvement in task score between the refined and the previous answer:

$$r_k \;=\; \Delta\mathrm{F1}_k \;=\; \mathrm{F1}(a_k^{\mathrm{new}}, g) \;-\; \mathrm{F1}(a_k^{\mathrm{old}}, g), \tag{2}$$

where g denotes the gold answer(s). This reward signal, and the gold answer on which it depends, is used exclusively during offline GRPO training. At inference time, the trained reflection model generates critiques conditioned only on the query, current plan, current answer, and retrieved passages—no gold answer or gold-derived score is available or required.

For each group, we form all ordered pairs $\mathcal{P}_q = \{(i,j) \mid r_i > r_j\}$ and train the reflection model π_θ with a pairwise preference loss over the *response segment* likelihoods. Let ℓ_k be the average log-likelihood of the generated reflection tokens under π_θ (conditioned on the prompt), then the GRPO loss is

$$\mathcal{L}_{\mathrm{pair}} \;=\; \mathbb{E}_{(i,j)\in\mathcal{P}_q}\Big[-\log\sigma\Big(\frac{\ell_i - \ell_j}{\tau}\Big)\Big], \tag{3}$$

where $\sigma(\cdot)$ is the logistic function and τ is a temperature. To bound drift, we add a lightweight KL penalty against a frozen reference policy π_{ref} computed *only* on the response tokens:

$$\mathcal{L}_{\mathrm{KL}} \;=\; \lambda\,\mathbb{E}\big[\mathrm{KL}\big(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x)\big)\big]. \tag{4}$$

Algorithm 1 GRPO fine-tuning of the reflection model (offline training; gold answers are used only to compute the reward signal in this phase).

Set maximum reflection steps $T_{\max} = 3$
for each group $\mathcal{G}_q = \{(x_k, y_k, r_k)\}_{k=1}^{n_q}$ (same query q) **do**
  Compute new logits with trainable π_θ and ref logits with frozen π_{ref}.
  Compute per-sample average log-likelihood on response tokens ℓ_k.
  Build $\mathcal{P}_q = \{(i,j) \mid r_i > r_j\}$; if empty, skip pair loss for this group.
  $\mathcal{L}_{\mathrm{pair}} \leftarrow \frac{1}{|\mathcal{P}_q|}\sum_{(i,j)\in\mathcal{P}_q} -\log\sigma\big((\ell_i - \ell_j)/\tau\big)$
  $\mathcal{L}_{\mathrm{KL}} \leftarrow \lambda \cdot \mathrm{AvgKL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$ on response tokens
  $\mathcal{L} \leftarrow \mathcal{L}_{\mathrm{pair}} + \mathcal{L}_{\mathrm{KL}}$; update θ by AdamW
end for

The final objective is

$$\mathcal{L} \;=\; \mathcal{L}_{\mathrm{pair}} + \mathcal{L}_{\mathrm{KL}}. \tag{5}$$

Stabilizing and normalizing. We skip reflection when critiques are unlikely to help (e.g., generic feedback or straightforward yes/no/factoid queries) and otherwise request a brief passage-grounded answer. For binary questions, we deterministically normalize to `{yes, no}` to ensure consistent scoring. For definition-type queries, we use a minimal-definition mode that outputs a single-sentence canonical definition or acronym expansion without hedging or examples. These controls reduce variance and remain active during GRPO. At inference time, the reflection model relies solely on passage-grounded self-critique without any oracle signals; the filtering and normalization rules described above—together with an answer-convergence early-stopping criterion—constitute the only heuristic controls applied during deployment.

Training details. We train on groups of candidate reflections built per query ID. Implementation uses 4-bit NF4 quantization with Low-Rank Adaptation (LoRA) [29] on top of `gemma-2-2b-it`. The quantization is employed to reduce memory overhead, allowing for larger batch sizes on consumer-grade GPUs, while LoRA parameters ($r = 16, \alpha = 32$, dropout $= 0.05$) are selected based on standard efficient fine-tuning practices to balance trainable parameter count and model expressivity. The reference policy π_{ref} is frozen at the LoRA initialization to prevent the model from deviating significantly from the base language distribution (reward hacking). The batch size is set to 16 using a group-by-query sampler to ensure stable gradient estimation for the group-relative loss. Detailed hyperparameters are listed in Table 1.

4. Experiments

4.1. Experimental setup

In this section, we conduct extensive experiments to demonstrate the effectiveness of ReflectRAG. We first detail three open-domain QA datasets, and then introduce evaluation metrics and experimental setups. Finally, we report the comparative results and present a comprehensive qualitative analysis.

4.1.1. Datasets

In our offline experiments, we choose the following three open-domain QA datasets:

- **MS MARCO v2.1** [30] contains over 1M real-world Bing queries with 8.8M passages. In our offline evaluation, we randomly sample 1000 queries for short-answer evaluation. To assess evaluation stability, we report results under two independent random seeds and provide the mean ± standard deviation in Appendix Table A.5.
- **TQA** [31] consists of 95K trivia questions with Wikipedia/web evidence. We select 1000 queries for concise, factual answers.

Table 1
GRPO training and system hyperparameters.

