Stop-RAG: Value-Based Retrieval Control for Iterative RAG

Jaewan Park[*]
Seoul National University
jaejae1112@snu.ac.kr

Solbee Cho[*]
Seoul National University
sbcho0325@snu.ac.kr

Jay-Yoon Lee[†]
Seoul National University
lee.jayyoon@snu.ac.kr

Abstract

Iterative retrieval-augmented generation (RAG) enables large language models to answer complex multi-hop questions, but each additional loop increases latency, costs, and the risk of introducing distracting evidence, motivating the need for an efficient stopping strategy. Existing methods either use a predetermined number of iterations or rely on confidence proxies that poorly reflect whether more retrieval will actually help. We cast iterative RAG as a finite-horizon Markov decision process and introduce **Stop-RAG**, a value-based controller that adaptively decides when to stop retrieving. Trained with full-width forward-view Q(λ) targets from complete trajectories, Stop-RAG learns effective stopping policies while remaining compatible with black-box APIs and existing pipelines. On multi-hop question-answering benchmarks, Stop-RAG consistently outperforms both fixed-iteration baselines and prompting-based stopping with LLMs. These results highlight adaptive stopping as a key missing component in current agentic systems, and demonstrate that value-based control can improve the accuracy of RAG systems. Our code is available at https://github.com/chosolbee/Stop-RAG.

1 Introduction

The rise of agentic artificial intelligence (AI) (Sapkota et al., 2025) is rapidly reshaping how large language models (LLMs) are conceived and deployed. Recent research increasingly views LLMs not as standalone responders in single-turn interactions, but as components within broader agentic systems. A central goal in this direction has been to equip LLMs with the ability for *multi-turn tool use*, enabling them to iteratively query external resources, call specialized models, and refine their outputs (Kimi Team, 2025; GLM-4.5 Team, 2025). However, comparatively little attention has been given to the skill of adaptive decision-making, specifically, the ability to judge when further tool calls or further generation are unnecessary and the task can be completed with the information already gathered. This gap highlights a critical yet underexplored dimension in building effective AI systems.

Existing approaches often sidestep this challenge by treating finishing as just another tool call, expecting large-scale training to implicitly teach models when to stop (Yao et al., 2023; Shinn et al., 2023; Singh et al., 2025). However, making this decision fundamentally requires the model to self-reflect and evaluate its own past reasoning trace. As Zhang et al. (2025) observe, most current agents struggle here, leading to long, inefficient sequences of actions.

The challenge of deciding when to stop becomes even more critical in the context of retrieval-augmented generation (RAG) (Borgeaud et al., 2022; Izacard et al., 2023; Shi et al., 2024). Recent RAG systems have also shifted towards employing iterative retrieve-generate loops to solve complex

[*]Equal contribution
[†]Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Multi-Turn Interactions in Large Language Models.

questions (Press et al., 2023; Trivedi et al., 2023; Shao et al., 2023). In deployed systems, however, each additional iteration carries costs—latency, tokens, and the risk of distractors—while benefits vary across queries, making optimal stopping crucial (Shi et al., 2023; Yoran et al., 2024). Hence, deciding when to stop these iterations fundamentally shapes system performance in iterative RAG.

However, existing stopping mechanisms in iterative RAG yet face critical limitations. Some approaches simply rely on LLM's self-assessment (Trivedi et al., 2023; Li et al., 2025; Yue et al., 2025) or internal telemetry signals (Jiang et al., 2023; Su et al., 2024), but both have been shown to be unreliable (Ren et al., 2025; Ni et al., 2024). More sophisticated methods train specialized modules (Asai et al., 2024; Baek et al., 2025), yet their supervision targets are shaped only to reflect the expected answer quality if stopped at the current step. This present-focused training can be misleading: for instance, even if all necessary evidence is already retrieved, the presence of many irrelevant documents may yield a low score and trigger further retrieval, which only adds more noise and harms the final answer. Conversely, an apparently confident answer at an early step might tempt the system to stop, even when continuing would clearly improve completeness.

To address these problems, we reframe iterative RAG as a finite-horizon Markov decision process (MDP) and train a Q-network using Q(λ), a Q-learning variant with λ-returns. This approach yields forward-looking estimates of whether continuing retrieval will improve answer quality. By estimating and comparing both immediate and future gains, our method enables more reliable stopping decisions. Importantly, this does not require internal telemetry and integrates with existing pipelines, making it compatible with black-box LLMs and widely deployable as a modular component.

We summarize our contributions as follows:

- We identify the underexplored problem of adaptive stopping in agentic AI, especially within iterative RAG.
- We propose a novel formulation of iterative RAG as a finite-horizon MDP and train a Q-network using Q(λ) to provide forward-looking estimates of stopping quality.
- We demonstrate that our approach improves performance on multi-hop question-answering (QA) benchmarks, while remaining modular and compatible with black-box LLMs and existing iterative RAG pipelines.

2 Related work

Iterative RAG Iterative RAG systems tackle complex queries requiring multi-step reasoning by running repeated retrieve-generate loops that incrementally accumulate evidence. Compared to single-shot retrieval, this paradigm (1) decomposes complex problems into tractable sub-questions, (2) continuously refines retrieval queries using intermediate hypotheses and findings, and (3) enables self-correction when early retrievals are insufficient. CoRAG (Wang et al., 2025) explicitly learns step-wise retrieval via automatically generated intermediate chains, while Iter-RetGen (Shao et al., 2023) uses prior model outputs to guide subsequent retrieval steps. However, these methods rely on a fixed number of iterations, which leads to fundamental inefficiencies regarding query-specific complexity: simple questions may waste computation through unnecessary loops, whereas difficult ones may halt prematurely before enough evidence is gathered. These limitations motivate adaptive approaches that dynamically decide when to stop.

