MAD-RAG:A Multi-Agent Framework for Dynamic Iterative Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Multi Agent Retrieval Augmented Generation shows promise for complex knowledge intensive tasks, but existing agentic systems often rely on prompt based coordination and lack explicit alignment among retrieval control, evidence selection, and answer generation. Differentiable Data Rewards provide utility signals for retrieved information, yet current DDR based RAG methods mainly target single step retrieval and cannot adapt to evolving information needs in multi hop reasoning. We propose MAD-RAG, a dynamic multi agent framework that extends DDR style utility estimation to iterative retrieval augmented generation. MAD-RAG coordinates a Reflective Retrieval Agent for knowledge gap identification and retrieval control, a Utility Guided Denoising Agent for utility based evidence selection, and a Conflict Resolving Generation Agent for integrating selected evidence with previous reasoning states. The agents are optimized in a stagewise manner with DDR enhanced Direct Preference Optimization, which converts utility signals into agent specific preference pairs. Experiments on six knowledge intensive benchmarks show that MAD-RAG outperforms strong RAG and agentic baselines on five datasets, while reducing retrieval frequency by up to 96.7% and end to end latency by up to 97.9%.

1 Introduction

Large language models (LLMs) have achieved strong performance across NLP tasks, but still suffer from hallucinations caused by incomplete, outdated, or incorrectly recalled parametric knowledge (Ji et al., 2023; Maynez et al., 2020; Petroni et al., 2019). Retrieval-Augmented Generation (RAG) mitigates this issue by grounding generation in external evidence (Lewis et al., 2020; Guu et al., 2020; Karpukhin et al., 2020; Izacard and Grave, 2021). Recent surveys further show that RAG has evolved from naive retrieve-then-generate pipelines toward advanced and modular retrieval-augmented systems (Gao et al., 2023). Beyond the original paradigm, recent studies improve retrieval-generation interaction through black-box retrieval augmentation, instruction tuning, and adaptive retrieval (Shi et al., 2024; Lin et al., 2024; Asai et al., 2024; Jiang et al., 2023). However, static or weakly adaptive RAG systems may still retrieve for every query, introducing redundant computation and noisy evidence (Asai et al., 2024; Jiang et al., 2023).

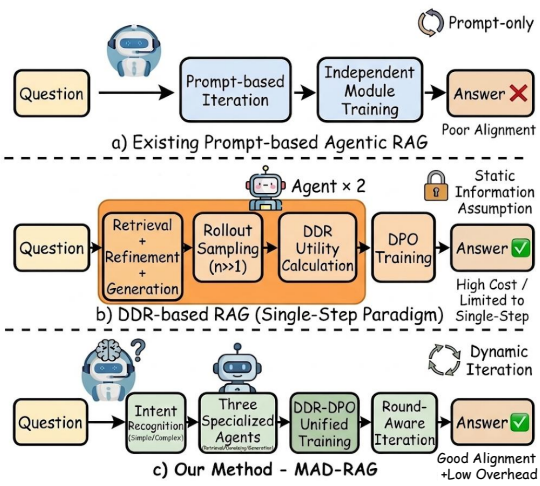

Figure 1: Comparison of three RAG paradigms. Agentic RAG is efficient but weakly aligned; DDR-based RAG provides utility-driven alignment but is mostly single-step; MAD-RAG combines utility-driven training with dynamic multi-round retrieval.

Recent iterative and agentic RAG methods improve flexibility by enabling multi-step retrieval, search-augmented reasoning, tool use, and role-based collaboration (Trivedi et al., 2023; Jiang et al., 2023; Yao et al., 2022; Schick et al., 2023; Asai et al., 2024; Jin et al., 2025; Nguyen et al., 2025; Du et al., 2026; Singh et al., 2025). More broadly, LLM-based agent frameworks show that

multiple specialized agents can coordinate through conversation, workflows, or role assignment (Wu et al., 2024; Hong et al., 2024). Nevertheless, many of these systems still rely heavily on prompt-level coordination or heuristic retrieval control, making it difficult to align retrieval, evidence filtering, and generation toward final answer quality. Differentiable Data Rewards (DDR) provide a principled way to estimate the downstream utility of retrieved information and construct preference signals for RAG optimization (Li et al., 2025), but existing DDR-based RAG methods are mainly designed for single-step retrieval and do not explicitly handle dynamically evolving information needs in multi-hop reasoning.

To address these limitations, we propose MAD-RAG, a multi-agent framework that extends DDR-enhanced preference optimization to dynamic iterative RAG. MAD-RAG contains three specialized agents: a Reflective Retrieval Agent that identifies remaining knowledge gaps and decides whether to retrieve, a Utility-Guided Denoising Agent that filters retrieved evidence according to downstream utility, and a Conflict-Resolving Generation Agent that integrates selected evidence with previous reasoning states. Instead of jointly optimizing all agents end-to-end, MAD-RAG constructs agent-specific preference pairs using DDR-derived utility signals and trains the agents in a stage-wise manner with Direct Preference Optimization (DPO) (Rafailov et al., 2023). This design preserves a unified utility-driven training principle while reducing the instability of simultaneous multi-agent optimization.

Our contributions are summarized as follows:

- We propose MAD-RAG, a dynamic multi-agent RAG framework that jointly models retrieval control, evidence denoising, and answer generation in an iterative decision process.
- We extend DDR-style utility estimation from single-step RAG to multi-round agentic RAG, using single-step utilities for denoising and generation and trajectory-level utilities for retrieval control.
- We develop a stage-wise DDR-DPO training framework that aligns the three agents with final answer quality while avoiding simultaneous multi-agent optimization.
- Experiments on six knowledge-intensive benchmarks show that MAD-RAG outperforms strong RAG and agentic baselines, while substantially reducing retrieval frequency and end-to-end latency.

2 Related Work

2.1 Retrieval-Augmented Generation and Iterative RAG

Retrieval-augmented generation (RAG) mitigates hallucinations in LLMs by grounding generation on external knowledge (Lewis et al., 2020; Guu et al., 2020). Early retrieval-augmented systems improve open-domain question answering and language modeling through dense retrieval, passage fusion, and large-scale external memory (Karpukhin et al., 2020; Izacard and Grave, 2021; Borgeaud et al., 2022). Atlas further shows that retrieval-augmented language models can achieve strong few-shot performance on knowledge-intensive tasks with fewer parameters (Izacard et al., 2023). Recent surveys summarize the evolution of RAG from standard retrieve-then-generate pipelines to more modular and adaptive systems (Gao et al., 2023).

Vanilla RAG follows a retrieve-then-generate paradigm, but its one-shot retrieval strategy is often insufficient for complex knowledge-intensive tasks (Lewis et al., 2020). To improve adaptivity, later methods introduce retrieval-aware generation or iterative reasoning. Self-RAG (Asai et al., 2024) learns to retrieve and critique its own generations through self-reflection, FLARE (Jiang et al., 2023) actively retrieves documents according to generation uncertainty, IRCoT (Trivedi et al., 2023) interleaves retrieval with chain-of-thought reasoning, and Iter-RetGen (Shao et al., 2023) iteratively improves retrieval and generation by using model outputs as retrieval contexts. Adaptive-RAG (Jeong et al., 2024) dynamically selects retrieval strategies according to question complexity. REPLUG (Shi et al., 2024) reweights generation with retrieved documents, RA-DIT (Lin et al., 2024) improves RAG through dual instruction tuning, and Search-R1 (Jin et al., 2025) integrates search and generation in an iterative reinforcement learning framework. Despite their effectiveness, these methods are mainly static or single-agent systems, making it difficult to coordinate specialized retrieval, filtering, and generation behaviors in a unified manner.