Component	Setting
Backbone / PEFT	`gemma-2-2b-it` + LoRA (r = 16, $\alpha = 32$, dropout = 0.05)
Quantization	4-bit NF4, double-quant, FP16 compute
Batching	Group-by-query sampler; batch size 16 (single GPU)
Sequence budget	prompt 128, response 256 (total 384)
Optimizer / LR	AdamW, 1×10^{-5}; cosine schedule, warmup 100 steps
Epochs / Accum	1 epoch, grad accum 1
GRPO	temperature $\tau = 1.0$; pairwise loss Eq. (3)
KL regularization	$\lambda = 0.02$; response-only KL Eq. (4)
Reflection loop	$T_{\max} = 3$; early-stop on no gain
Reward	ΔF1 as in Eq. (2) (training only; not used at inference)

- **NQ** [32] comprises 307K Google search queries with Wikipedia answers. We sample 1000 queries for context-rich responses.

Each dataset provides human-annotated gold answers, with 1000 queries sampled to ensure consistency across diverse question types and domains.

4.1.2. Evaluation metrics

While recent automated frameworks like RAGAS [33] provide nuanced metrics, in our offline evaluation, we adopt three standard gold-reference metrics to strictly assess factual correctness:

- **F1 Score** [1] measures token-level precision and recall. It evaluates the exact match of predicted tokens against gold answers, balancing correctness and completeness.
- **ROUGE-L** [34] captures structural similarity. This metric computes the longest common subsequence between generated and reference texts, emphasizing syntactic coherence.
- **BERTScore** [11] assesses semantic similarity using RoBERTa-large embeddings. It leverages contextual embeddings to capture deep semantic alignment, robust to lexical variations.

4.1.3. Model variants and baselines

To systematically evaluate the proposed framework, we design three progressive configurations. It is worth noting that ReflectRAG is designed as a modular framework where the underlying Large Language Model (LLM) serves as a replaceable engine for generation and reflection.

- **BASE**: Represents the standard RAG pipeline involving retrieval, re-ranking, and direct answer generation, without any planning or reflection mechanisms.
- **PLAN**: Builds upon BASE by incorporating the structured plan-based generation module (Section 3.3).
- **REFLECT**: Represents the full ReflectRAG framework, which extends PLAN by integrating the iterative reflection mechanism and RL-based fine-tuning (Section 3.4).

Backbone models. We adopt a multi-tiered evaluation protocol to verify both efficacy and robustness:

1. **Primary Evaluation (Standard)**: We utilize `gpt-3.5-turbo` as the backbone for all three datasets (MS MARCO, TQA, NQ). This setup serves to demonstrate how ReflectRAG can elevate a standard lightweight model to higher performance tiers.
2. **Generalization Evaluation (SOTA)**: To assess whether our framework provides diminishing returns or consistent gains on more capable models, we conduct experiments on MS MARCO using two state-of-the-art models:
   - **DeepSeek-R1-685B**: Representative of the SOTA in open-weight reasoning models [35], testing the framework's compatibility with strong Chain-of-Thought capabilities.
   - **GPT-4o**: Representative of the SOTA in closed-source general-purpose models [36], testing the framework's ability to refine an industry-leading baseline.

In addition to our in-house configurations, we compare with two recent reflective/agentic approaches under the same evaluation protocol: SelfRAG [14] and ReAct [37]. SelfRAG injects self-evaluation signals into retrieval and generation, whereas ReAct interleaves explicit reasoning traces with retrieval actions. Both are strong baselines for reflective QA. To ensure a fair comparison under a shared backbone, both SelfRAG and ReAct are implemented as GPT-3.5-turbo-based prompt adaptations that preserve the high-level mechanisms of the original methods. SelfRAG's relevance-support-refinement pipeline and ReAct's Thought→Action→Observation loop are each reproduced via structured prompts, with all components receiving the same BM25-retrieved and re-ranked passages as ReflectRAG. Key implementation details, including decoding parameters and normalization rules, are provided in Appendix Table A.5; prompt files and supporting resources are provided in the open-source repository directory.

The retrieval stage employs BM25 via Pyserini's `LuceneSearcher` ($k_1 = 1.2$, $b = 0.75$), indexing the full MS MARCO v2.1 corpus (8.8M passages) for MARCO experiments and Pyserini's prebuilt Wikipedia index (December 2018 snapshot) for TriviaQA and NQ. The top-10 retrieved passages are re-ranked by GPT-3.5-turbo (temperature= 0) using a JSON-structured prompt; the top-5 re-ranked passages are passed to the plan generation stage. All generation components use GPT-3.5-turbo (temperature= 0.2, max tokens= 256). Key implementation details, including GRPO hyperparameters and answer normalization rules, are provided in Appendix Table A.5.

4.2. Experimental results

We evaluate ReflectRAG on MS MARCO v2.1, TQA, and NQ, comparing our in-house configurations (BASE, PLAN, REFLECT) with two recent reflective/agentic baselines, SelfRAG and ReAct. The results, visualized in Fig. 2 and detailed in Table 2, highlight the effectiveness of ReflectRAG's components—re-ranking, planning, reflection, and RL—in enhancing open-domain QA performance.

