When Does Agentic Retrieval Pay Off? A Controlled Comparison of RAG and Agent-Wrapped RAG on Multi-Hop Question Answering

PIOTR KONDRATOWICZ, University of Twente, The Netherlands

Retrieval-Augmented Generation (RAG) has become the default architecture for answering questions over document collections using Large Language Models. Recently, the process of wrapping a RAG pipeline with iterative Large Language Model (LLM) reasoning, called *agentic retrieval*, in which the LLM iteratively decides what to search for and which documents to read using a set of retrieval tools, has become more popular. This architecture, however, comes at the cost of additional LLM calls, higher latency, and greater token consumption. Both solutions are currently being widely discussed and compared, but rarely in terms of cost and latency. Much of the research lacks both controlled testing of the two solutions and quantitative guidance on when the trade-offs are justified. This research is a study under controlled conditions that compares a standard architecture consisting of a one-shot RAG pipeline against a ReAct agent, which wraps the same retrieval pipeline with iterative LLM reasoning and tool use. The study assesses the results using two complementary benchmarks: MuSiQue, a more shortcut-resistant multi-hop benchmark, and MultiHop-RAG, a multi-document news benchmark used as a contamination contrast. The results are measured in terms of answer quality, token cost, latency, and number of retrieval steps across queries whose complexity varies. This study results in cost-quality curves, which are connected to the difficulty of the question. This outcome allows one to decide when additional costs of the agentic setup are justified by higher accuracy.

Additional Key Words and Phrases: retrieval-augmented generation, agentic retrieval, large language models, document question answering, cost-quality trade-offs

1 INTRODUCTION

1.1 Background and context

Large Language Models are known to perform extraordinarily well at fluent text generation, however, they have static and often outdated knowledge [2, 4]. Retrieval-Augmented Generation (RAG) addresses both of these limitations by retrieving relevant text from external sources and providing context to the LLM [3]. The standard RAG pipeline is one-shot, which means it embeds the user's query, retrieves the top-k most similar document chunks from a vector store, and passes them to an LLM, which then generates the answer that is based on the received chunks [4]. Inspired by the ReAct paradigm [9], agentic retrieval has emerged as a more flexible and better-reasoning alternative to one-shot RAG. Instead of a single retrieval call, an LLM agent has access to a set of retrieval tools, which allows it to make decisions on when and how to use the tools before giving the final answer [4]. There are some systems, such as coding assistants and enterprise file agents, where accuracy is valued over price, and in these cases, agentic reasoning is the primary choice.

TScIT 45, July 3, 2026, Enschede, The Netherlands
© 2026 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

1.2 Problem statement

Both architectures are widely deployed; however, the current literature lacks quantitative guidance on when the additional cost of *agentic retrieval* is justified. Many of the cost-quality comparison findings suffer from at least one of the following weaknesses:

- the two compared systems use different underlying retrieval methodologies, making the comparison unfair;
- reported metrics focus on answer accuracy and omit latency and token cost, which are often core differentiating points;
- query complexity is treated as a binary or anecdotal variable.

This results in practitioners making decisions on whether to add an agentic layer to an existing RAG system based on little quantitative guidance on when the cost is repaid by improved answers.

1.3 Contribution

The research contributes to the discussion about *agentic retrieval* in a couple of aspects, which research currently lacks:

Controlled comparison on a shared substrate. The classical RAG (Setup A) and ReAct agent (Setup B) use the same embedding model, the same vector index, the same document collection and the same Large Language Model. The only difference is the agent's loop. Using this architecture allows one to clearly distinguish the contribution of the agentic loop to any differences in answering.

Setup 0 → Setup A → Setup B decomposition. Apart from the two setups compared in the literature, this research adds a third one: baseline without retrieval (Setup 0), in which the model answers solely from its training knowledge. The difference between setups 0 and A shows how much retrieval itself adds in terms of accuracy. The idea behind Setups A and B is to see how much the loop itself changes.

Cost and quality reported together, in relation to difficulty of a question. Instead of reporting the metrics separately, the results are plotted as curves: answer quality, token consumption and latency, changing with the hop count needed to answer the question. This allows one to analyze for which type and difficulty of questions the agent is worth the premium cost.

Two quality metrics. Next to the classical F1 and exact match metrics, LLM-as-a-judge has been implemented, grounded in the benchmark's evidence. The second model rates whether the answer is correct based on the evidence provided. F1 is known for its issues with answers that are correct, but phrased differently than the gold answer (e.g. "10 years" vs "10-year"), so the second metric aims to fill this gap rather than replace F1.

The character of this work is a case study, over four LLMs and two benchmarks (MuSiQue, MultiHop-RAG, hereafter referred to as MHR). It does not claim to generalize to all models and all data collections. These limitations are named in Section 7.