2.2 Multi-Agent RAG

Agentic RAG extends standard RAG with tool use, task decomposition, and role-specialized workflow orchestration (Yao et al., 2022; Schick et al., 2023; Singh et al., 2025). More broadly, recent LLM agent frameworks demonstrate that multiple agents can coordinate through role-playing, conversation, workflow decomposition, or explicit role assignment (Li et al., 2023; Wu et al., 2024; Hong et al., 2024). For example, MA-RAG (Nguyen et al., 2025) improves RAG by assigning different roles to collaborative agents, while A-RAG (Du et al., 2026) explores scalable agentic retrieval through hierarchical retrieval interfaces. Recent agentic RAG studies and surveys further discuss multi-agent, graph-based, and tool-augmented retrieval workflows for complex reasoning (Du et al., 2026; Singh et al., 2025). These methods show the potential of agent collaboration for complex knowledge-intensive tasks. However, most existing multi-agent RAG systems rely primarily on prompt-based coordination or independently designed modules, and lack a unified training signal to align retrieval control, evidence selection, and answer generation toward final answer quality. In contrast, MAD-RAG uses utility-derived preferences to align multiple agents under a shared optimization principle.

2.3 Data-Driven RAG Alignment

Recent work explores data-driven optimization to better align RAG modules. RA-DIT (Lin et al., 2024) uses instruction tuning to improve retriever-generator coordination, while RAG-DDR (Li et al., 2025) introduces Differentiable Data Rewards (DDR) to estimate downstream utility and construct preference signals for RAG optimization. Preference-based optimization methods such as DPO (Rafailov et al., 2023) provide a stable way to learn from preference pairs without explicitly training a separate reward model. DDR provides a principled way to evaluate whether retrieved information improves final answer quality. However, existing DDR-based RAG is mainly designed for single-step retrieval and does not explicitly model evolving information needs across multiple reasoning rounds. MAD-RAG extends DDR-style utility estimation to iterative multi-agent RAG by combining single-step utilities for evidence denoising and generation with trajectory-level utilities for retrieval control.

3 Methodology

3.1 Problem Formulation

Given a query q and an external corpus $\mathcal{C}$, retrieval-augmented generation aims to produce a final answer Y^* grounded in retrieved evidence (Lewis et al., 2020; Guu et al., 2020). MAD-RAG formulates this process as dynamic multi-round decision making, inspired by iterative and adaptive retrieval paradigms (Trivedi et al., 2023; Jiang et al., 2023; Asai et al., 2024). At each round, the system identifies unresolved information needs, retrieves when necessary, filters noisy documents, and updates the answer state.

Let $T_{\max}$ denote the maximum number of refinement rounds. At round t, MAD-RAG maintains

$$H_t = (y_t, \mathcal{E}_t, c_t), \tag{1}$$

where y_t is the answer, $\mathcal{E}_t$ is the rationale, and c_t is the confidence score, with $H_0 = (\varnothing, \varnothing, 0)$. Instead of using a scalar retrieval score, MAD-RAG represents the remaining information need as a natural-language knowledge gap $\mathcal{G}_t$, which may describe a missing fact, entity relation, temporal condition, or reasoning bridge.

MAD-RAG consists of three agents. The retrieval agent $\mathcal{A}_{ret}$ identifies $\mathcal{G}_t$, decides whether to retrieve, and constructs a retrieval query r_t; the denoising agent $\mathcal{A}_{den}$ selects useful evidence; and the generation agent $\mathcal{A}_{gen}$ updates the answer by integrating parametric knowledge, selected evidence, and previous states. We use $\mathcal{A}_{ret}$ rather than $\mathcal{A}_{ref}$ to avoid confusion with the DPO reference policy.

3.2 MAD-RAG Inference

The first-round retrieval state is $S_1^{ret} = q$. For $t > 1$, the retrieval state is

$$S_t^{ret} = (q, H_{t-1}, \mathcal{D}_{1:t-1}, \mathcal{G}_{t-1}), \tag{2}$$

where $\mathcal{D}_{1:t-1} = \{\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}\}$ denotes previously selected evidence. At each round, the retrieval agent outputs

$$(\alpha_t, \mathcal{G}_t, r_t) = \mathcal{A}_{ret}(S_t^{ret}), \tag{3}$$

where $\alpha_t \in \{\text{Yes}, \text{No}\}$ indicates whether retrieval is needed.

If $\alpha_1 = \text{No}$, MAD-RAG directly invokes the generation agent without external evidence:

$$(Y^*, \mathcal{E}^*, c^*) = \mathcal{A}_{gen}(q, \varnothing, \varnothing, H_0). \tag{4}$$

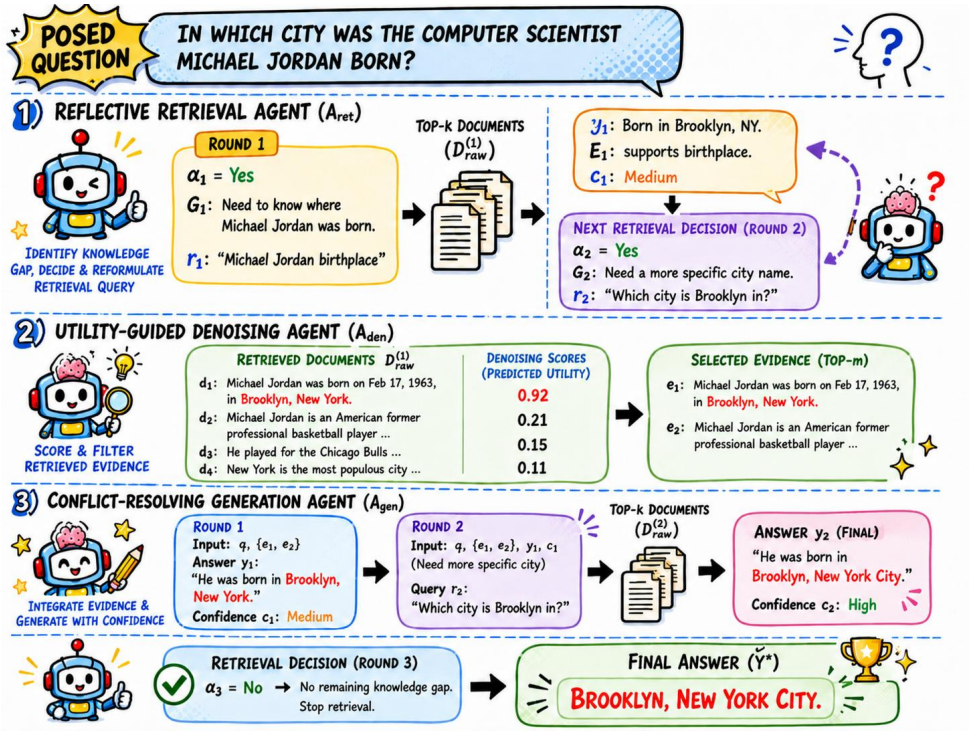

Figure 2: Inference pipeline of MAD-RAG. The retrieval agent $\mathcal{A}_{ret}$ identifies knowledge gaps and controls retrieval across rounds. The denoising agent $\mathcal{A}_{den}$ selects utility-relevant evidence, while the generation agent $\mathcal{A}_{gen}$ updates the answer state using selected evidence and previous reasoning states.