4.2.1. Comparative results

Fig. 2 illustrates the comparative results of the raw F1 scores for BASE, PLAN, REFLECT, SelfRAG, and ReAct across MS MARCO v2.1, TQA, and NQ.

On MS MARCO v2.1, REFLECT achieves an F1 of 0.3120, outperforming PLAN (0.2633; +4.87 pts) and BASE (0.2405; +7.15 pts) and also edging out SelfRAG (0.3055; +0.65 pts); ReAct trails at 0.2850. This is evident in queries requiring precise details, such as "Who plays Leif on *Man with a Plan*?", where REFLECT corrects uninformative outputs like "No relevant snippet found." to specific answers like "Geoff Stults".

On TQA, REFLECT attains the largest gains with an F1 of 0.6863, surpassing PLAN (0.5665; +11.98 pts) and BASE (0.4830; +20.33 pts), and clearly outperforming SelfRAG (0.6520) and ReAct (0.4950).

On NQ, REFLECT reaches an F1 of 0.4410, exceeding PLAN (0.3276; +11.34 pts), BASE (0.1285; +31.25 pts), and both baselines—SelfRAG (0.4250; +1.60 pts) and ReAct (0.3540). Notably, while REFLECT also leads in ROUGE-L on NQ (0.4440), SelfRAG attains a slightly higher BERTScore (0.8960) than REFLECT (0.8930; −0.30 pts). These trends underscore that explicit planning already improves over standard RAG (PLAN > BASE), and adding iterative reflection with RL (REFLECT) yields consistent gains across datasets; compared to recent reflective/agentic baselines, REFLECT is overall strongest on token- and sequence-level metrics, while SelfRAG shows a small edge on embedding-based similarity in NQ.

Robustness across model capabilities. As shown in Fig. 3, ReflectRAG proves effective not only on weaker baselines but also on powerful SOTA models. For **DeepSeek-R1**, which naturally excels at reasoning

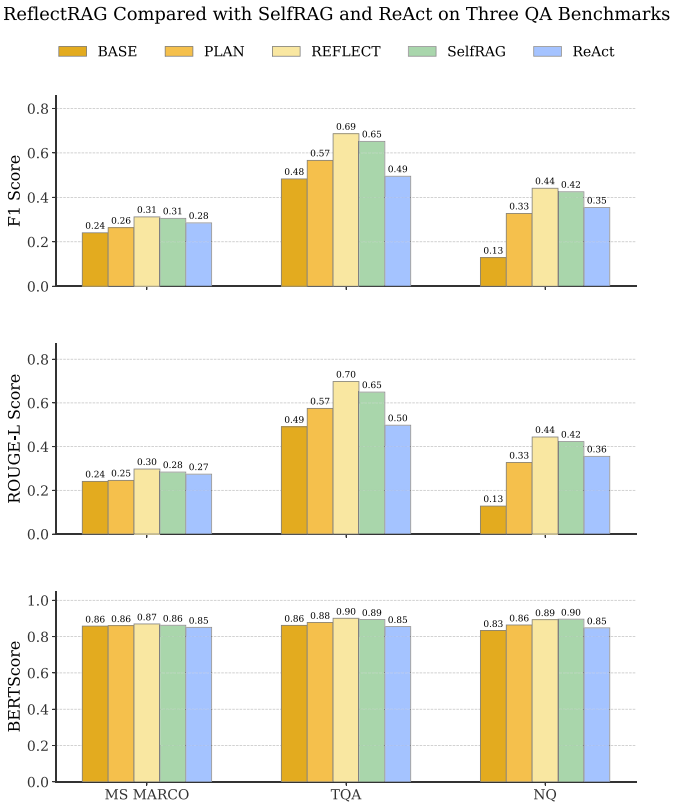

Fig. 2. Comparative results on F1, ROUGE-L, and BERTScore across three open-domain datasets (MS MARCO, TriviaQA, NQ) using GPT-3.5-Turbo as the backbone. ReflectRAG consistently outperforms the baselines.

Table 2
Comparison of BASE, PLAN, REFLECT, SelfRAG, and ReAct across MS MARCO, TQA, and NQ. Best results per metric are highlighted. *** denotes $p < 0.001$ (paired bootstrap, 10,000 iterations). SelfRAG and ReAct rows report corrected implementation results; see Appendix Table A.5 for key settings and the open-source repository directory for prompt files and supporting resources.

Dataset	Metric	BASE	PLAN	REFLECT	SelfRAG	ReAct
MS MARCO	F1	0.2405	0.2633	**0.3120*****	0.3055	0.2850
	ROUGE-L	0.2421	0.2458	**0.2975*****	0.2835	0.2740
	BERTScore	0.8220	0.8236	**0.8697*****	0.8625	0.8510
TQA	F1	0.4830	0.5665	**0.6863**	0.6520	0.4950
	ROUGE-L	0.4925	0.5768	**0.6985**	0.6495	0.4980
	BERTScore	0.8345	0.8486	**0.8961**	0.8940	0.8550
NQ	F1	0.1285	0.3276	**0.4410**	0.4250	0.3540
	ROUGE-L	0.1295	0.3304	**0.4440**	0.4230	0.3555
	BERTScore	0.8415	0.8698	0.8914	**0.8960**	0.8480

but often suffers from verbose outputs in short-answer tasks, ReflectRAG significantly boosts the F1 score (from 0.0789 to 0.2337). The reflection module effectively acts as a format constraint, aligning the model's extensive reasoning chains with concise, factual answer formats expected by the evaluation protocol. For **GPT-4o**, which establishes a high initial baseline (F1 0.3345), ReflectRAG still yields a consistent improvement (+0.0183 F1). This indicates that even for top-tier models, the structured planning and iterative critique mechanism can correct subtle hallucinations and semantic ambiguities that single-pass generation might overlook. This confirms that ReflectRAG is a model-agnostic framework capable of enhancing a wide spectrum of LLMs [38].