1.4 Research question

The main research question is:

> *How do answer quality, token cost, and latency differ between one-shot RAG and an agentic retrieval loop sharing the same retrieval substrate, and at what level of query complexity does the agentic loop's quality gain justify its additional cost?*

This is decomposed into four sub-questions:

- **RQ1.** How does answer quality (measured by token-level F1, exact match, and an LLM-as-judge) differ across the no-retrieval baseline, one-shot RAG, and agent-wrapped RAG, and how does each setup's quality change with query complexity measured by hop count?
- **RQ2.** How do token consumption, latency, and interaction depth (retrieval calls and reasoning steps) scale with query complexity for each setup?
- **RQ3.** At which level of query complexity does the agent's quality gain start outweighing its additional cost, and how strongly does this depend on how cost is weighed against quality?
- **RQ4.** How does the number of agent iterations scale with query complexity, and does the agent continue issuing tool calls after it already has sufficient information to answer, incurring unnecessary token cost?

2 RELATED WORK

Retrieval-Augmented Generation. Retrieval-Augmented Generation has already been widely discussed in research papers and was introduced by Lewis et al. [3], where instead of relying only on the model's pretraining knowledge, the retriever returns top-k passages, and the generator conditions on the input together with the retrieved passages to produce the output. Since then, many variants of the original architecture have appeared, ranging from retrieval improvements such as reranking and query rewriting (the enhanced RAG configurations studied by Ferrazzi et al. [1]) to modifications of the generation step. In this work, we use the naive architecture, without additional quality boosts, as the objective of this research is the loop itself, not a comparison between different RAG setups.

Agentic retrieval. Agentic retrieval is closely related to the ReAct framework by Yao et al. [9], where an LLM interleaves reasoning traces with actions specific to the tasks. A recent comparative study of traditional and agentic RAG is given by Neha and Bhati [4]. The most closely related system is IRCoT (Interleaving Retrieval with Chain-of-Thought) by Trivedi et al. [7], in which the model performs a chain-of-thought sequence and, at each step, decides whether it requires additional information to answer the question. This work extends that by reporting cost and quality as functions of query complexity on a shared retrieval substrate that isolates the loop.

Comparison with prior agentic RAG work. The most closely related question is asked by Ferrazzi et al. ("Is Agentic RAG worth it?") [1], where the authors also investigate the feasibility of agentic retrieval, however, Ferrazzi et al. [1] compare the enhanced RAG (with reranking, rewriting, etc.) with the agent, both engineered architectures, on several different types of retrieval tasks, without a query complexity differentiator. This work takes the opposite framing: it compares the agent against naive RAG, on a shared retrieval substrate on multi-hop questions, reporting cost and quality as a function of hop count to isolate the agent's loop as cleanly as possible.

MuSiQue benchmark. This work uses the MuSiQue benchmark, created by Trivedi et al. [6]. The choice of this benchmark was deliberate, mainly for three reasons: (i) marked hop counts, allowing questions to be stratified according to difficulty, (ii) annotations identifying the supporting paragraphs, which make retrieval recall measurable, (iii) resistance to shortcuts: the questions are composed and filtered and later steps depend on earlier-hop answers. This differentiates MuSiQue from older benchmarks such as HotpotQA [8], which was later shown to be more vulnerable to shortcut reasoning than MuSiQue [6].

MultiHop-RAG benchmark. As a second, contrasting benchmark we use MultiHop-RAG [5], built on a news corpus from late 2023, designed to reduce the risk that some widely used LLMs at the time, including ChatGPT and LLaMA, had already been exposed to the content during training. Although it is framed around multi-hop queries, its questions are answered by aggregating or comparing facts across several documents, not by following a chain in which each step depends on the previous answer. This study is built to isolate exactly that sequential dependency, so we use MHR as a multi-document and contamination contrast rather than as a second multi-hop benchmark (Section 3).

3 METHODOLOGY AND APPROACH

3.1 Experimental Design

This research is a controlled, paired comparison of the three setups named before: Setup 0 (LLM alone), Setup A (RAG pipeline), and Setup B (agentic retrieval). Setups A and B share the same retrieval substrate, having the same: (i) embedding model, (ii) chunking strategy, (iii) vector index, and (iv) corpus. The only difference between these setups is the agent's loop and minor differences in prompt framing. Because of this design choice, differences cannot be attributed to the retriever, index, or corpus; they reflect the agent-wrapped architecture, including the loop and its prompt framing. Within each model, all three setups use that same model, so any difference between setups is attributable to the setup rather than model or question sampling. The comparison is paired: the same questions are asked of all three setups, and compared question by question.

3.2 Setups

Setup 0 — LLM (baseline without retrieval). In this setup, the question is passed directly to the LLM, which then produces an answer without any retrieval step, only from its training knowledge. It serves as a baseline and shows how much retrieval adds to accuracy.