Otherwise, it enters the retrieval-refinement loop. At round t, MAD-RAG retrieves top-k documents and applies denoising:

$$\begin{aligned} \mathcal{D}_{raw}^{(t)} &= \text{Retrieve}(r_t, \mathcal{C}, k), \\ S_t^{den} &= (q, r_t, \mathcal{G}_t, \mathcal{D}_{raw}^{(t)}, H_{t-1}), \\ \mathcal{D}_t &= \mathcal{A}_{den}(S_t^{den}). \end{aligned} \tag{5}$$

The generation agent then updates the answer state:

$$\begin{aligned} S_t^{gen} &= (q, \mathcal{D}_t, \mathcal{G}_t, H_{t-1}), \\ H_t &= (y_t, \mathcal{E}_t, c_t) = \mathcal{A}_{gen}(S_t^{gen}). \end{aligned} \tag{6}$$

After generation, $\mathcal{A}_{ret}$ re-examines the updated state. If $\alpha_{t+1} = \text{No}$, the system returns y_t; otherwise, it continues with r_{t+1}.

3.3 Reflective Retrieval Agent

The Reflective Retrieval Agent $\mathcal{A}_{ret}$ serves as the global controller. Given S_t^{ret}, it predicts $(\alpha_t, \mathcal{G}_t, r_t)$. If the current answer is sufficiently supported, it sets $\alpha_t = \text{No}$; otherwise, it updates $\mathcal{G}_t$ and constructs r_t from the query and current reasoning state.

To train $\mathcal{A}_{ret}$, we estimate the long-term utility of alternative retrieval actions using DDR-style utility estimation (Li et al., 2025). For each retrieval state S_t^{ret}, we sample n_{ret} candidate retrieval actions:

$$a_{t,j}^{ret} = (\alpha_{t,j}, \mathcal{G}_{t,j}, r_{t,j}), \quad j = 1, \ldots, n_{ret}. \tag{7}$$

Each action is evaluated by rolling out the remaining process with fixed downstream agents:

$$Y_{t,j}^* = \text{Rollout}(S_t^{ret}, a_{t,j}^{ret}; \mathcal{A}_{den}, \mathcal{A}_{gen}), \tag{8}$$

and its utility is

$$U_{ret}(a_{t,j}^{ret}) = R(q, Y_{t,j}^*), \tag{9}$$

where $R(q, y)$ is a task-specific answer-quality reward. We select

$$\begin{aligned} j_{ret}^{+} &= \arg\max_j U_{ret}(a_{t,j}^{ret}), \\ j_{ret}^{-} &= \arg\min_j U_{ret}(a_{t,j}^{ret}). \end{aligned} \tag{10}$$

and construct the preference pair

$$a_{ret}^{+} = a_{t,j_{ret}^{+}}^{ret}, \quad a_{ret}^{-} = a_{t,j_{ret}^{-}}^{ret}. \tag{11}$$

This rollout utility is used only for offline preference construction.

Algorithm 1: MAD-RAG Inference Procedure

Input: Query q, corpus $\mathcal{C}$, maximum rounds $T_{\max}$
Output: Final answer Y^*

$H_0 \leftarrow (\varnothing, \varnothing, 0)$;
$S_1^{ret} \leftarrow q$;
$(\alpha_1, \mathcal{G}_1, r_1) \leftarrow \mathcal{A}_{ret}(S_1^{ret})$;
if $\alpha_1 = \textit{No}$ **then**
  $(Y^*, \mathcal{E}^*, c^*) \leftarrow \mathcal{A}_{gen}(q, \varnothing, \varnothing, H_0)$;
  return Y^*;
for $t \leftarrow 1$ **to** $T_{\max}$ **do**
  $\mathcal{D}_{raw}^{(t)} \leftarrow \text{Retrieve}(r_t, \mathcal{C}, k)$;
  $S_t^{den} \leftarrow (q, r_t, \mathcal{G}_t, \mathcal{D}_{raw}^{(t)}, H_{t-1})$;
  $\mathcal{D}_t \leftarrow \mathcal{A}_{den}(S_t^{den})$;
  $S_t^{gen} \leftarrow (q, \mathcal{D}_t, \mathcal{G}_t, H_{t-1})$;
  $H_t = (y_t, \mathcal{E}_t, c_t) \leftarrow \mathcal{A}_{gen}(S_t^{gen})$;
  $S_{t+1}^{ret} \leftarrow (q, H_t, \mathcal{D}_{1:t}, \mathcal{G}_t)$;
  $(\alpha_{t+1}, \mathcal{G}_{t+1}, r_{t+1}) \leftarrow \mathcal{A}_{ret}(S_{t+1}^{ret})$;
  if $\alpha_{t+1} = \textit{No}$ **then**
    return y_t;
return $y_{T_{\max}}$;

3.4 Utility-Guided Denoising Agent

The Utility-Guided Denoising Agent $\mathcal{A}_{den}$ filters noisy, redundant, or weakly useful evidence. At round t, the raw retrieved documents are

$$\mathcal{D}_{raw}^{(t)} = \{d_i^{(t)}\}_{i=1}^{k}. \tag{12}$$

For each document, the denoising state and keep-or-discard action are

$$\begin{aligned} S_{t,i}^{den} &= (q, r_t, \mathcal{G}_t, d_i^{(t)}, H_{t-1}), \\ z_i^{(t)} &= \mathcal{A}_{den}(S_{t,i}^{den}), \quad z_i^{(t)} \in \{\text{Keep}, \text{Discard}\}. \end{aligned} \tag{13}$$

The filtered evidence set is

$$\mathcal{D}_t = \{d_i^{(t)} \mid z_i^{(t)} = \text{Keep}\}, \tag{14}$$

with at most m documents retained according to denoising scores.

For preference construction, we estimate document utility by its downstream contribution to answer quality (Li et al., 2025). The fixed generation agent produces

$$\tilde{H}_{t,i} = \mathcal{A}_{gen}(q, \{d_i^{(t)}\}, \mathcal{G}_t, H_{t-1}), \tag{15}$$

where $\tilde{H}_{t,i} = (\tilde{y}_{t,i}, \tilde{\mathcal{E}}_{t,i}, \tilde{c}_{t,i})$. The document utility is

$$U_{den}(d_i^{(t)}) = R(q, \tilde{y}_{t,i}). \tag{16}$$

We identify

$$\begin{aligned} i^+ &= \arg\max_i U_{den}(d_i^{(t)}), \\ i^- &= \arg\min_i U_{den}(d_i^{(t)}). \end{aligned} \tag{17}$$

For $d_{i^+}^{(t)}$, Keep is preferred over Discard; for $d_{i^-}^{(t)}$, Discard is preferred over Keep. During inference, $\mathcal{A}_{den}$ predicts actions directly.

3.5 Conflict-Resolving Generation Agent

The Conflict-Resolving Generation Agent $\mathcal{A}_{gen}$ updates the answer using the query, selected evidence, current knowledge gap, and previous state. It incorporates useful evidence while avoiding unsupported updates under noisy or conflicting evidence. At round t,

$$H_t = \mathcal{A}_{gen}(S_t^{gen}), \quad H_t = (y_t, \mathcal{E}_t, c_t). \tag{18}$$

When $\mathcal{D}_t = \varnothing$, the agent relies on parametric knowledge and the previous state.