Adaptive RAG Adaptive RAG methods address the limitations of fixed iteration budgets by dynamically determining when to stop the retrieval process based on query-specific characteristics and intermediate results. Existing approaches in adaptive RAG can be categorized into three main paradigms based on how they make stopping decisions. **Prompting-based** methods allow models to textually decide whether to continue or stop retrieval. IRCoT (Trivedi et al., 2023) and IterDRAG (Yue et al., 2025) interleave reasoning and retrieval, stopping when a final answer is produced. Search-o1 (Li et al., 2025) uses special search tokens during reasoning and terminates upon a final answer. **Confidence-based** methods leverage uncertainty signals to trigger stopping decisions. FLARE (Jiang et al., 2023) uses token-level probability thresholds, while DRAGIN (Su et al., 2024) combines multiple uncertainty measures including entropy and attention weights to determine when retrieval is necessary. **Training-involved** methods learn explicit stopping mechanisms. Self-RAG (Asai et al., 2024) trains a model to generate reflection tokens at each step to guide stopping decisions,

using data distilled from a larger LLM. Probing-RAG (Baek et al., 2025) trains a prober to score if future retrieval is necessary, with supervision from binary labels indicating whether the generated answer at the current step is correct. Adaptive-RAG (Jeong et al., 2024) instead trains a complexity classifier, but operates only pre-execution and cannot be applied within the iterative loop. Unlike these methods, our approach does not rely on model-internal or present-focused signals, but instead provides forward-looking value estimates by framing stopping as a decision in a finite-horizon MDP.

3 Preliminaries

Q-learning In our work, we formulate the iterative RAG setup as a finite-horizon MDP (Sutton and Barto, 1998) and apply reinforcement learning (RL) to optimize the adaptive retrieval strategy. An MDP $\mathcal{M}$ is defined as a tuple $(\mathcal{S}, \mathcal{A}, r, p)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the function that maps a state-action pair to its corresponding transition dynamics distribution. We use the notation $p(s' \mid s, a)$ to denote the probability of transitioning to state s' when action a is taken in state s. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a function that maps a state to a distribution over actions. In value-based RL, specifically Q-learning (Watkins, 1989; Mnih et al., 2013), we take a two-step approach where we first iteratively approximate the optimal action-value function

$$Q^*(s_t, a_t) = \max_{\pi} \mathbb{E}_{p,\pi}\left[\sum_{k=0}^{T-t-1} \gamma^k r(s_{t+k}, a_{t+k}) \;\middle|\; s_t, a_t\right], \tag{1}$$

[3] then derive a greedy policy $\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$. π^* provably converges to the true optimal policy under mild assumptions (Watkins and Dayan, 1992; Jaakkola et al., 1994; Tsitsiklis, 1994).

Full-width Q(λ) for finite-horizon MDPs Function approximation-based Q-learning in its most basic form iteratively performs the following optimization process on an off-policy dataset $\mathcal{D}$:

$$\underset{\theta}{\text{minimize}}\, \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\Bigg(\underbrace{r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} [\![Q_\theta(s', a')]\!] \;\middle|\; s, a\right]}_{=:\hat{Q}(s,a)} - Q_\theta(s,a)\Bigg)^2\right]. \tag{2}$$

[4] As this one-step bootstrap target $\hat{Q}$ is biased, we can unroll the backup over n steps, reducing bias at the expense of higher variance. This yields the following recursive definition of the n-step Bellman optimality backup:

$$\hat{Q}^{(0)}(s,a) = [\![Q_\theta(s,a)]\!], \tag{3}$$

$$\hat{Q}^{(n)}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\max_{a'\in\mathcal{A}} \hat{Q}^{(n-1)}(s', a') \;\middle|\; s, a\right]. \qquad (n > 0) \tag{4}$$

Normally, the expectation over s' and maximization over a' up to the $(n-1)$-th transition are replaced by sampling and integrated into $\mathcal{D}$. However, if the action space is sufficiently small, at each data collection step we can instead build a *full-width tree* over actions (but not over transitions); that is, at every state we evaluate all possible actions, while state transitions are still obtained by sampling. This removes variance introduced by action-level sampling. Let $s'(s,a)$ denote the successor state sampled when taking action a in state s. Then eq. (4) reduces to

$$\hat{Q}^{(n)}(s,a) = r(s,a) + \gamma \max_{a'\in\mathcal{A}} \hat{Q}^{(n-1)}(s'(s,a), a'). \qquad (n > 0) \tag{5}$$

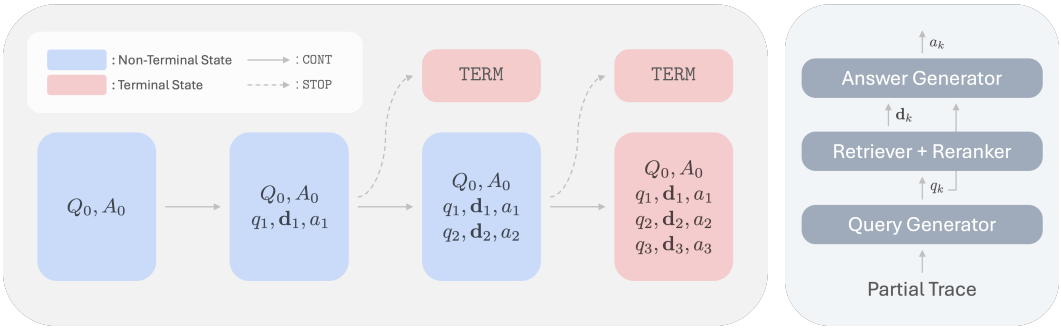
Also, we can further relax these targets into a forward-view Q(λ) (Watkins, 1989; Maei and Sutton, 2010; Sutton et al., 2014) target, which is a continuous interpolation between the discrete n-step targets. Let T the maximum horizon length and $t(s)$ the iteration at which state s occurred. Then the Q(λ) target can be written as

$$\hat{Q}^{\lambda}(s,a) = (1-\lambda) \sum_{n=1}^{T-t(s)-1} \lambda^{n-1} \hat{Q}^{(n)}(s,a) + \lambda^{T-t(s)-1} \hat{Q}^{(T-t(s))}(s,a), \tag{6}$$

where $\hat{Q}^{(0)}$ and $\hat{Q}^{(n)}$ $(n > 0)$ follow the definitions of eq. (3) and eq. (5). Now we can use $\hat{Q}^{\lambda}$ as an alternative to $\hat{Q}$ in eq. (2). We call this resulting algorithm **full-width forward-view Q(λ)**.