Setup A — Standard RAG (baseline). Query → embedding → top-k semantic search → retrieved chunks placed in prompt → single LLM generation call. This is the naive RAG pipeline: the model is given the retrieved paragraphs and answers based on them.

Setup B — Agent-wrapped RAG. Query → the agent reasons about which tool to use → semantic search OR list files OR produce the final answer → repeat if no final answer was produced. A ReAct-style agent [9] receives the query and is exposed to two retrieval tools: `semantic_search(query)` and `list_files(filter)`. The agent reasons in a loop, calling tools as needed, and terminates with a final answer. The loop is capped at 8 tool-calling iterations; if the agent has not answered by then, one additional forced final-answer call is made, so the logged number of model calls per query can reach 9.

3.3 Model and infrastructure

Four language models from four different families are evaluated as systems across Setups 0, A, and B: Qwen3.5, MiniMax M2.7, Claude Haiku 4.5, and GPT-4o mini. They span both reasoning and non-reasoning architectures, allowing the findings to be tested for robustness across model types rather than on a single model. Qwen3.5 and MiniMax M2.7 are accessed through an endpoint hosted at the University of Twente. Claude Haiku 4.5 and GPT-4o mini are accessed through their respective provider APIs. All system calls use temperature 0 for reproducibility, with the maximum output length capped. Four system models plus an independent judge (five models total) are summarised in Table 1. For embeddings, the all-MiniLM-L6-v2 model has been used, a widely-used, lightweight sentence-embedding model.[1] Because the embedding model is held constant across Setups A and B, retrieval quality is not an independent variable in this study. A compact, well-established model is therefore sufficient, and the choice does not affect the comparison of the agentic loop. The vector database was implemented using Chroma, a lightweight embedded vector store with a persistent on-disk index, so the corpus was embedded only once.[2] Same as the embedding model, the vector store is held constant across Setups A and B and is therefore not an independent variable. For assessing quality through an LLM-as-a-judge, DeepSeek-V4-Flash model has been used. The use of a model from a different LLM family is deliberate: it avoids exact self-evaluation and reduces the risk of self-enhancement bias, a known concern in LLM-as-a-judge evaluation [10].

Table 1. Language models evaluated as systems (Setups 0/A/B) and the independent judge. The four systems span four model families and both reasoning and non-reasoning architectures; the judge belongs to a fifth family, reducing self-preference bias. All calls use temperature 0 and a capped maximum output length.

Model	Family	Role	Reason.	Tool calls
Qwen3.5 (think. off)	Alibaba	System	No	Prompted
MiniMax M2.7	MiniMax	System	Yes	Native
Claude Haiku 4.5	Anthropic	System	No	Native
GPT-4o mini	OpenAI	System	No	Native
DeepSeek-V4-Flash	DeepSeek	Judge	No	—

[1] all-MiniLM-L6-v2: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[2] Chroma: https://www.trychroma.com

3.4 Dataset and sampling

For this research, two benchmarks with different characteristics have been used: (i) MuSiQue [6] as the main benchmark and (ii) MultiHop-RAG [5] as a contrast.

MuSiQue. This benchmark [6] is a multi-hop QA benchmark built on the Wikipedia corpus, in which each question is a composition of single-hop questions. This benefits the research in two important ways. Firstly, it gives a clearly defined number of hops, which allows the experiments to be stratified by difficulty. Secondly, MuSiQue provides annotations identifying the supporting paragraphs (gold supporting paragraphs), which makes retrieval recall measurable. The sample consists of $n = 400$ questions. Each model is asked the same set of questions, evenly distributed across difficulty ($n = 100$ for each of 1-, 2-, 3-, and 4-hop). The sample is drawn deterministically ($seed = 42$) and paired across setups. MuSiQue-Answerable natively contains only 2–4-hop questions, so the 1-hop questions were derived from the `question_decomposition` field of the validation split. Each decomposition step is a single-hop question with its own answer and its own supporting paragraph. As 1-hop candidates, only the steps satisfying all of the following criteria were kept: (i) a non-empty question and answer; (ii) no # placeholder (which would make the question depend on a previous step's answer); (iii) no `entity >> relation` template; and (iv) a valid supporting-paragraph index. Candidates were deduplicated by question content, and 100 questions were sampled from the resulting pool. The gold answer for each such question is the answer of the corresponding step, and its supporting paragraph is the single paragraph at the annotated index. This, however, raises the gold-noise in the 1-hop bin, because some sub-questions depend on the full question. Therefore, F1 measured there should be treated as a lower bound of true accuracy. The retrieval corpus for Setups A and B consists of 21,100 deduplicated paragraphs (a Chroma collection). Each paragraph forms one chunk, with no overlap. Retrieval is open-set: every query searches the whole corpus, not just the reduced candidate set provided with each question.