4.2.2. Ablation study

We ablate by comparing **BASE → PLAN → REFLECT** under identical retrieval pools and decoding budgets, and further conduct four targeted

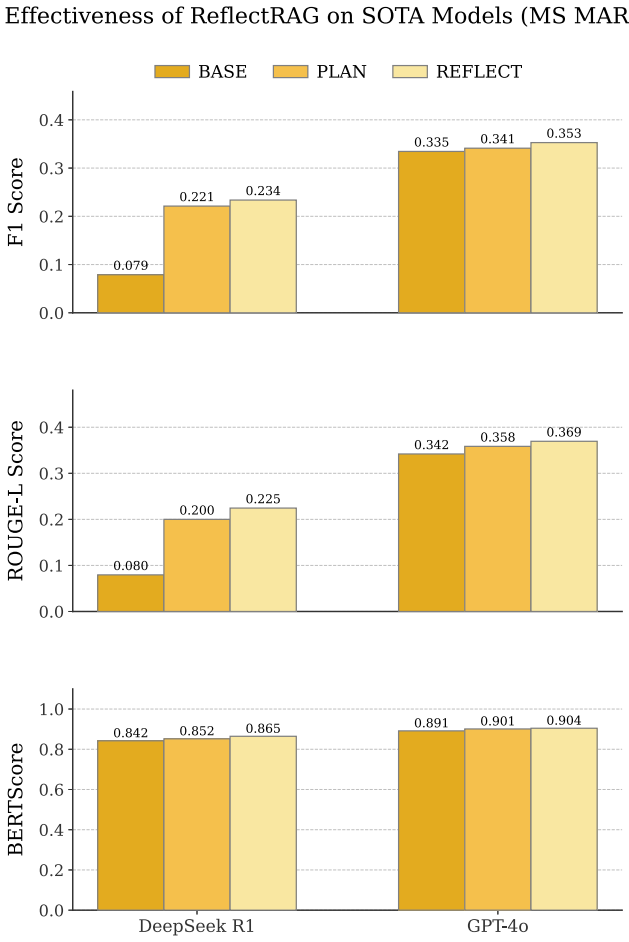

Fig. 3. Performance analysis on MS MARCO using state-of-the-art open-weights (DeepSeek-R1-685B) and closed-source (GPT-4o) models. ReflectRAG demonstrates robust improvements across different model capabilities.

Table 3
Ablation gains (absolute F1) under identical retrieval pools and decoding budgets. Full metrics are in Table 2.

Dataset	ΔPLAN–BASE	ΔREFLECT–PLAN	ΔREFLECT–BASE
MS MARCO	+0.0228	+0.0487	+0.0715
TQA	+0.0835	+0.1198	+0.2033
NQ	+0.1991	+0.1134	+0.3125
Avg	+0.1018	+0.0940	+0.1958

component ablations on MS MARCO to isolate the contributions of GRPO training, iterative refinement, and heuristic filtering. Full results are reported in Appendix Table A.7. Absolute F1 gains are summarized in Table 3: on average, PLAN adds **+0.1018** over BASE and REFLECT adds a further **+0.0940**, for **+0.1958** total over BASE. ROUGE-L and BERTScore show the same monotonic trend (see Table 2). These results indicate complementary roles—planning structures salient evidence and reduces noise, while reflection (GRPO-tuned) repairs residual omissions and inconsistencies—and, with the stabilization in Section 3.4, refinement improves accuracy without inflating length or variance.

4.3. Case study

To provide a deeper understanding of how ReflectRAG operates, we present a qualitative analysis of two distinct cases. The first case illustrates a successful correction of a reasoning error, while the second discusses a failure case to highlight current limitations.

4.3.1. Success case: correcting incomplete reasoning

In this case from the NQ dataset, the query asks for a specific actor, but the initial retrieval contains noise. Fig. 4 visualizes the reflection process.

Analysis. As shown in Fig. 4, the baseline model gave up because the entity "Leif" was not in the abstract. However, ReflectRAG's reflection module successfully critiqued the draft by cross-referencing the retrieved context again, explicitly pointing out the evidence in Document #3. This demonstrates that the reflection mechanism acts as a "second pair of eyes," reducing false negatives where the answer is present but overlooked.

4.3.2. Failure case: ambiguity and retrieval void

Despite its strengths, ReflectRAG is not immune to errors. We analyze a failure case regarding a temporal ambiguity.