[3] The subscript p, π is an abbreviation for $s_{t+1} \sim p(\cdot \mid s_t, a_t)$, $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$, $\ldots$, $a_{T-1} \sim \pi(\cdot \mid s_{T-1})$.

[4] $[\![\cdot]\!]$ denotes the stop-gradient operator.



(a) Iterative RAG as an MDP (max iterations = 3) (b) CONT in detail

Figure 1: Illustration of the iterative RAG pipeline formulated as a finite-horizon MDP. Blue nodes denote non-terminal states while red nodes denote terminal states. At each step, the agent can either STOP (terminate to the absorbing state TERM) or CONT (perform a RAG iteration). In addition, states that reach the maximum iteration limit are also treated as terminal. A RAG iteration consists of query generation (q_k), retrieval and reranking ($\mathbf{d}_k$), and intermediate answer generation (a_k). Rewards are given only at terminal states, reflecting the answer generation quality at that point.

4 Stop-RAG

In this section, we describe Stop-RAG, our *value-based* approach to decide whether further retrieval is necessary in an iterative RAG pipeline. We begin by formulating the iterative RAG process as an MDP, then apply full-width forward-view Q(λ) described in section 3 to derive an effective decision strategy. Our method naturally leverages offline data collected before training, without requiring additional online interaction.

4.1 Iterative RAG as an MDP

A typical iterative RAG pipeline can generally be illustrated as follows. Given a question Q_0 with ground-truth answer A_0, at each iteration of RAG we repeat a sequence of external interactions: (1) generate a query q_k to perform retrieval; (2) retrieve, rerank, and select relevant documents $\mathbf{d}_k$; (3) generate an intermediate answer a_k to the query using the retrieved documents. The process continues until the agent chooses to stop or reaches the maximum allowed number of iterations, at which point a final prediction $\hat{A}_0$ is produced based on all prior interaction records.

In many implementations, both query generation and intermediate answer generation are performed through LLM prompting (Wang et al., 2025; Yue et al., 2025), often within a single inference call (Shao et al., 2023). If the stop decision is also prompt-based, it can likewise be integrated into the same call (Trivedi et al., 2023; Li et al., 2025).

This procedure can be modeled as a finite-horizon MDP, as illustrated in Figure 1. The set of prior interaction records at each iteration becomes a *state*, i.e., the state at the t-th iteration is $\{Q_0, A_0\} \cup \bigcup_{k=1}^{t} \{q_k, \mathbf{d}_k, a_k\}$. We have two possible *actions*: STOP and CONT, which indicate whether to stop the pipeline or not, respectively. Any execution of STOP *deterministically* transitions the system to an absorbing terminal state TERM, while CONT invokes the actual RAG iteration, *stochastically* transitioning to the next state. Note that other than TERM, any state that has reached the maximum allowed number of iterations is also terminal. *Rewards* are assigned only upon transitions to terminal states and reflect the quality of answer generation at that state.

4.2 Stop-RAG training

Specialization of Q(λ) to iterative RAG Building on the MDP formulation of iterative RAG, we now return to the full-width forward-view Q(λ) algorithm described in section 3. In particular, we specialize eq. (6) to our setting. Since we have a small action space ($|\mathcal{A}| = 2$) and one action (STOP) deterministically terminates the process, it is both tractable and natural to construct a full-width tree over actions. We assume no-discounting.

Let T be the maximum number of iterations. For notational simplicity, denote the initial state $\{Q_0, A_0\}$ as s_0, and define subsequent states recursively by

$$s_{t+1} \coloneqq s'(s_t, \texttt{CONT}), \qquad t = 0, \ldots, T-1, \tag{7}$$

under the assumption that the process does not stop prematurely. Note that $s'(s_t, \texttt{STOP}) = \texttt{TERM}$ holds deterministically.

We now derive the Q(λ) targets at an arbitrary state s_t. First, for $a = \texttt{STOP}$, we have $\hat{Q}^{(n)}(s_t, \texttt{STOP}) = r(s_t, \texttt{STOP})$ for $n = 1, \ldots, T-t$. Therefore eq. (6) simplifies to

$$\hat{Q}^{\lambda}(s_t, \texttt{STOP}) = r(s_t, \texttt{STOP}). \tag{8}$$

Next, for $a = \texttt{CONT}$, unrolling eqs. (3) and (5) yields

$$\hat{Q}^{(n)}(s_t, \texttt{CONT}) = \max\left[\left\{r(s_{t+k}, \texttt{STOP})\right\}_{k=1}^{n-1} \cup \left\{[\![Q_\theta(s_{t+n}, \texttt{STOP})]\!], [\![Q_\theta(s_{t+n}, \texttt{CONT})]\!]\right\}\right], \tag{9}$$

$$\hat{Q}^{(T-t)}(s_t, \texttt{CONT}) = \max\left[\left\{r(s_{t+k}, \texttt{STOP})\right\}_{k=1}^{T-t-1} \cup \left\{r(s_{T-1}, \texttt{CONT})\right\}\right], \tag{10}$$

where eq. (9) only applies for $n = 1, \ldots, T-t-1$. Then substituting into eq. (6) gives

$$\hat{Q}^{\lambda}(s_t, \texttt{CONT}) = (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} \hat{Q}^{(n)}(s_t, \texttt{CONT}) + \lambda^{T-t-1} \hat{Q}^{(T-t)}(s_t, \texttt{CONT}). \tag{11}$$

Especially, if $t = T-1$, this collapses to

$$\hat{Q}^{\lambda}(s_{T-1}, \texttt{CONT}) = \hat{Q}^{(1)}(s_{T-1}, \texttt{CONT}) = r(s_{T-1}, \texttt{CONT}). \tag{12}$$

Offline dataset construction To run training on the Q(λ) targets demonstrated above, we construct an offline dataset by generating full interaction trajectories and decomposing them into state-reward pairs. The procedure is as follows.