MultiHop-RAG (MHR). This benchmark [5] is a QA benchmark built on the world news corpus from late 2023 (September–December) designed so that its content postdates the cutoff of contemporary models. Its authors frame it as a multi-hop benchmark. We use it differently here, and the reason is worth stating plainly. A MuSiQue question forces an ordered chain: you need the answer to one hop before you can even ask the next. MHR questions do not behave that way. They gather several facts about one entity, or check several conditions and then compare them, and each of those facts can be retrieved on its own, in any order. This is visible even in the benchmark's own query taxonomy: inference, comparison, and temporal queries all describe aggregation or comparison over independently retrievable facts, not the dependent chaining that defines multi-hop reasoning. For that reason, MHR is treated as a multi-document and contamination contrast rather than as a second multi-hop benchmark.

Table 2 makes the difference concrete. A MuSiQue question forces an order: its second step is written in terms of the first answer (#1), so it cannot be retrieved until the first hop resolves. The MHR

comparison question names both of its sources up front, and each fact can be pulled on its own, in any order. Neither fact references the other.

Shared substrate. In both benchmarks, Setups A and B share the same retrieval substrate: the embedding model (all-MiniLM-L6-v2) and Chroma vector store with persistent index. The corpus is embedded just once and all model calls use $temperature = 0$. Every model answers the same set of questions across all three setups.

Table 2. Structural difference between the two benchmarks.

MuSiQue, 2-hop: dependent chain
Who is the spouse of the Green performer? (gold: Miquette Giraudy)

Hop	Sub-question	Answer
1	Who performed *Green*?	Steve Hillage
2	Who is the spouse of Steve Hillage?	Miquette Giraudy

MHR, comparison (2-doc): independent facts
Do the TechCrunch article on software companies and the Hacker News article on The Epoch Times both report an increase in revenue related to payment and subscription models, respectively? (gold: Yes)

	Source	Fact
A	TechCrunch (Rainforest)	"…accept payments and generate more revenue…"
B	Hacker News (Epoch Times)	"…$76M subscription revenue in 2021 vs. $7M in 2019."

3.5 Metrics

Each question in each setup is evaluated in terms of:

- **Answer quality** (F1 and EM): token-level F1 and exact match against the gold answer provided by the benchmark, computed using the benchmark's official evaluation script.
- **Answer quality** (LLM-as-a-judge): the LLM judge is given the query, the supporting paragraphs provided by benchmark, the model's answer, and the gold answer. Based on this information, the judge labels the answer as (i) CORRECT, (ii) PARTIAL, or (iii) INCORRECT, along with an explanation. The judge is grounded in the evidence rather than in its parametric knowledge. This addresses a limitation of token-level F1, which penalizes answers that are correct but phrased differently from the gold answer.
- **Cost**: input and output token consumption, reported separately and jointly per question.
- **Latency**: wall-clock time spent per question.
- **Interaction depth**: the number of tool calls and reasoning steps (Setup B only).
- **Retrieval quality**: recall of the benchmark's supporting paragraphs among those reached by the setup.
- **Over-exploration (Setup B only)**: the number of agent steps taken *after* the step at which the complete set of gold supporting paragraphs was first retrieved. A query contributes to over-exploration only if full recall was reached before the agent stopped. The mean number of such surplus steps and the mean number of tokens they consume are reported as a measure of wasted effort. The number of additional steps is counted by taking the total number of steps the agent used, minus the step at which gold evidence was retrieved, minus 1 (mandatory answer turn not counted toward the limit of 8 iterations). Only queries that self-terminated are taken into account.
- **Statistical significance**: paired bootstrap CIs on mean $B - A$. CI excluding $0 \rightarrow$ significant (Table B.9).

3.6 Implementation

The entire pipeline is coded in Python. The ReAct agentic loop has been implemented from scratch, on top of OpenAI's SDK, without predefined frameworks (e.g. LangChain, LlamaIndex). This choice was deliberate, as such frameworks often hide the metrics measured in this research: token consumption per call, latency per step, or tool calls. Tool calls in Setup B are made through prompted tool calling and native tool calling, depending on the model. Each call is logged to disk with: the question, setup, number of iterations, fetched paragraphs, input and output tokens, latency, final answer, gold answer, F1, and exact match.

3.7 Analysis

Quality, costs, and latency are presented as curves as a function of the hop count. This allows the metrics to be seen and analyzed as the difficulty rises. At the end, the cost effectiveness range is identified, which is the level of complexity at which the gained accuracy has the best cost-quality ratio. To quantify over-exploration (RQ4), each Setup B run is replayed against the gold supporting paragraphs. The agent's step at which full recall is first reached is identified. This means that any further tool-calls and iterations were essentially not needed. The token cost of these additional steps is summed, and contributes to wasted tokens. Queries that never reached full recall are excluded from this measure.