To train $\mathcal{A}_{gen}$, we use dual-source candidate generation. For the same query and state, we sample candidates from an evidence-aware context

$$(q, \mathcal{D}_t, \mathcal{G}_t, H_{t-1}), \tag{19}$$

and an evidence-free context

$$(q, \varnothing, \mathcal{G}_t, H_{t-1}). \tag{20}$$

This encourages evidence use when helpful and robustness to noisy or insufficient evidence.

Let $\mathcal{H}_t = \{H_{t,j}\}_{j=1}^{n_g}$ denote sampled candidate answer states, where $H_{t,j} = (y_{t,j}, \mathcal{E}_{t,j}, c_{t,j})$. Each candidate is scored by

$$U_{gen}(H_{t,j}) = R(q, y_{t,j}). \tag{21}$$

We select

$$\begin{aligned} j_{gen}^+ &= \arg\max_j U_{gen}(H_{t,j}), \\ j_{gen}^- &= \arg\min_j U_{gen}(H_{t,j}). \end{aligned} \tag{22}$$

and construct

$$H_{gen}^+ = H_{t,j_{gen}^+}, \quad H_{gen}^- = H_{t,j_{gen}^-}. \tag{23}$$

3.6 DDR-Enhanced Preference Optimization

MAD-RAG uses Differentiable Data Rewards (DDR) (Li et al., 2025) as a utility estimation mechanism rather than end-to-end differentiable back-propagation. The reward function

$$R(q, y) \in \mathbb{R} \tag{24}$$

measures answer quality and is instantiated by task-specific metrics such as Accuracy, ROUGE-L, or F1. For $\mathcal{A}_{den}$ and $\mathcal{A}_{gen}$, utilities are estimated from current-round answer quality; for $\mathcal{A}_{ret}$, utilities are estimated through full-trajectory rollout.

For each agent type $z \in \{ret, den, gen\}$, we construct preference triples

$$(x^z, a^{z,+}, a^{z,-}), \tag{25}$$

where x^z is the agent input, and $a^{z,+}$ and $a^{z,-}$ are preferred and dispreferred actions. The agent-specific inputs and actions are

$$\begin{aligned} x^{ret} &= S_t^{ret}, \quad a^{ret} = (\alpha_t, \mathcal{G}_t, r_t), \\ x_i^{den} &= S_{t,i}^{den}, \quad a_i^{den} = z_i^{(t)}, \\ x^{gen} &= S_t^{gen}, \quad a^{gen} = H_t. \end{aligned} \tag{26}$$

Let π_θ^z be the trainable policy and π_0^z the frozen reference policy. We define

$$\rho_\theta^z(a, x) = \log \frac{\pi_\theta^z(a|x)}{\pi_0^z(a|x)}. \tag{27}$$

The preference margin is

$$\Delta_\theta^z = \rho_\theta^z(a^{z,+}, x^z) - \rho_\theta^z(a^{z,-}, x^z). \tag{28}$$

Following Direct Preference Optimization (DPO) (Rafailov et al., 2023), the loss for agent z is

$$\mathcal{L}_{DPO}^z = -\mathbb{E}\left[\log \sigma\left(\beta \Delta_\theta^z\right)\right], \tag{29}$$

where β controls the deviation from the reference policy.

All agents share this objective but are trained stage-wise. We first train $\mathcal{A}_{gen}$ with response-level preferences, then train $\mathcal{A}_{den}$ with $\mathcal{A}_{gen}$ fixed, and finally train $\mathcal{A}_{ret}$ with both downstream agents fixed. During inference, no utility rollout is performed; each agent directly predicts its action.

4 Experiment

4.1 Experimental Setup

4.1.1 Datasets

We evaluate MAD-RAG on six knowledge-intensive benchmarks covering open-domain question answering, multi-hop reasoning, fact lookup, and dialogue grounding. We use Natural Questions (NQ) (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and MARCO QA (Bajaj et al., 2016) for open-domain QA, HotpotQA (Yang et al., 2018) for multi-hop reasoning, T-REx (Elsahar et al., 2018) for one-hop fact lookup, and Wizard of Wikipedia (WoW) (Dinan et al., 2018) for dialogue grounding. These datasets jointly evaluate evidence retrieval, factual reasoning, and grounded response generation.

4.1.2 Evaluation Metrics

Following common evaluation protocols, we use Accuracy for NQ, TriviaQA, HotpotQA, and T-REx, ROUGE-L for MARCO QA, and token-level F1 for WoW. For Accuracy, a prediction is correct if its normalized answer matches one of the normalized references, where normalization includes lowercasing, punctuation removal, article removal, and whitespace normalization. For multiple references, we report the maximum score. During preference construction, these task-specific metrics are used as answer-quality rewards $R(q, y)$ to rank candidate outputs or actions without training an additional reward model.

4.1.3 Baselines

We compare MAD-RAG with representative non-retrieval, retrieval-augmented, adaptive retrieval, and multi-agent RAG baselines, including LLM w/o RAG, Vanilla RAG (Lewis et al., 2020), REPLUG (Shi et al., 2024), Self-RAG (Asai et al., 2024), RA-DIT (Lin et al., 2024), RAG-DDR (Li et al., 2025), Search-R1 (Jin et al., 2025), MA-RAG (Nguyen et al., 2025), and A-RAG (Du et al., 2026). These baselines cover retrieve-then-generate pipelines, instruction-tuned RAG, DDR-style optimization, reinforcement-learning-based retrieval, adaptive retrieval, and multi-agent RAG.

For fair comparison, Search-R1, MA-RAG, and A-RAG are re-run under the same model and parameter configuration as MAD-RAG. The remaining baseline results are taken from RAG-DDR (Li et al., 2025), since they follow the same evaluation protocol and report results on the same benchmark suite. The source of each baseline result and the re-run protocol are provided in Appendix A.12.

4.1.4 Implementation

We implement all three agents with Llama-3-8B-Instruct (Grattafiori et al., 2024). The Reflective Retrieval Agent $\mathcal{A}_{ret}$, Utility-Guided Denoising Agent $\mathcal{A}_{den}$, and Conflict-Resolving Generation Agent $\mathcal{A}_{gen}$ are initialized from the same backbone and optimized separately under stage-wise DDR-DPO (Li et al., 2025; Rafailov et al., 2023). We use BGE-large-en-v1.5 (Xiao et al., 2024) as the dense

Table 1: Overall performance of different RAG models across six knowledge-intensive benchmarks. The best results are shown in bold, and the second-best results are underlined.

Method	Open-Domain QA			Multi-Hop QA	Slot Filling	Dialogue
	NQ	TriviaQA	MARCO QA	HotpotQA	T-REx	WoW
LLM w/o RAG	35.4	78.4	17.0	27.7	16.0	14.6
Vanilla RAG	46.2	84.0	20.6	30.1	26.9	12.5
REPLUG	45.1	83.1	22.5	29.4	23.1	14.0
Self-RAG	39.6	78.2	12.5	24.3	29.5	14.2
RA-DIT	46.2	87.4	20.3	34.9	41.7	14.8
RAG-DDR	50.7	88.2	25.1	37.3	37.3	14.9
Search-R1	55.2	88.9	**26.7**	39.7	42.1	14.4
MA-RAG	54.6	87.8	23.7	38.2	40.2	14.0
A-RAG	56.7	89.0	24.9	40.1	41.9	14.4
MAD-RAG (Ours)	**60.1**	**89.8**	25.4	**41.1**	**44.4**	**15.1**

Table 2: Ablation results of MAD-RAG. "TQA", "MQA", and "HQA" denote TriviaQA, MARCO QA, and HotpotQA, respectively. "Ret", "Den", and "Gen" denote the retrieval, denoising, and generation agents, respectively.