1. Begin with a corpus of questions Q_0 and corresponding ground-truth answers A_0.
2. For each pair, we run the iterative RAG pipeline without intermediate stopping, always continuing up to the horizon limit T. This yields full-length trajectories containing all interaction records.
3. Each prefix of a trajectory is treated as a partial trace, corresponding to a state in the MDP formulation. We include every such partial trace as a datapoint in the offline dataset. For each state, we estimate rewards by running multiple independent answer generation trials conditioned on that state, and scoring the outputs against A_0. Formally, for $t = 1, \ldots, T-1$:

$$r(s_t, \texttt{STOP}) = \frac{1}{N} \sum_{i=1}^{N} S\left(\hat{A}_0^{(i)}(s_t), A_0\right), \tag{13}$$

$$r(s_t, \texttt{CONT}) = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} S\left(\hat{A}_0^{(i)}(s_T), A_0\right), & (t = T-1), \\ 0, & (\text{otherwise}), \end{cases} \tag{14}$$

   where S is an arbitrary scoring function and $\hat{A}_0^{(i)}(s)$ indicates the prediction generated from state s at the i-th trial. In our experiments, we use F1 scores for S and $N = 8$.
4. Filter out terminal states and states where $r(s_t, \texttt{STOP}) = 0$ and $\hat{Q}^{(T-t)}(s_t, \texttt{CONT}) = 0$. Discarding such states stabilizes training, since they provide no useful signal.

This construction yields an offline dataset that spans all possible actions along each trajectory, while focusing training on informative signals. It enables consistent estimation of Q(λ) targets without requiring online interaction.

Q(λ) training via function approximation We train a function-approximated Q-network on the Q(λ) targets described above. While the full-width formulation eliminates variance from action-level sampling, variance remains due to stochastic state transitions. To mitigate instability, we initialize training with $\lambda = 1$, which corresponds to the longest-horizon targets and reduces bias at the cost of higher variance. As training progresses, we gradually decay λ toward smaller values, thereby interpolating toward shorter-horizon backups that provide lower variance. This annealing schedule balances the bias-variance tradeoff inherent in Q(λ), stabilizing learning while preserving the benefits of long-horizon targets. The complete training procedure is summarized in Algorithm 1.

Algorithm 1 Stop-RAG training with *full-width forward view Q(λ)*

Initialize: Q-network Q_θ, λ, learning rate η

$\mathcal{D} \leftarrow$ sample full trajectories w/o stopping, then split all into partial traces and calculate rewards
while not converged **do**
  sample batch $\mathcal{B} \subset \mathcal{D}$
  $\mathcal{L}_\theta \leftarrow 0$
  for each state $s_t \in \mathcal{B}$ **do** ▷ run at batch-level
    $\hat{Q}^\lambda(s_t, \texttt{STOP}) \leftarrow r(s_t, \texttt{STOP})$
    if $t < T-1$ **then**
      compute $Q_\theta(s_{t+k}, \texttt{STOP})$ and $Q_\theta(s_{t+k}, \texttt{CONT})$ for $k = 1, \ldots, T-t-1$ ▷ no grad
      $\hat{Q}^\lambda(s_t, \texttt{CONT}) \leftarrow (1-\lambda) \sum_{n=1}^{T-t-1} \lambda^{n-1} \hat{Q}^{(n)}(s_t, \texttt{CONT}) + \lambda^{T-t-1} \hat{Q}^{(T-t)}(s_t, \texttt{CONT})$
    else
      $\hat{Q}^\lambda(s_t, \texttt{CONT}) \leftarrow r(s_{T-1}, \texttt{CONT})$
    $\mathcal{L}_\theta \leftarrow \mathcal{L}_\theta + \left[\left(\hat{Q}^\lambda(s_t, \texttt{STOP}) - Q_\theta(s_t, \texttt{STOP}) \right)^2 + \left(\hat{Q}^\lambda(s_t, \texttt{CONT}) - Q_\theta(s_t, \texttt{CONT}) \right)^2 \right]$
  take optimizer step on $\mathcal{L}_\theta$
  adjust λ and η

5 Experiments

5.1 Evaluation setup

Datasets We evaluate Stop-RAG on three *multi-hop QA* benchmarks: MuSiQue (Trivedi et al., 2022), HotpotQA (Yang et al., 2018), and 2WikiMultihopQA (Ho et al., 2020). These datasets require reasoning across multiple retrieval steps, with the number of steps varying by instance. This property makes them particularly suitable for evaluating our proposed framework. For training we use the official 'train' partitions, and from the 'dev' partitions we each randomly sample 1000 questions for validation and another 1000 for testing.

The retrieval corpus differs across datasets. For HotpotQA, we adopt the processed Wikipedia corpus officially released by the authors. For MuSiQue and 2WikiMultihopQA, which were originally formulated as reading comprehension tasks, we construct a retrieval corpus by aggregating all contexts provided in the 'train', 'dev', and 'test' partitions, following Trivedi et al. (2023); Jeong et al. (2024).