Analysis. In this instance (Fig. 5), the reflection module ironically introduced a hallucination. The initial plan correctly identified 1973. However, the reflection model, trying to be "comprehensive," misinterpreted the semantic nuance between an "NBA Championship" and a "Conference Championship" mentioned in a retrieved document. This reveals a key limitation of iterative reflection: when retrieved passages contain semantically related but factually distinct distractors, the reflection model may override a correct initial answer with a plausible-but-wrong revision—a failure mode we term *semantic drift via over-reflection*. This failure mode is most likely when (a) the query involves a fine-grained semantic distinction (e.g., NBA championship vs. conference title), and (b) the retrieved passages contain multiple plausible candidate answers. Three design choices could mitigate this risk in future work: (1) limiting reflection to queries where the initial answer confidence is below a threshold, (2) requiring reflection suggestions to

Case Study 1: Success Analysis

Query: *"Who plays Leif on Man with a Plan?"*

Initial Attempt (BASE):
"No relevant snippet found."
(The model failed to extract the name due to scattered context.)

ReflectRAG (Step 1: Planning):
Plan: "Search for cast list of 'Man with a Plan'. Look for character 'Leif'."
Draft Answer: "The cast includes Matt LeBlanc, but Leif is not explicitly listed in the main summary."

ReflectRAG (Step 2: Reflection):
Critique: "The draft answer is evasive. The retrieved document #3 actually mentions 'Geoff Stults as Leif directly'. The plan missed this specific detail."
Suggestion: "Extract 'Geoff Stults' from document #3 and state it directly."

Final Answer (Refined):
"Geoff Stults plays Leif on Man with a Plan."

Fig. 4. A successful case where the reflection module identifies an overlooked detail in Document #3 that the initial planner missed.

Case Study 2: Failure Analysis

Query: *"When was the last time the Knicks won a championship?"*

Draft Answer:
"The New York Knicks won championships in 1970 and 1973."

Reflection Loop:
Critique: "The user asks for the *last* time. The answer lists two dates. Ensure 1973 is emphasized as the latest."
Revised Draft: "The last time was 1973."

Reflection Loop (Error):
Critique: "Wait, retrieval passage #2 mentions a 'conference title in 1999'. Is a conference title considered a championship? The answer might be 1999."
Suggestion: "Update to 1999 to include more recent wins."

Final Answer (Incorrect):
"The Knicks last won a championship in 1999."
(Gold Answer: 1973. The model confused 'Conference title' with 'NBA Championship'.)

Fig. 5. A failure case where over-reflection leads to semantic drift. The model confuses different types of "championships".

cite a specific passage index as evidence before accepting a revision, and (3) incorporating a dynamic early-stopping mechanism that halts reflection when the revised answer diverges substantially from the initial answer in embedding space.

4.3.3. Error taxonomy

To systematically characterize when and why ReflectRAG fails, we manually annotated all 380 incorrect predictions (F1 < 0.5) from the MS MARCO seed99 evaluation (1000 queries) and classified them into five categories. The distribution is summarized in Appendix Table A.3:

- **Wrong Entity** (40.0%). The model extracts an entity of the correct type but wrong identity (e.g., predicting a sibling's name instead of the queried person). This is the most frequent error category.
- **Semantic Drift via Over-reflection** (34.2%). Reflection overrides an initially correct answer with a plausible-but-wrong alternative drawn from a retrieved distractor. This is the most consequential failure mode (see Section 4.3).
- **Format Error** (21.6%). This includes yes/no format mismatches, verbose over-generation, and partial matches where correct information is embedded in imprecise phrasing.
- **Retrieval Gap** (2.6%). The retrieved passages do not contain the answer at all.
- **Unanswerable Conflation** (1.6%). The model produces a speculative answer for a genuinely unanswerable query instead of abstaining.

5. Conclusion

Our experiments confirm three consistent findings. First, structured planning alone (PLAN) improves over standard RAG by an average of +0.1018 absolute F1 across MS MARCO, TriviaQA, and NQ, demonstrating that organizing retrieved evidence before generation substantially reduces noise-induced errors. Second, adding GRPO-optimized iterative reflection (REFLECT) yields a further +0.0940 average F1 improvement, with the largest improvements on datasets requiring precise factual extraction (TriviaQA, NQ). Third, these gains generalize across model capabilities, from GPT-3.5-turbo to DeepSeek-R1 and GPT-4, indicating that the framework is model-agnostic.

One limitation of ReflectRAG is the increased inference cost introduced by the planning and reflection stages. On MS MARCO, the full pipeline consumes approximately 6945 tokens and 50.1 s of batch-averaged wall-clock time per query (Table A.7), with the reflection stage accounting for 66% of the total latency. The Plan-only configuration reduces this to 3470 tokens and 17.1 s while retaining a +0.102 average F1 improvement over BASE, and is recommended for latency-sensitive deployments. The wall-clock figures are dominated by API response time rather than algorithmic complexity; local deployment with continuous batching could substantially reduce end-to-end latency.