Variant	NQ	TQA	MQA	HQA	T-REx	WoW
Agent Flow Ablation						
w/o Ret	45.4	83.2	18.2	28.5	26.6	12.3
w/o Den	50.1	85.5	21.6	33.7	28.9	13.1
w/o Ret+Den	42.1	79.8	16.7	26.1	24.4	10.1
Training Strategy Ablation						
w/o train Den	58.8	88.8	24.1	40.2	44.0	14.2
w/o train Ret	57.1	87.2	23.7	35.6	38.2	13.5
w/o train Gen	55.9	86.3	20.3	32.8	36.5	12.1
Full	**60.1**	**89.8**	**25.4**	**41.1**	**44.4**	**15.1**

retriever and retrieve documents from MS MARCO 2.1 (Bajaj et al., 2016). For each retrieval call, the retriever returns top-$k = 10$ passages, and $\mathcal{A}_{den}$ retains at most $m = 5$ passages. We set $T_{\max} = 3$ for main inference and train the agents stage-wise in the order of $\mathcal{A}_{gen}$, $\mathcal{A}_{den}$, and $\mathcal{A}_{ret}$. The maximum generation length is 64 tokens for NQ, TriviaQA, HotpotQA, T-REx, and WoW, and 128 tokens for MARCO QA. Detailed hyperparameters are provided in Appendix A.

4.2 Overall Performance

Table 1 summarizes the main results across six benchmarks. Although several baselines support adaptive retrieval, iterative reasoning, or agent-based collaboration, they do not explicitly combine reflective retrieval control, utility-guided evidence denoising, and conflict-aware generation under unified stage-wise preference optimization.

MAD-RAG achieves the best performance on five out of six datasets. Compared with the strongest baseline on each dataset, MAD-RAG improves by 3.4 points on NQ, 0.8 on TriviaQA, 1.0 on HotpotQA, 2.3 on T-REx, and 0.2 on WoW. On MARCO QA, MAD-RAG ranks second and remains competitive with Search-R1 while outperforming other RAG and multi-agent baselines. These results show a strong trade-off between answer quality and adaptive evidence use.

4.3 Ablation Study

Table 2 evaluates MAD-RAG from two perspectives. *Agent Flow Ablation* removes agents from the inference pipeline, while *Training Strategy Ablation* keeps the full pipeline but disables preference optimization for one agent.

Agent Flow Ablation. Removing $\mathcal{A}_{ret}$ causes an average drop of 10.3 points, showing the importance of reflective retrieval control. Removing $\mathcal{A}_{den}$ leads to a 7.2-point drop, indicating that filtering noisy or redundant evidence improves generation quality. Removing both agents results in the largest drop of 12.8 points, suggesting that adaptive retrieval control and evidence denoising are complementary.

Training Strategy Ablation. Disabling $\mathcal{A}_{gen}$ training causes the largest average drop of 5.3 points, especially on HotpotQA and T-REx, showing that generation optimization is crucial for reasoning over evidence and resolving conflicts. Disabling $\mathcal{A}_{ret}$ training leads to a 3.4-point drop, while disabling $\mathcal{A}_{den}$ training causes a smaller but consistent 1.0-point drop. Overall, MAD-RAG benefits from both modular agent design and stage-wise preference optimization.

Table 3: Efficiency-quality trade-off of reflective retrieval in the two-round setting with $T_{\max} = 2$. "Triggered" denotes the number of queries for which $\mathcal{A}_{ret}$ performs the second-round retrieval. "Full Lat." and "Partial Lat." denote total latency over the evaluation subset. Δ denotes the performance difference between Partial Retrieval and Full Retrieval.

Dataset	Triggered	Ret. Ratio	Partial	Full	Δ	Full Lat.	Partial Lat.	Lat. Red.
NQ	338/2837	11.9%	60.1	60.3	-0.2	13,120.7s	2,172.7s	83.4%
TriviaQA	179/5359	3.3%	89.8	89.9	-0.1	58,472.1s	8,705.7s	85.1%
MARCO QA	919/3000	30.6%	25.4	24.4	+1.0	15,665.0s	320.0s	98.0%
HotpotQA	824/5600	14.7%	41.1	41.5	-0.4	37,590.0s	1,668.0s	95.6%
T-REx	3249/5000	65.0%	44.4	45.7	-1.3	24,601.0s	9,487.6s	61.4%
WoW	2318/3054	75.9%	15.1	14.7	+0.4	5,001.0s	674.1s	86.5%

4.4 Effect of DDR-Enhanced Preference Optimization

To isolate the effect of DDR-DPO from the multi-agent architecture, we compare MAD-RAG with a supervised fine-tuning variant under the same pipeline. MAD-RAG + SFT imitates supervised targets such as retrieval decisions, evidence labels, or reference answers, while MAD-RAG + DDR-DPO constructs preference pairs from downstream utility signals.

Table 4 shows that DDR-DPO consistently outperforms SFT on all three datasets. Compared with MAD-RAG + SFT, DDR-DPO improves by 12.5 points on NQ, 6.0 on HotpotQA, and 2.7 on MARCO QA, indicating that the gains come not only from additional training or architecture design, but also from downstream utility based preference supervision.

Table 4: Effect of training objectives under the same MAD-RAG architecture.

Method	NQ	HotpotQA	MARCO QA
MAD-RAG + SFT	47.6	35.1	22.7
MAD-RAG + DDR-DPO	**60.1**	**41.1**	**25.4**

4.5 Efficiency Analysis of Reflective Retrieval

We evaluate whether $\mathcal{A}_{ret}$ reduces retrieval cost while preserving answer quality. Although the main experiments use $T_{\max} = 3$, we set $T_{\max} = 2$ here to isolate the second-round retrieval decision, comparing Full Retrieval, which always retrieves again, with Partial Retrieval, where $\mathcal{A}_{ret}$ decides whether retrieval is needed. The latency measurement protocol is detailed in Appendix A.13.

Table 3 shows that $\mathcal{A}_{ret}$ greatly reduces second-round retrieval. Only 3.3% of TriviaQA queries and 11.9% of NQ queries trigger another retrieval step, reducing MARCO QA latency from 15,665.0s to 320.0s and HotpotQA latency from 37,590.0s to 1,668.0s.

Partial Retrieval remains close to Full Retrieval in most cases, with differences within 0.4 points on NQ, TriviaQA, and HotpotQA. It even improves MARCO QA and WoW, likely because avoiding unnecessary retrieval reduces exposure to redundant or weakly relevant evidence. The largest drop appears on T-REx, where the 65.0% retrieval ratio suggests that fact lookup tasks require more frequent external evidence. Overall, $\mathcal{A}_{ret}$ achieves a favorable efficiency-quality trade-off.