Evaluation metrics We report exact match (EM) and F1 scores between the generated prediction and gold answer, following Rajpurkar et al. (2016). We also provide the accuracy (Acc) score, which indicate whether the gold answer appears anywhere in the generated prediction, following Mallen et al. (2023); Schick et al. (2023).

5.2 Baselines

Since Stop-RAG is designed as a plug-and-play stopping module, we evaluate it on top of two distinct iterative RAG pipelines. To cover both sides of the design space, we select one pipeline from prior work and construct another ourselves. CoRAG (Wang et al., 2025) serves as a representative method that involves generator fine-tuning and reports strong performance with a clear, reproducible setup, making it a suitable representative of this class. In contrast, pipelines without generator training vary substantially in prompts and retriever-generator configurations, and typically fall short in reported performance. To provide a fair and competitive counterpart in this category, we implement a modern, general pipeline that incorporates best practices in retrieval and prompting. For reference, we also include in Appendix A a table summarizing reported results of other existing pipelines.

For each pipeline, we compare three variants: the **raw** version, which uses a fixed number of iterations; the **LLM-Stop** version, where the model is prompted to decide when to stop; and the **Stop-RAG** version, which augments the pipeline with our proposed stopping mechanism. This setup allows us to evaluate not only the standalone effectiveness of Stop-RAG, but also its ability to improve performance over both fixed-iteration and naive prompting baselines.

Future work may explore applying Stop-RAG to other high-performing iterative RAG frameworks.

Table 1: Performance comparison on MuSiQue, HotpotQA, and 2WikiMultihopQA. For each pipeline, we report results for the raw version which runs for a fixed number of iterations, a prompting-based stopping variant (LLM-Stop), and the same pipeline augmented with our method (Stop-RAG). Stop-RAG consistently improves performance over both fixed iteration and naive prompting, demonstrating its effectiveness as a plug-and-play stopping module.

Method	MuSiQue			HotpotQA			2WikiMultihopQA		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
CoRAG (L = 10, greedy)	30.9	41.9	36.3	**55.0**	**67.5**	**65.4**	65.3	71.7	68.0
▷ + LLM-Stop	31.1	42.0	36.3	54.9	67.6	64.8	65.4	72.2	67.8
▷ + **Stop-RAG**	**31.5**	**43.0**	**36.9**	54.7	67.4	64.4	**65.7**	**72.5**	**68.2**
Our Pipeline	34.5	44.8	41.1	51.0	65.0	60.8	64.9	73.1	65.9
▷ + LLM-Stop	34.2	44.6	41.3	51.4	65.2	60.7	64.9	73.3	66.1
▷ + **Stop-RAG**	**36.8**	**47.0**	**43.9**	**52.4**	**66.1**	**62.6**	**68.2**	**75.7**	**69.0**

5.3 Implementation details

Our pipeline follows the general formulation of section 4.1. Specifically, we use Llama-3.1-8B-Instruct (Llama Team, 2024) as both the query and answer generator, Contriever-MSMARCO (Izacard et al., 2022) as the retriever, and bge-reranker-v2-m3 (Li et al., 2023; Chen et al., 2024) as the reranker. At each iteration 10 documents are retrieved and the top-ranked one is selected, with a maximum of 10 iterations allowed. For CoRAG, we reimplement the pipeline using the official checkpoint released by the authors and their original prompts, but replace the retriever and corpus with those used in our pipeline. The full set of prompts used in the experiments is provided in Appendix D.

For the Q-network in Stop-RAG, we adopt a DeBERTa-v3-large (He et al., 2023) backbone with two separate prediction heads—each a feed-forward network with one hidden layer—respectively corresponding to the `STOP` and `CONT` actions. While Q-networks in general take both state and action as input, here the network conditions only on the state and predicts outputs for all actions, which is feasible given our small action space. Each state s_t is formatted as the main question Q_0 concatenated with all retrieved documents $\mathbf{d}_1, \ldots, \mathbf{d}_t$, separated by the separator token `[SEP]`. Note that intermediate queries and answers are not included in the state representation, as we found the retrieved evidence alone sufficient to learn effective stopping policies. We provide the complete list of training hyperparameters in Appendix B.

After training, we select the best checkpoint on the validation set. At inference time, rather than following the greedy policy induced by the trained Q-network, we compare the two head outputs and apply a margin-based decision rule: if $Q_\theta(s, \texttt{STOP}) - Q_\theta(s, \texttt{CONT})$ exceeds a tuned threshold, the agent stops retrieval and produces a final answer; otherwise it continues retrieval. This threshold is chosen on the validation set.

6 Results

6.1 Main results

Table 1 presents the overall results of our experiments. Within our pipeline, Stop-RAG consistently outperforms both baselines across all benchmarks. This indicates that naive prompting is insufficient for effective stopping and highlights the importance of a principled stopping mechanism. When applied to CoRAG, Stop-RAG demonstrates its effectiveness as a plug-and-play module. On MuSiQue and 2WikiMultihopQA, applying Stop-RAG achieves higher performance than both baselines, showing that the method generalizes across different iterative RAG architectures. These results emphasize that Stop-RAG is not tied to a specific retriever-generator design, but can integrate seamlessly into varied systems to yield consistent gains.

However, on HotpotQA, Stop-RAG slightly underperforms both baselines. This result likely stems from the fact that CoRAG employs a fine-tuned generator trained under the assumption of a fixed number of iterations. This training setup appears to make the generator more robust to distractors

Table 2: Retrieval performance on MuSiQue and 2WikiMultihopQA, tested on our pipeline variants. Precision and recall are measured for document retrieval, while 'Steps' denotes the average number of iterations executed. Compared to LLM-Stop, Stop-RAG runs slightly longer on average, which increases recall and leads to better performance.