4 RESULTS

The results draw on two benchmarks of different character. MuSiQue is the main one: multi-hop questions over Wikipedia, where the low Setup 0 scores show that the models rarely know the answer from training alone, so retrieval pays off. MultiHop-RAG (MHR) plays the opposite role. It is built on news from late 2023, meant to sit past the cutoff of contemporary models [5]. The models used here have later cutoffs, so they already know these events from training. Side by side, the two benchmarks show that the agent's advantage depends heavily on whether retrieval is actually needed, and that is exactly what Setup 0 measures.

4.1 Quality across setups (RQ1)

MuSiQue and the $0 \rightarrow A \rightarrow B$ decomposition. The pattern is consistent across all four models, both in the token-level F1 results (Table B.1) and under grounded LLM-as-a-judge (Table B.2). Retrieval helps, but only at the bottom of the difficulty range. Going from Setup 0 to Setup A lifts 1-hop F1 sharply, from about 0.37 to about 0.70 on average, a gain that runs from +0.28 (Haiku 4.5) to +0.37 (Qwen3.5) depending on the model. It does not survive added complexity. By 2-hop the gap between Setup 0 and A is no longer visible, and at 4-hop retrieval does not help, often dragging accuracy

below the no-retrieval baseline. The lesson is narrow: a single top-k retrieval works only when the whole answer already sits in the retrieved context. The agent's loop tells the opposite story. Setup B brings relatively small advantage over Setup A at 1-hop, where one retrieval is enough and the extra iteration only adds costs; a clear advantage first appears at 2-hop and holds through 3- and 4-hop. Under the judge, Setup B wins every 2–4-hop bin for every model (Fig. 1). Under F1 the same holds for three of the four. Haiku 4.5 is why the judge was needed. Under F1 these two are statistically similar ($B - A = -0.031$, 95% CI $[-0.071, +0.009]$), F1's disadvantage is within noise. The judge shows a large and significant advantage ($B - A = 0.315$, CI $[+0.269, +0.364]$), F1 fails to capture the effect the judge captures decisively. Full intervals are in Table B.9. The same Setup B is the strongest agent of any model (0.589 overall), and it clears Setup A by a wide margin. The cause is that Haiku 4.5 answers correctly but at length, rather than with the short answer it was asked for, so token-F1 lines up a full sentence against a one-word gold answer and penalizes it. A manual audit of 48 low-F1 cases backs this up: a quarter of them came from formatting/alias mismatches or gold-answer errors, not wrong answers, and the judge agreed with the human reviewer in 89.6% of cases. Figure D.1 shows the split. Exact match (Table B.3) follows the same ordering as token-F1: Setup B leads on three out of four models and reproduces the Haiku 4.5 inversion, since the strictest overlap metric is penalized the same way by correct but verbose answers.

MHR, where Setup 0 performs great. What stands out first on MHR is how high Setup 0 sits: F1 of 0.503 (M2.7), 0.817 (Qwen3.5), 0.747 (GPT-4o mini), and 0.541 (Haiku 4.5, Table C.10), roughly two and a half to about four times the MuSiQue baseline of about 0.2. The models answer most questions correctly with no retrieval, straight from training. Setup A often lands below Setup 0 here, which looks odd until the reason is clear: on the yes/no comparison and temporal questions, Setup A retrieves and then abstains ("I don't know") rather than guessing, while Setup 0 guesses with a yes-bias and is rewarded by the base rate. That is an artifact of the question format, not of wrong retrieved evidence. Haiku 4.5's F1 versus judge split reappears here as well: its Setup B collapses under F1 (0.169 overall) yet is the best Haiku 4.5 setup under the judge (0.753, Table C.11), so its F1 row should never be read without the judge row beside it.

4.2 Cost, latency, and interaction depth (RQ2)

The cost ordering is the same for almost every model. For three of the four, the setups separate cleanly. Setup 0 stays low and flat, around ≈80–150 tokens per query (for three instruct models, M2.7 reasoning inflates its baseline), Setup A is just slightly higher and just as flat, near 500 tokens per query and Setup B is far more expensive, with the steepest growth in token use, running from about 7,500 to 19,000 tokens per query (Fig. 2). As a ratio, the agent costs between 16 and 35 times one-shot RAG.

Rapid growth of Setup B's token consumption. This is not surprising, as it makes on average 5.94 steps on the MuSiQue benchmark including the last answer step (Fig. D.2). That's roughly five more LLM calls per question than one-shot RAG, and therefore is the most expensive option out of all setups. The cheapest systems turned out to be GPT-4o mini (Setup A 464, Setup B 7,551 tokens) and Qwen3.5 (Setup A 504, Setup B 9,064), both well below the other two, Table B.4.