5 Conclusion

We propose MAD-RAG, a dynamic multi-agent framework for iterative retrieval-augmented generation. MAD-RAG coordinates a Reflective Retrieval Agent, a Utility-Guided Denoising Agent, and a Conflict-Resolving Generation Agent to identify knowledge gaps, select useful evidence, and refine answers across retrieval rounds. By constructing agent-specific preference pairs with DDR-derived utility signals and optimizing the agents through stage-wise DPO, MAD-RAG aligns retrieval control, evidence selection, and generation with final answer quality. Experiments on six knowledge-intensive benchmarks show that MAD-RAG outperforms strong RAG and agentic baselines while reducing unnecessary retrieval and end-to-end latency. These results demonstrate the effectiveness of combining adaptive retrieval control with utility-driven multi-agent alignment for efficient and reliable RAG.

6 Limitations

Although MAD-RAG improves the effectiveness-efficiency trade-off of iterative RAG, it has several

limitations. First, its preference construction requires additional offline computation, especially for trajectory-level utility estimation of retrieval actions. Second, the method depends on task-specific automatic rewards to build preference pairs, which may not fully reflect factual faithfulness or human preference in open-ended settings. Third, MAD-RAG is still limited by retriever and corpus quality; when relevant evidence is absent from the top-k results, downstream agents may fail to recover the correct answer. Finally, our experiments are mainly conducted on English knowledge-intensive benchmarks, leaving multilingual, domain-specific, and long-form generation scenarios for future work.

7 Ethical Considerations

This work uses publicly available datasets and retrieval corpora, and does not involve collecting new personal data or human-subject annotations. However, MAD-RAG may inherit biases, factual errors, or harmful content from the underlying LLM, retriever, and external corpus. Although the retrieval and denoising agents reduce unnecessary or noisy evidence, they cannot guarantee that all generated answers are correct or unbiased. Users should therefore avoid deploying MAD-RAG directly in high-stakes domains such as medical, legal, financial, or safety-critical decision-making without human oversight. The offline DDR-DPO preference construction also introduces additional computational cost, which may increase energy consumption. We will provide implementation details to support reproducibility and responsible use.

References

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

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, and 1 others. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. 2026. A-rag: Scaling agentic retrieval-augmented generation via hierarchical retrieval interfaces. *arXiv preprint arXiv:2602.03442*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, Haofen Wang, and 1 others. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 7969–7992.

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. *arXiv preprint arXiv:2503.09516*.

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 Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, and 1 others. 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, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008.

Xinze Li, Sen Mei, Zhenghao Liu, Yukun Yan, Shuo Wang, Shi Yu, Zheni Zeng, Hao Chen, Ge Yu, Zhiyuan Liu, and 1 others. 2025. Rag-ddr: Optimizing retrieval-augmented generation using differentiable data rewards. In *International Conference on Learning Representations*, volume 2025, pages 9606–9627.

Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, and 1 others. 2024. Ra-dit: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*, volume 2024, pages 19138–19162.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 1906–1919.

Thang Nguyen, Peter Chin, and Yu-Wing Tai. 2025. Ma-rag: Multi-agent retrieval-augmented generation via collaborative chain-of-thought reasoning. *arXiv preprint arXiv:2505.20096*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 2463–2473.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.

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

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. 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 (volume 1: long papers)*, pages 10014–10037.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*.

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.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

A Implementation Details

A.1 Software Environment

All experiments are implemented in Python 3.10.14. The main dependencies include PyTorch 2.2.1, Transformers 4.40.0, FAISS 1.8.0, TRL 0.8.6, vLLM 0.4.1, Accelerate 0.30.1, DeepSpeed 0.14.2, and PEFT 0.10.0. We use Llama-3-8B-Instruct as the backbone language model for all agents in MAD-RAG. The Reflective Retrieval Agent $\mathcal{A}_{ret}$, Utility-Guided Denoising Agent $\mathcal{A}_{den}$, and Conflict-Resolving Generation Agent $\mathcal{A}_{gen}$ are initialized from the same backbone and optimized separately under the stage-wise DDR-DPO framework.

A.2 Retrieval and Corpus Configuration

We use BGE-large-en-v1.5 as the dense retriever and retrieve evidence from the MS MARCO 2.1 corpus. For each retrieval call, the retriever returns the top-$k = 10$ candidate passages. The denoising agent then selects at most $m = 5$ passages as the filtered evidence set for generation. In the main MAD-RAG inference experiments, the maximum number of refinement rounds is set to $T_{\max} = 3$. In the efficiency analysis, we additionally evaluate the two-round setting with $T_{\max} = 2$ to isolate the effect of the second-round retrieval decision.

We use MS MARCO 2.1 as a unified retrieval corpus to keep the retrieval environment controlled across all datasets and to isolate the effect of adaptive retrieval control and evidence denoising from corpus-specific engineering. This unified setting ensures that all compared methods retrieve from the same corpus with the same dense retriever. We acknowledge that several benchmarks, such as HotpotQA, T-REx, and WoW, are naturally associated with Wikipedia-style evidence or task-specific knowledge sources. Therefore, our results should be interpreted as evaluating MAD-RAG under a unified open-domain retrieval setting rather than reproducing every dataset's canonical retrieval setup.

A.3 Stage-wise Agent Training

MAD-RAG trains the three agents in a stage-wise manner to reduce non-stationarity among interacting modules. We first train the generation agent $\mathcal{A}_{gen}$ using response-level preference pairs. Then, with $\mathcal{A}_{gen}$ fixed, we construct document-level preference pairs and train the denoising agent $\mathcal{A}_{den}$. Finally, with both downstream agents fixed, we train the retrieval agent $\mathcal{A}_{ret}$ using trajectory-level preference pairs. During inference, no utility rollout is performed; each agent directly predicts its action in a single forward pass.

A.4 DPO Training Configuration

Table 5 summarizes the main DPO training configuration used for the retrieval agent. The retrieval agent is trained with DeepSpeed on four GPUs. We use LoRA-based parameter-efficient tuning, bf16 precision, AdamW optimization, and a ZeRO-2 DeepSpeed configuration. The training data consists of trajectory-level preference pairs constructed from DDR-derived utilities, and the development set is used for periodic evaluation.

Table 5: Main DPO training parameters for the retrieval agent.

Parameter	Value
Training framework	DeepSpeed
Number of GPUs	4
GPU IDs	0, 1, 2, 3
Master port	2346
Backbone checkpoint	Llama-3-8B retrieval checkpoint
Training data	Retrieval DPO training set
Evaluation data	Retrieval DPO development set
Output checkpoint	Retrieval DPO checkpoint
Maximum sequence length	512
Maximum prompt length	428
Per-device train batch size	1
Per-device evaluation batch size	1
Gradient accumulation steps	1
Learning rate	2×10^{-5}
Number of epochs	1
Optimizer	AdamW
Precision	bf16
LoRA	Enabled
DeepSpeed stage	ZeRO-2
Save steps	50
Evaluation steps	50
Logging steps	10
Evaluation strategy	Steps
Logging strategy	Steps
Llama-style formatting	Enabled
Template formatting	Enabled
Flash attention	Disabled

A.5 Generation and Evaluation Configuration

For generator evaluation, we enable retrieval-augmented generation, reranking, LoRA loading, Llama-style formatting, and user chat templates. The current reasoning round is explicitly provided to the model during iterative evaluation. When required by the task setting, the generated answer and its explanation are concatenated for evaluation.

For the main experiments, we use task-specific maximum generation lengths. The maximum gen-

eration length is 64 tokens for NQ, TriviaQA, HotpotQA, T-REx, and WoW, and 128 tokens for MARCO QA. For the WoW evaluation setting, we use the merged Llama-3-8B generator checkpoint, retain top-$m = 5$ evidence passages, set the maximum generation length to 64 tokens, and evaluate with token-level F1.