Method	MuSiQue			2WikiMHQA		
	Prec	Recall	Steps	Prec	Recall	Steps
Our Pipeline	17.4	68.2	10.0	21.4	88.3	10.0
▷ + LLM-Stop	31.6	64.5	7.6	55.0	79.1	5.0
▷ + **Stop-RAG**	22.2	66.5	8.6	49.5	82.6	5.1

Table 3: Performance of Stop-RAG variants on MuSiQue when using different learning targets. Detailed descriptions of each method are provided in Appendix C.

Method	MuSiQue		
	EM	F1	Acc
Binary	33.3	43.7	40.3
MC	36.0	46.2	43.0
Q(λ)	**36.8**	**47.0**	**43.9**
Q(0)	36.0	46.3	42.8

and less sensitive to retrieval inefficiency, which may reduce the relative benefit of adaptive stopping. Similar behavior can be observed with CoRAG on other datasets, where Stop-RAG yields only slight improvements compared to its effect on our pipeline. In HotpotQA specifically, where the retrieval corpus is broader and more challenging than the other benchmarks, retrieval recall plays a critical role. Thus, continuing retrieval for the full iteration count appears to outweigh the cost of additional distractors, which may explain the stronger performance of the fixed-iteration baseline.

Overall, these results highlight two key insights: (1) Stop-RAG substantially improves pipelines by identifying more optimal stopping points, and (2) the method demonstrates broad applicability as a plug-and-play component across different architectures.

6.2 Analysis

Retrieval results To further assess the effect Stop-RAG, we analyze retrieval performance on MuSiQue and 2WikiMultihopQA, where the retrieval corpus is manually constructed and ground-truth supporting documents are given. As shown in Table 2, Stop-RAG executes slightly more steps on average than LLM-Stop, resulting in higher recall. This suggests that Stop-RAG improves end-task performance by adaptively continuing retrieval when additional evidence is beneficial, rather than terminating early like LLM-Stop does. In doing so, it achieves a better balance between retrieval depth and precision.

Ablation study To better understand the impact of our learning objective, we performed an ablation study comparing several alternatives to our Q(λ)-based objective. In addition to our main approach with annealed Q(λ) targets, which balance variance reduction with forward-looking credit assignment, we evaluate Monte Carlo (MC) returns (no bootstrapping) and one-step temporal-difference (TD) targets (fully bootstrapped). For clarity, we refer to the one-step TD variant as Q(0), by analogy to TD(0). We also test a binary classification variant, where each state is labeled positive or negative depending on whether the MC return for `STOP` exceeds that for `CONT`. Full details of the training objectives are provided in Appendix C.

As shown in Table 3, Q(λ) achieves the best overall balance, yielding the highest F1 and accuracy. MC performs reasonably well but suffers from high variance, while Q(0) also performs well but is limited by its myopic nature. The binary variant lags behind, suggesting that richer value-based targets are crucial for learning effective stopping policies.

7 Conclusion

We introduced Stop-RAG, a value-based controller for adaptive stopping in iterative retrieval-augmented generation. By framing the problem as a finite-horizon MDP and training with Q(λ) targets, Stop-RAG provides forward-looking estimates of the benefits of continued retrieval, enabling more optimal stopping decisions than fixed-iteration or prompting-based baselines.

Experiments across multiple multi-hop QA benchmarks show that Stop-RAG consistently improves performance, demonstrating its ability to balance retrieval recall and precision while remaining

modular and compatible with black-box LLMs. These results demonstrate the effectiveness of value-based stopping control for iterative RAG and highlight the potential for adaptive decision making in broader LLM-driven systems.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

Ingeol Baek, Hwan Chang, ByeongJeong Kim, Jimin Lee, and Hwanhee Lee. Probing-RAG: Self-Probing to Guide Language Models in Selective Document Retrieval. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3287–3304, Albuquerque, New Mexico, 2025. Association for Computational Linguistics.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving Language Models by Retrieving from Trillions of Tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR, 2022.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv:2403.05530 [cs.CL]*, 2024.

GLM-4.5 Team. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *arXiv:2508.06471 [cs.CL]*, 2025.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. In *The Eleventh International Conference on Learning Representations*, 2023.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online), 2020. International Committee on Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

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

Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 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, Mexico City, Mexico, 2024. Association for Computational Linguistics.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore, 2023. Association for Computational Linguistics.

Kimi Team. Kimi K2: Open Agentic Intelligence. *arXiv:2507.20534 [cs.LG]*, 2025.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. Gecko: Versatile Text Embeddings Distilled from Large Language Models. *arXiv:2403.20327 [cs.CL]*, 2024.

Chaofan Li, Zheng Liu, Shitao Xiao, and Yingxia Shao. Making Large Language Models A Better Foundation For Dense Retrieval. *arXiv:2312.15503 [cs.CL]*, 2023.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic Search-Enhanced Large Reasoning Models. *arXiv:2501.05366 [cs.AI]*, 2025.

Llama Team. The Llama 3 Herd of Models. *arXiv:2407.21783 [cs.AI]*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*, 2019.

Hamid R. Maei and Richard S. Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, pages 91–96. Atlantis Press, 2010.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada, 2023. Association for Computational Linguistics.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs.LG]*, 2013.

Shiyu Ni, Keping Bi, Jiafeng Guo, and Xueqi Cheng. When Do LLMs Need Retrieval Augmentation? Mitigating LLMs' Overconfidence Helps Retrieval Augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11375–11388, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.675.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and Narrowing the Compositionality Gap in Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore, 2023. Association for Computational Linguistics.

Qwen Team. Qwen2.5 Technical Report. *arXiv:2412.15115 [cs.CL]*, 2025a.

Qwen Team. QwQ-32B: Embracing the Power of Reinforcement Learning, 2025b. URL https://qwenlm.github.io/blog/qwq-32b/.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics.