CRediT authorship contribution statement

Xuanhe Chen: Writing–original draft, Visualization, Software, Methodology, Data curation. **Yuchen Li:** Writing–review & editing, Supervision, Conceptualization. **Youyi Bi:** Writing–review & editing, Project administration, Funding acquisition. **Shuaiqiang Wang:** Supervision, Resources. **Linghe Kong:** Supervision, Resources. **Dawei Yin:** Supervision, Resources.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT (OpenAI) to assist with content organization (e.g., outlining sections) and to improve language and readability. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to acknowledge the financial support from the National Natural Science Foundation of China (52575300) and the National Key R&D Program of China (2022YFB4702400).

Appendix A. Additional tables

See Tables A.1, A.2, A.4 and A.6.

Table A.1
Training vs. Inference Signal Summary for the Reflection Module.

Signal	Training	Inference
Query	Yes	Yes
Current plan	Yes	Yes
Current answer	Yes	Yes
Retrieved passages	Yes	Yes
Gold answer(s)	Yes (reward only)	No
Evaluation score	Yes (reward only)	No

Table A.2
Complete Implementation Details.

Component	Setting
Retrieval	
Corpus (MARCO)	MS MARCO v2.1 passage corpus, 8.8M passages
Corpus (TQA/NQ)	Wikipedia (Dec 2018), Pyserini prebuilt index
Retrieval method	BM25 via Pyserini LuceneSearcher
BM25 parameters	$k_1 = 1.2$, $b = 0.75$
Top-k retrieval	10 passages
Re-ranking	
Re-ranking model	GPT-3.5-turbo
Re-ranking input	JSON: query + 10 passages
Decoding	temperature= 0, max retries= 3
Post-rerank selection	Top-5 passages
Plan & Answer Generation	
Generator model	GPT-3.5-turbo
Decoding	temperature= 0.2, top_p= 1.0, max tokens= 256
Reflection (Inference)	
Reflection model	gemma-2-2b-it + LoRA ($r = 16$, $\alpha = 32$, dropout= 0.05)
Quantization	4-bit NF4, double-quant, FP16 compute
Inference inputs	Query, plan, current answer, retrieved passages (no gold)
Decoding	temperature= 0.7, top_p= 0.9, max new tokens= 128
Iteration	$T_{max} = 3$; early-stop on answer convergence
GRPO Training (Offline)	
Optimizer / LR	AdamW, 1×10^{-5}; cosine schedule, 100 warmup steps
Batch size	16 (group-by-query sampler)
Epochs / Accum	1 epoch, grad accum= 1
GRPO temperature	$\tau = 1.0$
KL regularization	$\lambda = 0.02$; response-only KL
Reward	ΔF1 (training only)
Baselines	
SelfRAG backbone	GPT-3.5-turbo (gpt-3.5-turbo-0125)
SelfRAG structure	Relevance grading → support grading → conditional re-generation
SelfRAG decoding	temperature= 0.2, max tokens= 256, top_p= 1.0
SelfRAG bug fixed	Abstention output not suppressed
ReAct backbone	GPT-3.5-turbo (gpt-3.5-turbo-0125)
ReAct structure	Thought → Action → Observation loop (max 7 steps)
ReAct decoding	temperature= 0.0, max tokens= 512, stop = [Observation:]
ReAct bug fixed	Strict Finish[answer] parser replaced
Answer normalization	SQuAD-style (shared across all methods)

Table A.3
Error Taxonomy Summary (MS MARCO, seed99, 1000 queries, 380 errors with F1 < 0.5).

Error Category	Count	Proportion
Wrong Entity	152	40.0%
Semantic Drift (over-reflection)	130	34.2%
Format Error	82	21.6%
Retrieval Gap	10	2.6%
Unanswerable Conflation	6	1.6%
Total	**380**	**100%**

Table A.4
Targeted Ablation results on MS MARCO (seed99).

Configuration	F1	ROUGE-L	BERTScore
PLAN (no reflection)	0.2633	–	–
(a) Prompt-only reflection (no GRPO)	0.2790	0.2760	0.8760
(b) T= 1 / single-pass GRPO	0.3085 ± 0.0035	0.2955 ± 0.0018	0.8675 ± 0.0030
(c) T= 2	0.3092 ± 0.0041	0.2960 ± 0.0024	0.8678 ± 0.0029
(c) T= 3	0.3091 ± 0.0041	0.2959 ± 0.0023	0.8676 ± 0.0030
(d) No heuristic filter	0.2818	0.2781	0.8761
Full REFLECT (T= 3, GRPO, filter)	**0.3120**	**0.2975**	**0.8697**

Mean ± std reported for T= 1/2/3 (two seeds: seed99, seed77). Single-seed (seed99) results are reported for (a) and (d). Full REFLECT reports the seed99 result; the multi-seed mean is 0.3091 ± 0.0029 (see Table A.5). All results are on MS MARCO, 1000-query evaluation subset.

Table A.5
Multi-Seed evaluation results (F1).

Dataset	Config	Seed99	Seed77	Mean ± Std
MS MARCO	BASE	0.2405	0.2460	0.2433 ± 0.0028
	PLAN	0.2633	0.2686	0.2660 ± 0.0027
	REFLECT	0.3120	0.3062	0.3091 ± 0.0029
TQA	BASE	0.4830	0.4645	0.4738 ± 0.0093
	PLAN	0.5665	0.5480	0.5573 ± 0.0093
	REFLECT	0.6863	0.6623	0.6743 ± 0.0120
NQ	BASE	0.1285	0.1310	0.1298 ± 0.0013
	PLAN	0.3276	0.3348	0.3312 ± 0.0036
	REFLECT	0.4410	0.4460	0.4435 ± 0.0025

F1 scores reported. Each seed samples 1000 queries independently. The monotonic improvement BASE → PLAN → REFLECT is consistent across all seeds and datasets.