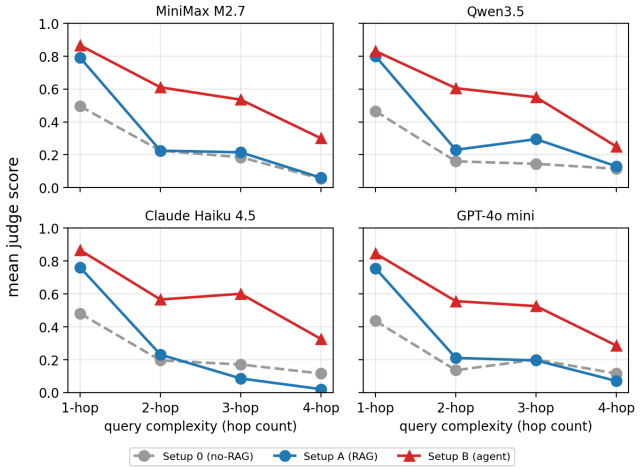

Fig. 1. Answer quality under the grounded LLM-as-a-judge by setup and hop count for each model on MuSiQue. Under the judge, Setup B outperforms Setups 0 and A at every 2–4-hop bin for all four models, including Haiku 4.5 - reversing the F1 picture in Fig. D.1.

M2.7 consumes more tokens. It is the most expensive model in every setup, due to its reasoning nature. The reasoning tokens inflate even Setup 0 and A above other models. The same reasoning is visible also in latency: its no-retrieval Setup 0 already averages 95 seconds per query, while three other models answer Setup 0 in under two seconds, Table B.5.

Monetary cost. At rates current during the experiments (Claude Haiku 4.5: \$1.00/\$5.00 per million input/output tokens; GPT-4o mini: \$0.15/\$0.60), GPT-4o mini is substantially cheaper across all setups (\$0.0000143, \$0.00007, \$0.00117 for Setups 0, A, and B respectively), while Claude Haiku 4.5 costs 10–30 times more at the same setups (\$0.00043, \$0.00072, \$0.02150), with the gap widening to roughly 18 times at Setup B despite comparable judge scores (Table B.2, Table B.7); Qwen3.5 and MiniMax M2.7 ran on an institutional endpoint and no public per-token pricing is available for those deployments.

4.3 Cost-quality trade-off (RQ3)

Agent is never cheaper. On each hop the token cost of Setup B is higher than Setup A's, therefore the cost curves never cross, meaning there is no direct crossover point.

Quality gain and cost scale differently with difficulty. The token cost ratio (B/A) rises monotonically with hop count, but the quality gain (judge $B - A$) does not (Fig. 3, left side). On 1-hop queries the gain is marginal, approximately 0.08 by judge: one-shot retrieval is sufficient and the agent's loop does not help, so the extra cost is not justified. The gain jumps to 0.36 at 2-hop, holds at 0.36 at 3-hop, then falls to 0.22 at 4-hop, even as the token ratio keeps climbing.

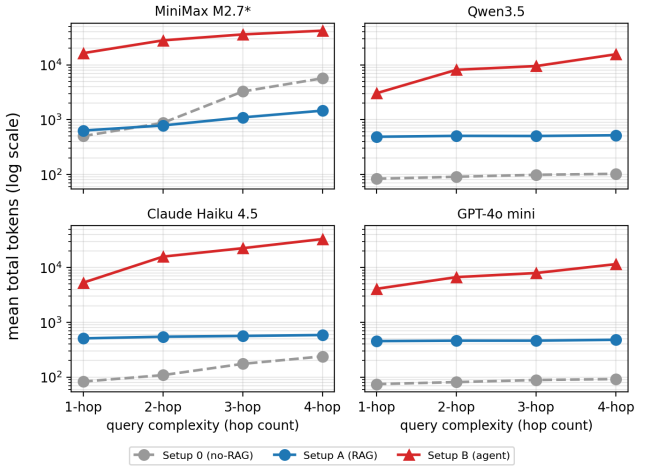

Fig. 2. Mean total tokens per query by model, setup, and hop count on MuSiQue (log scale). Setup B token cost grows with hop count while Setup A stays nearly flat. *MiniMax M2.7 Setup 0/A counts include <think> reasoning tokens, which inflate its baseline above Setup A.

The effectiveness peaks at the queries of medium difficulty. The gain per unit of token cost is highest at 2–3-hop and drops at 4-hop (Fig. 3, right panel). The reasons lie in the iteration limit set to 8 due to costs. On the most difficult questions the agent hits the iteration ceiling and returns the forced answer, based on not fully retrieved information. Therefore, the quality gain dips, but the cost stays at the maximum. This analysis is on MuSiQue only, on MHR the trade-off is questionable, since retrieval is not needed.

4.4 Iteration scaling and over-exploration (RQ4)

Iterations scale with complexity. The mean number of agent steps rises with hop count, from 3.5 steps at 1-hop questions (inflated by M2.7 taking on average 5 iterations) up to 7.9 at 4-hop questions (average across all models). At 4-hop questions, the 8-step bound is hit by the vast majority of M2.7 queries (96%) and most Haiku 4.5 queries (77%), versus under half for GPT-4o mini (43%) and just over half for Qwen3.5 (53%), which confirms that the most difficult queries make the agent hit the iteration ceiling (Fig. 4, left).