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hua Wu, Ji-Rong Wen, and Haifeng Wang. Investigating the Factual Knowledge Boundary of Large Language Models with Retrieval Augmentation. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3697–3715, Abu Dhabi, UAE, 2025. Association for Computational Linguistics.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *TREC*, volume 500-225 of *NIST Special Publication*, pages 109–126. National Institute of Standards and Technology (NIST), 1994.

Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI Agents vs. Agentic AI: A Conceptual Taxonomy, Applications and Challenges. *arXiv:2505.10468 [cs.AI]*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc., 2023.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore, 2023. Association for Computational Linguistics.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR, 2023.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 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, Mexico City, Mexico, 2024. Association for Computational Linguistics.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 8634–8652. Curran Associates, Inc., 2023.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic Reasoning and Tool Integration for LLMs via Reinforcement Learning. *arXiv:2505.01441 [cs.AI]*, 2025.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12991–13013, Bangkok, Thailand, 2024. Association for Computational Linguistics.

Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 9(5):1054–1054, 1998.

Richard S. Sutton, A. Rupam Mahmood, Doina Precup, and Hado van Hasselt. A new Q(λ) with interim forward view and Monte Carlo equivalence. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 568–576, Bejing, China, 2014. PMLR.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv:2307.09288 [cs.CL]*, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 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, Toronto, Canada, 2023. Association for Computational Linguistics.

John N. Tsitsiklis. Asynchronous Stochastic Approximation and Q-Learning. *Machine Learning*, 16(3):185–202, 1994.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv:2212.03533 [cs.CL]*, 2024.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-Retrieval Augmented Generation. *arXiv:2501.14342 [cs.IR]*, 2025.

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989.

Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. Inference Scaling for Long-Context Retrieval Augmented Generation. *arXiv:2410.04343 [cs.CL]*, 2025.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. RLVMR: Reinforcement Learning with Verifiable Meta-Reasoning Rewards for Robust Long-Horizon Agents. *arXiv:2507.22844 [cs.LG]*, 2025.

A Performance summary of existing iterative RAG pipelines

Table 4 summarizes reported results of prior iterative RAG methods for reference. We include this table primarily to motivate the design of our own pipeline: while several iterative RAG systems exist, their implementations vary substantially in prompts, retrievers, and generators, and reported results are inconsistent across datasets. Scores for Iter-RetGen, Adaptive-RAG, Probing-RAG, IterDRAG, and Search-o1 are taken from their original papers. Results for IRCoT come from Jeong et al. (2024) and results for Self-RAG from Wang et al. (2025), since the original works do not report full results on these datasets. When multiple configurations are reported, we select the setup most comparable to ours, while reporting lower-spec results if they outperform higher-spec variants. The implementation details of each pipeline are as follows.

- **IRCoT** and **Adaptive-RAG**: Uses FLAN-T5-XL (3B) and FLAN-T5-XXL (11B) (Chung et al., 2024) as the generator and BM25 (Robertson et al., 1994) as the retriever.
- **Iter-RetGen**: Uses text-davinci-003 (Ouyang et al., 2022) as the generator and Contriever-MSMARCO as the retriever.
- **Self-RAG**: Uses the same fine-tuned Llama-2-7B (Touvron et al., 2023) checkpoint provided by the original authors but replaces the retrieval system with E5-large (Wang et al., 2024).
- **IterDRAG**: Uses Gemini 1.5 Flash (Gemini Team, 2024) as the generator and Gecko 1B (Lee et al., 2024) as the retriever.
- **Search-o1**: Uses QwQ-32B (Qwen Team, 2025a,b) as the generator and the Bing Web Search API as the retriever.

B Hyperparameters

We train the Stop-RAG Q-network with standard settings. The full list is provided in Table 5.

Table 4: Reported performance of existing iterative RAG methods on MuSiQue, HotpotQA, and 2WikiMultihopQA. Gray-highlighted text denotes results from lower-spec models that outperform larger variants. Missing results ('–') indicate those not reported in the corresponding sources.

Method	MuSiQue			HotpotQA			2WikiMultihopQA		
	EM	F1	Acc	EM	F1	Acc	EM	F1	Acc
IRCoT (3B & 11B)	23.0	31.9	25.8	47.0	57.8	49.4	57.6	67.7	64.0
Iter-RetGen ($T = 7$)	26.1	42.0	–	45.1	60.4	-	35.4	47.4	–
Self-RAG (7B)	4.6	13.2	–	16.6	29.4	–	12.2	24.1	–
Adaptive-RAG (3B & 11B)	23.6	31.8	26.0	44.2	54.8	46.8	47.6	57.4	54.0
IterDRAG (32k)	12.5	23.1	19.7	38.3	49.8	44.4	44.3	54.6	56.8
Search-o1	16.6	28.2	–	45.2	57.3	–	58.0	71.4	–

Table 5: Hyperparameters used for training the Stop-RAG Q-network.

Hyperparamter	Value
Hidden size of prediction heads	4096
Optimizer	AdamW (Loshchilov and Hutter, 2019)
Learning rate	Cosine decay from 5e-5 to 0.0
Warmup ratio	0.1
Weight decay	0.01
Batch size	128
Epochs	3 for MuSiQue, 1 for HotpotQA and 2WikiMultihopQA
Precision	Mixed precision (bfloat16)
λ	Cosine decay from 1.0 to 0.1

C Training objectives of ablation variants

Below we detail the training objectives considered in our ablation study.