Table A.6
Bootstrap Significance Test (MS MARCO, seed99).

Metric	Improvement	95% CI	*p*-value
F1	+0.0487	[+0.0372, +0.0594]	< 0.001
ROUGE-L	+0.0517	[+0.0395, +0.0641]	< 0.001
BERTScore	+0.0461	[+0.0358, +0.0573]	< 0.001

Paired bootstrap resampling, 10,000 iterations. Comparing PLAN vs. REFLECT on MS MARCO seed99 (1000 queries). One-sided *p*-value reported.

Table A.7
Stage-Level Cost Analysis (MS MARCO, seed99, 1000 queries).

Stage	Avg Tokens/Query	Avg Wall-Clock (s)
Step 1: BM25 Retrieval	–	< 0.1
Step 2: LLM Re-ranking	1865	1.0
Step 3: Base Generation	339	10.9
Step 4: Plan Generation	1605	5.2
Step 5: Iterative Reflection	3136	33.0
Total (REFLECT)	**6945**	**50.1**
Total (PLAN only)	**3470**	**17.1**

Per-query averages based on the MS MARCO seed99 batch run (1000 queries). Wall-clock figures include API round-trip time and rate-limit handling.

Data availability

The datasets used in this study are publicly available benchmark datasets (Natural Questions (NQ), TriviaQA, and MS MARCO) obtained from their official releases and widely used public mirrors. No new datasets were created for this study. To support reproducibility, the code, prompts, and experiment configurations, along with processed evaluation outputs, are publicly available at: https://github.com/uncxlirh/ReflectRAG.

References

[1] P. Lewis, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems (NeurIPS), 2020.
[2] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr. 3 (4) (2009) 333–389.
[3] R. Nogueira, K. Cho, Passage re-ranking with BERT, arXiv preprint arXiv:1901.04085, 2019.
[4] V. Karpukhin, et al., Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.
[5] W. Sun, et al., Is ChatGPT good at search? Investigating large language models as re-ranking agents, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023.
[6] Y. Zeng, et al., LLM-RankFusion: mitigating intrinsic inconsistency in LLM-based ranking, arXiv preprint arXiv:2406.00231, 2024.
[7] T. Chakraborty, V. La Gatta, V. Moscato, G. Sperlì, Information retrieval algorithms and neural ranking models to detect previously fact-checked information, Neurocomputing 557 (2023) 126680.
[8] A. Madaan, et al., Self-refine: iterative refinement with self-feedback, arXiv preprint arXiv:2303.17651, 2023.
[9] H. Lightman, et al., Let's verify step by step, in: International Conference on Learning Representations (ICLR), 2024.
[10] Z. Shao, et al., DeepSeekMath: pushing the limits of mathematical reasoning in open language models, arXiv preprint arXiv:2402.03300, 2024.
[11] T. Zhang, V. Kishore, F. Wu, K.Q. Weinberger, Y. Artzi, BERTScore: evaluating text generation with BERT, in: International Conference on Learning Representations (ICLR), 2020.
[12] Y. Gao, et al., Retrieval-augmented generation for large language models: a survey, arXiv preprint arXiv:2312.10997, 2023.
[13] G. Izacard, E. Grave, Distilling knowledge from reader to retriever for question answering, in: International Conference on Learning Representations (ICLR), 2021.
[14] A. Asai, et al., Self-RAG: learning to retrieve, generate, and critique through self-reflection, in: International Conference on Learning Representations (ICLR), 2024.
[15] S.-Q. Yan, J.-C. Gu, Y. Zhu, Z.-H. Ling, Corrective retrieval augmented generation, arXiv preprint arXiv:2401.15884, 2024.
[16] Z. Jiang, et al., Active retrieval augmented generation, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023.
[17] J. Wei, et al., Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems (NeurIPS), 2022.
[18] L. Wang, et al., Plan-and-solve prompting: improving zero-shot chain-of-thought reasoning by large language models, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 2023.
[19] H. Trivedi, et al., Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 2023.
[20] C. Li, M. Liu, J. Zhang, K. He, M. Zhang, S. Ma, Relation consistent chain-of-thought prompting for retrieval-augmented generation, Neurocomputing 637 (2025) 129903.
[21] O. Khattab, A. Singhvi, P. Liang, M. Zaharia, C. Potts, DSPy: compiling declarative language model calls into self-improving pipelines, in: International Conference on Learning Representations (ICLR), 2024.