The agent over-explores. Once full gold recall is reached, three of the four models stop almost at once. Their median number of extra search steps is zero, and the mean stays low (Qwen3.5 0.21, GPT-4o mini 0.46, Haiku 4.5 0.08). M2.7 is the exception. The reasoning model keeps searching after the evidence is complete, taking on average 2.06 extra tool-calling steps in 75% of its self-terminated queries. Those extra steps cost it roughly 5,400 tokens per query, about 53% of its token consumption on those queries, with no gain in evidence. The wasted cost concern is therefore real but localized to the reasoning model, not a general property of the agentic loop (Fig. 4, right).

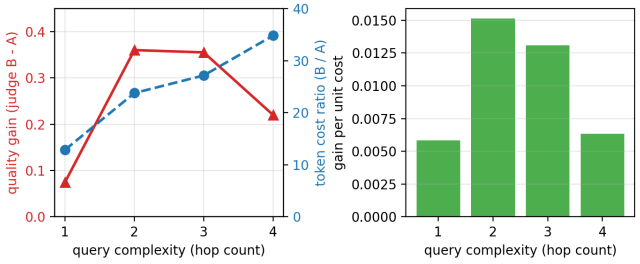

Fig. 3. Cost-quality trade-off on MuSiQue (averaged over four models). Left: the agent's token cost ratio (B/A) rises monotonically with complexity, while its quality gain over one-shot RAG (judge B−A) peaks at 2–3-hop and declines at 4-hop; the agent is never cheaper. Right: quality gain per unit of token cost peaks at 2–3-hop and declines at 4-hop, where the 8-step bound limits the agent. The agent is most cost-effective on moderately complex queries, not the hardest ones. Cost-effectiveness is defined per hop bin as the mean judge gain divided by the mean token-cost ratio, $\left(\overline{\text{Judge}_B} - \overline{\text{Judge}_A}\right) / \left(\overline{\text{Tokens}_B / \text{Tokens}_A}\right)$, with both terms averaged over the four models.

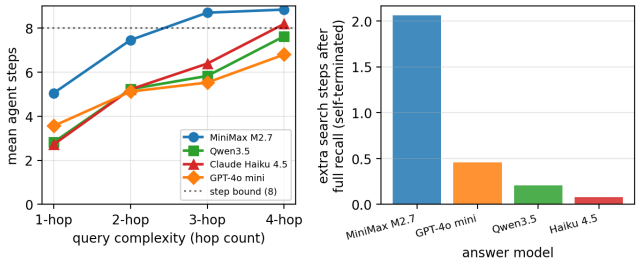

Fig. 4. Iteration behaviour of Setup B on MuSiQue. Left: Mean agent steps rise with hop count for every model. The deepest (M2.7 and Haiku 4.5) approach the 8-step bound at 4-hop. Right: Voluntary over-exploration: extra search steps taken after full gold recall, on self-terminated queries (excluding the mandatory answer turn and step-bound-forced runs). Over-exploration is concentrated in the reasoning model M2.7 (mean 2.06 extra steps, 75% of queries). Remaining models stop almost immediately once evidence is complete (median≈0).

4.5 Retrieval quality

Recall explains agent's advantage in accuracy. Fig. 5 shows why Setup B pulls ahead of the other setups at higher complexity. Setup A is limited by a set retrieval of top-k chunks. It always retrieves these three fragments, so when the question requires a greater number of supporting paragraphs, recall dips rapidly, decreasing from 0.77 on 1-hop questions down to 0.18 on 4-hop questions (full set; the $n = 150$ $k = 3$ value in Table B.6 is 0.205). The curve is identical throughout all models, because it depends only on the retrieval, not the model. Setup B does not have this limitation. Thanks to the iterations, the agent retrieves the next paragraphs in a couple of retrievals, and therefore its recall is substantially higher: from 0.86 (close to Setup A) on 1-hop questions to 0.52 at 4-hop questions (mean over the four models, full set). The difference arises on 2-hop questions, where Setup A dips sharply and Setup B declines far more gradually. The curve depends on the model (the red shaded stripe

on Fig. 5 is the difference between the models): Haiku 4.5 achieves the highest recall, GPT-4o mini the lowest.