Table 6: Main generator evaluation parameters.

Parameter	Value
Backbone generator	Llama-3-8B-Instruct
Generator checkpoint	Merged
Retrieval augmentation	Enabled
LoRA loading	Enabled
Reranking	Enabled
Top retained passages	5
Chat format	Llama-style chat template
Current round	1
Answer-explanation concatenation	Enabled
WoW metric	Token-level F1
WoW max new tokens	64

A.6 Evidence Compression Configuration

For the document compression step, we apply a BART-large-CNN compression model to the reranked retrieved documents before downstream evidence use. The compressor takes the reranked document file as input and produces a compressed document file for subsequent reasoning. In our HotpotQA compression setting, the maximum compression length is set to 180 tokens. This step reduces the length of retrieved evidence while preserving information useful for answer generation.

Table 7: Main document compression parameters.

Parameter	Value
Compression model	BART-large-CNN
Input file	Reranked retrieval results
Output file	Compressed retrieval results
Dataset setting	HotpotQA
Maximum compression length	180 tokens
GPU ID	0

A.7 Evaluation Metrics and Reward Instantiation

We use task-specific metrics consistent with the main evaluation protocol. Accuracy is used for NQ, TriviaQA, HotpotQA, and T-REx. ROUGE-L is used for MARCO QA. Token-level F1 is used for WoW. For datasets with multiple reference answers, we compute the score against each reference and report the maximum score. The same task-specific metrics are used as answer-quality rewards $R(q, y)$ during DDR-DPO preference construction.

A.8 Code and Data Availability

To support reproducibility, we provide an anonymous repository containing the implementation code, data processing scripts, prompt templates, configuration files, processed data, and preference pairs used in our experiments:

https://anonymous.4open.science/r/anonym-MAR-RAG-007

The released artifacts include the code for MAD-RAG inference, stage-wise DDR-DPO training, retrieval and denoising pipelines, evaluation scripts, model checkpoints, LoRA weights, and processed preference data for agent training. Original benchmark datasets and retrieval corpora are not redistributed; users should obtain them from their official sources and follow their original licenses or terms of use.

A.9 Artifact Licenses, Release Plan, and Intended Use

We use publicly available datasets, retrieval corpora, pretrained models, and software packages under their original licenses or terms of use. All datasets and corpora are used only for research evaluation. MS MARCO 2.1 is used as an experimental retrieval corpus, and we do not redistribute the original MS MARCO corpus or other original benchmark data. The Llama-3-8B-Instruct backbone is used in accordance with the Meta Llama 3 license. BGE-large-en-v1.5 and BART-large-CNN are used through their publicly released checkpoints for dense retrieval and evidence compression, respectively.

We plan to release the implementation code, model checkpoints, LoRA weights, processed data, preference pairs, and data construction scripts to facilitate reproducibility. The code license has not been finalized at the time of submission. Released processed data and preference pairs will be derived from public benchmarks and automatic evaluation signals, and will be distributed only when compatible with the terms of the original artifacts. We do not use any of the datasets, corpora, or model outputs for commercial deployment or user-facing high-stakes applications.

A.10 Data Privacy and Content Risk

We do not collect new personal data, conduct human-subject experiments, or use human annotators. All experiments are based on publicly available benchmarks and retrieval corpora. We do not attempt to identify individuals, infer sensitive attributes, or construct user profiles from the data. Since open-domain corpora and pretrained language models may contain names, biased statements, factual errors, or offensive content, MAD-RAG may inherit such risks through retrieval or generation. We therefore restrict our experiments to research evaluation and do not recommend direct deployment in medical, legal, financial, safety-critical, or other high-stakes scenarios without human oversight.

When releasing processed data or preference pairs, we will not redistribute sensitive original text beyond what is permitted by the original artifacts. Released examples, scripts, and derived statistics will be checked to avoid exposing private or offensive content beyond the original public resources.

A.11 Dataset and Preference Statistics

Table 8 reports the evaluation statistics used in our experiments. The evaluation subsets are sampled from the official development or test splits of each benchmark and are kept fixed across all compared methods. We use the same subsets for the main comparison and the efficiency analysis, so that performance and latency differences are attributable to retrieval and generation strategies rather than data variation.

Table 8: Dataset statistics used in our experiments.

Dataset	Task	Eval. Size	Metric
NQ	Open-domain QA	2,837	Accuracy
TriviaQA	Open-domain QA	5,359	Accuracy
MARCO QA	Open-domain QA	3,000	ROUGE-L
HotpotQA	Multi-hop QA	5,600	Accuracy
T-REx	Fact lookup	5,000	Accuracy
WoW	Dialogue grounding	3,054	F1

Preference pairs are constructed from public training examples and automatic task-specific reward signals. For the retrieval agent $\mathcal{A}_{ret}$, we construct trajectory-level preference pairs by comparing alternative retrieval actions according to downstream utility. Table 9 reports the statistics of the retrieval-agent DPO data. The dataset contains 98,363 preference pairs in total. Among them, 46,625 pairs are constructed from strengthening correct retrieval behavior, including 29,987 cases where retrieval is preferred after an incorrect answer and 16,638 cases where stopping retrieval is preferred after a correct answer. The remaining 51,738 pairs are constructed from correcting retrieval errors, including 45,722 missed-retrieval cases and 6,016 redundant-retrieval cases.

Table 9: Statistics of retrieval-agent DPO preference pairs.

Category	#Pairs	Ratio
Total retrieval DPO pairs	98,363	100.0%
Correct wrong → retrieve	29,987	30.5%
Correct right → stop	16,638	16.9%
Subtotal: strengthened correct behavior	46,625	47.4%
Missed retrieval	45,722	46.5%
Redundant retrieval	6,016	6.1%
Subtotal: corrected retrieval errors	51,738	52.6%

A.12 Baseline Source and Re-run Protocol

Table 10 summarizes the source of baseline results in Table 1. For baselines already evaluated in RAG-DDR (Li et al., 2025), we directly use the reported numbers to ensure consistency with a prior DDR-based RAG evaluation protocol. For recent agentic or search-based baselines that are not covered by RAG-DDR, including Search-R1, MA-RAG, and A-RAG, we re-run them under the same model and parameter configuration as MAD-RAG. Specifically, the re-run baselines use the same backbone model, retriever, retrieval corpus, top-k retrieval setting, maximum retained evidence number, and generation length whenever applicable.

Table 10: Source of baseline results in Table 1.

Method	Source	Setting
LLM w/o RAG	RAG-DDR	Reported result
Vanilla RAG	RAG-DDR	Reported result
REPLUG	RAG-DDR	Reported result
Self-RAG	RAG-DDR	Reported result
RA-DIT	RAG-DDR	Reported result
RAG-DDR	RAG-DDR	Reported result
Search-R1	Re-run	Same config. as MAD-RAG
MA-RAG	Re-run	Same config. as MAD-RAG
A-RAG	Re-run	Same config. as MAD-RAG

A.13 Computational Budget and Latency Protocol

Preference construction and agent training are conducted on a server with 4 NVIDIA A100 GPUs.

The three agents are based on Llama-3-8B-Instruct. For $\mathcal{A}_{gen}$ and $\mathcal{A}_{den}$, preference construction only requires single-round trajectory sampling, while $\mathcal{A}_{ret}$ requires up to three-round trajectory sampling to estimate the downstream utility of retrieval actions. This makes retrieval-agent preference construction more computationally expensive than generation-agent and denoising-agent preference construction.