Monte Carlo (MC) target A straightforward option is to use Monte Carlo (MC) rollouts of the stopping process. The Q-value is estimated as the immediate reward plus the maximum return achievable from subsequent states:

$$\hat{Q}^{\text{MC}}(s,a) = r(s,a) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}^{\text{MC}}(s'(s,a),a'). \tag{15}$$

For our MDP formulation, we have the following.

$$\hat{Q}^{\text{MC}}(s_t, \texttt{STOP}) = r(s_t, \texttt{STOP}), \tag{16}$$

$$\hat{Q}^{\text{MC}}(s_t, \texttt{CONT}) = \max\left[\left\{r(s_{t+k}, \texttt{STOP})\right\}_{k=1}^{T-t-1} \cup \left\{r(s_{T-1}, \texttt{CONT})\right\}\right]. \tag{17}$$

The training objective here is the same as in our original setup; only the target differs.

Q(0) target We also consider the standard one-step temporal difference target, which is equivalent to the $\lambda = 0$ fixed case of Q(λ):

$$\hat{Q}^{\text{Q(0)}}(s,a) = \hat{Q}^{\lambda}(s,a)\Big|_{\lambda=0} = \hat{Q}^{(1)}(s,a) = r(s,a) + \gamma \max_{a' \in \mathcal{A}} [\![Q_\theta(s'(s,a),a')]\!]. \tag{18}$$

For our MDP formulation, we have the following.

$$\hat{Q}^{\text{Q(0)}}(s_t, \texttt{STOP}) = r(s_t, \texttt{STOP}), \tag{19}$$

$$\hat{Q}^{\text{Q(0)}}(s_t, \texttt{CONT}) = \max\left\{Q_\theta(s_{t+1}, \texttt{STOP}), Q_\theta(s_{t+1}, \texttt{CONT})\right\}. \tag{20}$$

As with the MC target, the training objective is unchanged; only the target values are replaced.

Binary classification Finally, we test a binary formulation where the model directly predicts whether to stop or continue at each state. The label is defined by comparing the Monte Carlo estimates of stop and continue:

$$y(s_t) = \mathbb{1}_{\hat{Q}^{\text{MC}}(s_t,\texttt{STOP}) \geq \hat{Q}^{\text{MC}}(s_t,\texttt{CONT})}. \tag{21}$$

The training objective uses the standard binary cross entropy loss:

$$\mathcal{L}_\theta(s_t) = -y(s_t) \log \sigma(\mathcal{M}_\theta(s_t)) - \big(1 - y(s_t)\big) \log\big(1 - \sigma(\mathcal{M}_\theta(s_t))\big), \tag{22}$$

where $\mathcal{M}_\theta$ is the stopping model parameterized by θ, which we optimize.

D Prompts

This section presents the key prompts used in our experimental setup. Prompts 1 to 3 correspond to those used in the **query generation**, **intermediate answer generation**, and **LLM-Stop** modules, respectively. Blue-highlighted text in brackets indicates placeholders to be filled in.

In particular, Prompt 3 implements the stopping mechanism for the **LLM-Stop** baseline described in section 5.2, enabling the generator itself to decide whether to continue retrieval or stop.

For final answer generation, we employ IRCoT (Trivedi et al., 2023)-style prompting, following the prompt template and in context examples of the original work.

Prompt 1: Query Generation

System Prompt:

Given an original question and a series of follow-up questions and answers, generate the next logical follow-up question that targets a specific missing piece of information needed to answer the original question.

The question will be used to retrieve additional information from a knowledge base (e.g., Wikipedia).

Process:
- You receive:
  - Main question: <original question>
  - Zero or more rounds of:
    - Follow up: <previous follow up question>
    - Document: <top Wikipedia snippet>
    - Intermediate answer: <answer to the Follow-up question based on the Document>

- Your task:
  1. Ask the next logical follow-up question.
  2. Base it solely on gaps in the existing trace.
  3. Do not repeat information already covered.
  4. Make sure it targets exactly the missing fact you need to answer the original question.
  5. Output only the new question itself; no explanations or extra text.

Here are some examples:
`[Few-shot examples of the query generation process]`

User Prompt:

Main question: `[Main question]`
`[Current trace]`

Respond with a simple follow-up question that will help answer the main question, do not explain yourself or output anything else.

Prompt 2: Intermediate Answer Generation

System Prompt:

Given a question with its corresponding Wikipedia snippet, generate an answer using only the provided snippet.

Process:
- You receive:
  - Question: <a question>
  - Document: <Wikipedia snippet for the question>

- Your task:
  1. Focus on answering the question.
  2. Use only the provided Document as your evidence.
  3. Output only the answer, with no additional text.
  4. Keep the answer concise and directly relevant to the question.

Here are some examples:
`[Few-shot examples of the intermediate answer generation process]`

User Prompt:

`[Current query and retrieved documents]`

Prompt 3: LLM-Stop

System Prompt:

Given an original question and a series of follow-up questions each paired with its top Wikipedia snippet plus intermediate answers, analyze step by step whether there is sufficient information to provide a complete and accurate final answer to the original question.

Process:
- You receive:
  - Main question: <original question>
  - One or more rounds of:
    - Follow up: <follow-up question>
    - Document: <top Wikipedia snippet>
    - Intermediate answer: <answer to the Follow-up question based on the Document>

- Your task:
  1. Think step by step about what information is needed to answer the original question.
  2. Analyze what information has been gathered so far.
  3. Determine if any critical information is still missing.
  4. Make a reasoned decision about whether to stop or continue.

Format your response as:
Analysis: Step-by-step reasoning about information completeness
Decision: <STOP> or <CONTINUE>

Decision criteria:
- <STOP>: All components of the original question can be answered with current information such that a complete and final answer can be derived from it.
- <CONTINUE>: Missing any required information needed to answer the original question.

Here are some examples:
`[Few-shot examples of the LLM-Stop process]`

User Prompt:

Main question: `[Main question]`
`[Current trace]`

Based on the information provided, give a short analysis of whether the original question can be answered. Then, decide whether to stop or continue gathering information.