[22] D. Jiang, X. Ren, B.Y. Lin, LLM-blender: ensembling large language models with pairwise ranking and generative fusion, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL), 2023.
[23] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: language agents with verbal reinforcement learning, in: Advances in Neural Information Processing Systems (NeurIPS), 2023.
[24] S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, J. Weston, Chain-of-verification reduces hallucination in large language models, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL), 2024.
[25] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, W. Chen, CRITIC: large language models can self-correct with tool-interactive critiquing, in: International Conference on Learning Representations (ICLR), 2024.
[26] P. Cheng, J. Dai, J. Liu, J. Liu, P. Jia, Reinforcement learning for few-shot text generation adaptation, Neurocomputing 558 (2023) 126689, https://doi.org/10.1016/j.neucom.2023.126689
[27] L. Ouyang, et al., Training language models to follow instructions with human feedback, in: Advances in Neural Information Processing Systems (NeurIPS), 2022.
[28] R. Rafailov, et al., Direct preference optimization: your language model is secretly a reward model, in: Advances in Neural Information Processing Systems (NeurIPS), 2023.
[29] E.J. Hu, et al., LoRA: low-rank adaptation of large language models, in: International Conference on Learning Representations (ICLR), 2022.
[30] P. Bajaj, et al., MS MARCO: a human generated machine reading comprehension dataset, arXiv preprint arXiv:1611.09268, 2016.
[31] M. Joshi, et al., TriviaQA: a large scale distantly supervised challenge dataset for reading comprehension, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), 2017.
[32] T. Kwiatkowski, et al., Natural questions: a benchmark for question answering research, Trans. Assoc. Comput. Linguist. 7 (2019) 452–466, https://doi.org/10.1162/tacl_a_00276.
[33] S. Es, J. James, L. Espinosa Anke, S. Schockaert, RAGAS: automated evaluation of retrieval augmented generation, Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL): System Demonstrations. arXiv:2309.15217, 2024.
[34] C.-Y. Lin, ROUGE: a package for automatic evaluation of summaries, in: Text Summarization Branches Out: Proceedings of the ACL-04 Workshop, 2004.
[35] DeepSeek-AI, DeepSeek-R1: incentivizing reasoning capability in LLMs via reinforcement learning, arXiv preprint arXiv:2501.12948, 2025.
[36] OpenAI, GPT-4o system card, 2024. https://openai.com/index/gpt-4o-system-card/.
[37] S. Yao, et al., ReAct: synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023.
[38] DeepSeek-AI, DeepSeek-V3 technical report, arXiv preprint arXiv:2412.19437, 2024.

Author biography

Xuanhe Chen is currently a senior student at the Global College, Shanghai Jiao Tong University, China, majoring in Electrical and Computer Engineering (ECE). His research interests focus on natural language processing, specifically in retrieval-augmented generation (RAG) and large language models (LLMs).

Yuchen Li received the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University in 2024. He is currently a senior researcher at Baidu Search Science, Beijing, China, and also serves as an adjunct researcher at Shanghai Jiao Tong University. His research focuses on data-driven intelligent computing for interconnected systems and its applications in AI Search, Agentic AI, and the efficiency of LLMs.

Youyi Bi is currently an Associate Professor at Global College, Shanghai Jiao Tong University, China. He obtained his Ph.D. in Mechanical Engineering from Purdue University in 2017. He received his bachelor's degree and master's degree from Beihang University in 2009 and 2012, respectively. His research interests include data-driven design, AI for robotics and smart manufacturing.

Shuaiqiang Wang is currently a Principal Architect at Baidu Inc., Beijing, China. Previously, he was a Research Scientist and Senior Algorithm Engineer at JD Inc. Before that, he worked as an Assistant Professor at the University of Manchester in the UK in 2017 and at the University of Jyväskylä in Finland from 2014 to 2017, respectively. Earlier, he served as an Associate Professor at Shandong University of Finance and Economics in China from 2011 to 2014 and as a Postdoctoral Researcher at Texas State University in the USA from 2010 to 2011. He received his Ph.D. and B.Sc. in Computer Science from Shandong University in 2009 and 2004. He is interested in several research areas, including information retrieval, recommender systems, and data mining. He has published over 100 papers in leading journals and conferences.

Linghe Kong is currently a professor with the Department of Computer Science and Engineering at Shanghai Jiao Tong University, Shanghai, China. Before that, he was a postdoctoral researcher at Columbia University, McGill University, and the Singapore University of Technology and Design. He received his Ph.D. degree from Shanghai Jiao Tong University in 2013, his Master's degree from TELECOM SudParis in 2007, and a B. Eng. degree from Xidian University in 2005. His research interests include wireless networks, big data, mobile computing, and the Internet of Things.

Dawei Yin received the BS degree from Shandong University in 2006, and the MS and PhD degrees from Lehigh University in 2010 and 2013. He is a Senior Director of Engineering at Baidu Inc., Beijing, China. He manages the search science team at Baidu, leading Baidu's scientific efforts in web search, question answering, video search, image search, news search, app search, and more. Previously, he was a senior director, managing the recommendation engineering team at JD.com between 2016 and 2020. Prior to JD.com, he was a senior research manager at Yahoo Labs, leading the relevance science team and being in charge of the Core Search Relevance of Yahoo Search. From 2007 to 2008, he was an MPhil student at the University of Hong Kong. His research interests include data mining, applied machine learning, information retrieval, and recommender systems. He has published research papers in prestigious conferences and journals and was the recipient of eight Best Paper Awards (or runner-ups), including the KDD, WSDM, and ICDM Best Paper Awards.