Efficiency evaluation is conducted on a single NVIDIA RTX 3090 GPU. Dense embeddings and retrieval indexes are precomputed before evaluation and are not included in the reported latency. Unless otherwise specified, latency includes retrieval calls, evidence denoising, optional compression, and answer generation, but excludes model loading and index construction.

B Prompt Templates

This section presents the prompt templates used for answer generation, retrieval need prediction, and retrieval-augmented generation in MAD-RAG. We omit MiniCPM-specific templates and report the prompts used for Llama-style generation and general task formatting. All prompts require the model to output a valid JSON object with four fields: `answer`, `explanation`, `confidence`, and `gen_retrieval`. The `answer` field stores the predicted answer, `explanation` provides a concise rationale, `confidence` estimates answer reliability, and `gen_retrieval` indicates whether additional external retrieval is needed. This format supports both answer evaluation and retrieval control in the iterative MAD-RAG pipeline.

B.1 General Output Format

The general output prompt standardizes the model response across different tasks. We use a strict JSON format to reduce parsing errors and to ensure that confidence and retrieval need signals can be extracted automatically.

General JSON Output Prompt

```
You MUST output ONLY valid JSON with no extra text.

Required format:
{
  "answer": "answer",
  "explanation": ["reasoning"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. "answer" must be non-empty.
2. "explanation" must contain exactly one non-empty string.

3. "confidence" must contain exactly one integer from 0 to
      10.
4. "gen_retrieval" must be either "Yes" or "No".
5. Do not output any text outside the JSON object.

Confidence grading:
8-10: very confident.
5-7: relatively confident.
0-4: not confident.
```

B.2 Retrieval-Augmented Generation Prompt

When retrieved evidence is available, we prepend it as background information. The model is instructed to use the background only when it is relevant and to avoid unsupported claims when the evidence is noisy or conflicting. This prompt corresponds to the generation step after $\mathcal{A}_{den}$ selects evidence.

Retrieval-Augmented Generation Prompt

```
Background:
{background}

Question:
{question}

Use the background only when it is relevant to the
      question. If the background is noisy, incomplete, or
       conflicting, avoid unsupported claims and rely on
      the most reliable evidence.

Output ONLY valid JSON:
{
  "answer": "answer",
  "explanation": ["reasoning based on the question and
      useful background"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. The answer must be non-empty and supported by the
      question or useful background.
2. The explanation must contain exactly one non-empty
      string.
3. The confidence field must contain exactly one integer
      from 0 to 10.
4. The gen_retrieval field must be either "Yes" or "No".
5. Do not output any extra text outside the JSON object.
```

B.3 Multiple-Choice and Open Question Prompts

For multiple-choice tasks, the model is required to output the option label as the answer. For open question answering, the model outputs a concise exact answer. These templates are used for non-retrieval generation and for candidate answer generation during preference construction.

Multiple-Choice Question Prompt

```
Multiple-choice question:
{question}
```

```
Choose the best answer and output ONLY valid JSON.

Required format:
{
  "answer": "option",
  "explanation": ["reasoning linked to the question"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. "answer" must be a non-empty option, such as "(C)".
2. "explanation" must contain exactly one non-empty
      reasoning string.
3. "confidence" must contain exactly one integer from 0 to
       10.
4. "gen_retrieval" must be either "Yes" or "No".
5. Do not output explanations, labels, or comments outside
       the JSON object.
```

Open Question Prompt

```
Open question:
{question}

Answer the question exactly and output ONLY valid JSON.

Required format:
{
  "answer": "exact_answer",
  "explanation": ["reasoning linked to the question"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. The answer must be non-empty and derived from the
      question or relevant evidence.
2. The explanation must contain exactly one non-empty
      string.
3. The confidence field must contain exactly one integer
      from 0 to 10.
4. The gen_retrieval field must be either "Yes" or "No".
5. Do not output any extra text outside the JSON object.
```

B.4 Chain-of-Thought Prompts

For reasoning-heavy tasks, we use chain-of-thought style prompts that ask the model to include a concise reasoning trace inside the `explanation` field. The reasoning is still constrained to a single JSON string, which keeps the output easy to parse while providing a rationale for confidence estimation and retrieval decisions.

Chain-of-Thought Multiple-Choice Prompt

```
Multiple-choice question:
{question}

Select the correct answer and provide concise reasoning.
      Output ONLY valid JSON.

Required format:
{
  "answer": "option",
  "explanation": ["step-by-step reasoning based on the
      question"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. The answer must be a valid non-empty option.
2. The explanation must contain exactly one reasoning
      string.
3. The confidence score must be an integer from 0 to 10.
4. The gen_retrieval field must indicate whether external
      retrieval is needed.
5. Do not output any text outside the JSON object.
```

Chain-of-Thought Open Question Prompt

```
Open question:
{question}

Answer the question exactly and provide concise reasoning.
       Output ONLY valid JSON.

Required format:
{
  "answer": "answer",
  "explanation": ["step-by-step reasoning using question-
      specific information"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. The answer must be non-empty and question-specific.
2. The explanation must contain exactly one non-empty
      reasoning string.
3. The confidence field must contain exactly one integer
      from 0 to 10.
4. The gen_retrieval field must be either "Yes" or "No".
5. Do not output any extra text outside the JSON object.
```

B.5 Few-Shot Demonstration Template

Few-shot demonstrations are used only to specify the required output structure. The model is explicitly instructed not to copy numbers, entities, or answers from the examples.

Few-Shot Demonstration Template

```
Question:
{question}

Valid output example:
{
  "answer": "{answer}",
  "explanation": ["{explanation}"],
  "confidence": [{confidence}],
  "gen_retrieval": "{Yes/No}"
}

Use the example only to understand the required output
      format. Do not copy numbers, entities, or answers
      from the example. The answer must be specific to the
       current question.
```

B.6 Llama-Style Prompts

The following prompts are used with the Llama-style chat format. They preserve the same JSON schema as the general templates, but use concise instructions better suited to Llama-3-8B-Instruct.

Llama Multiple-Choice Prompt

```
Answer this multiple-choice question and output ONLY valid
       JSON.
```

```
Question:
{question}

Required format:
{
  "answer": "option",
  "explanation": ["reasoning linked to the question"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. "answer" must be a non-empty option, such as "(C)".
2. "confidence" must be a list with exactly one integer
      from 0 to 10.
3. "explanation" must be a list with exactly one non-empty
       string.
4. "gen_retrieval" must be either "Yes" or "No".
5. Output only the JSON object.
```

Llama Open Question Prompt

```
Answer this question and output ONLY valid JSON.

Question:
{question}

Confidence grading:
8-10: very confident; the answer is correct and the
      reasoning has no major flaw.
5-7: relatively confident; the answer is mostly correct
      but may involve minor simplifications.
0-4: not confident; the answer is speculative or key
      information is missing.

Required format:
{
  "answer": "answer",
  "explanation": ["reasoning linked to the question"],
  "confidence": [confidence_score],
  "gen_retrieval": "Yes/No"
}

Rules:
1. The answer must be non-empty and question-specific.
2. The explanation must contain exactly one non-empty
      string.
3. The confidence field must contain exactly one integer
      from 0 to 10.
4. The gen_retrieval field must be either "Yes" or "No".
5. Output only the JSON object.
```