Bigger retrieval window does not close the multi-hop gap. One could ask if the agent's advantage does not come from the poor retrieval of top-k ($k = 3$). To assess that, we ran the questions through the same retrieval, with $k = 3, 5, 10, 20, 30, 50$ and calculated the recall of the gold paragraphs ($n = 150$, MuSiQue, Table B.6). Recall increases with k, but slowly and unsteadily. On 4-hop questions the recall rises from 0.205 with $k = 3$ just up to 0.435 with $k = 50$. A more strict measure of full-recall (the complete set of gold paragraphs in top-k) indicates that at 4-hop it is 0% ($k = 3$) and 6% for $k = 50$. The reason is structural. Bridge paragraphs, such as the birthplace of X, are not semantically similar to the original question, so a single embedded query will not retrieve them at any k. This is what iteration fixes: on these same 150 questions, at 4-hop the agent reaches 0.555 recall, above single-shot retrieval at $k = 50$ (0.435). The advantage comes from the loop, not from a bigger window, which is why the deliberately naive $k = 3$ baseline is justified.

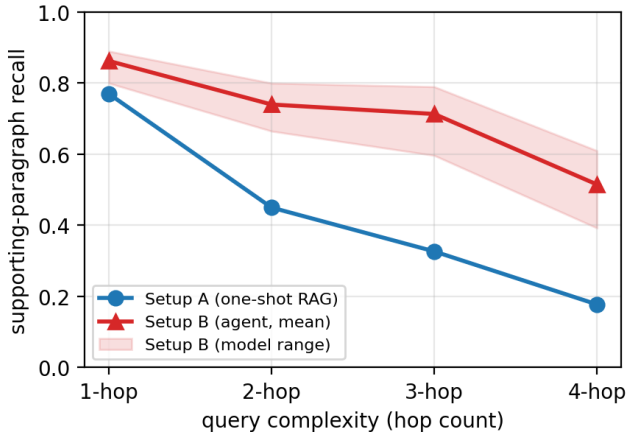

Fig. 5. Supporting-paragraph recall by setup and hop count on MuSiQue. Setup A recall is fixed by the top-3 retrieval window and falls from 0.77 at 1-hop to 0.18 at 4-hop. Setup B recall (mean across four models; shaded band shows model range) stays substantially higher at 2–4 hop, explaining the agent's quality advantage at higher complexity.

5 DISCUSSION

The most practical conclusion from RQ1-RQ3 is that the use of the agent's loop is not all-or-nothing. It is worth its premium, 16 to 35 times the token cost, only for specific use cases. These include multi-hop questions for which the model does not already know the answer from training and which require 2–3 hops. In most other cases it would be a waste of tokens. However, when accuracy matters most and price is not a factor, the agentic system might be preferable. Because in many situations the difficulty of the query varies (e.g. a mix of all hop counts), routing would be a good, natural solution: by default the cheap one-shot RAG is used, and if the question appears to be multi-hop, the agent is used.

There are several use cases where these systems can be applied. The loop is worth its price when the answer requires retrieving from multiple scattered sources that the model does not know from training. Examples include internal databases, technical documentation, case files, and questions linked across many documents that cannot be answered with one query. By contrast, for factual questions about well-known news, or one-hop questions, the loop is an unnecessary cost and latency overhead without a big difference in accuracy. Where latency is crucial (for example real-time assistants, writing hints, etc.), the one-shot RAG stays the default choice, because the difference between less than a second and 6–22 seconds is huge while the accuracy is often at a similar level. Another axis is the choice of the model, which can influence the latency (in Setup B: M2.7 26.8 s, GPT-4o mini 6.3 s on MuSiQue).

Two observations go beyond the comparison of setups. The first is that the Haiku 4.5 inversion shows that token-level F1 can switch the verdict on whether the method helps, if the model answers correctly but verbosely. That is why a knowledge-grounded judge should be a standard metric alongside F1, not an optional add-on. The second is that MultiHop-RAG shows that a benchmark based on recent news has an "expiry date" connected to the models' knowledge cutoff. This case shows why Setup 0 is useful as a contamination checker: a high result without retrieval means that we are measuring memory, not retrieval. These two points should serve as a caution when building future RAG comparisons.

6 CONCLUSIONS

6.1 Main findings

Wrapping a one-shot RAG pipeline in an agentic loop helps, but only under two conditions: (i) retrieval is necessary and (ii) the question requires more than one hop. On MuSiQue, the poor results of Setup 0 show that the models do not know the answers from training, and Setup B wins every 2–4-hop bin under the judge; under F1 this holds for three of four models, Haiku 4.5 excepted. On MultiHop-RAG the models largely answer from training, so the agent's quality benefit does not repay its cost premium. The mechanism that makes the agent win on multi-hop queries is recall, which allows the agent to find bridge-paragraphs that are inaccessible to a single top-k retrieval, and enlarging the retrieval window does not solve this issue. The iteration loop is never cheaper, so it is worth its cost only on questions of medium difficulty, where the agent's iterations do not hit the ceiling but the accuracy is better. Two key remarks to note: (i) token-level F1 can reverse the ranking of setups (the Haiku 4.5 case), therefore a knowledge-grounded judge is a must-have; (ii) over-exploration is a characteristic of only one model (M2.7), not the whole agentic setup.

6.2 RQ1: quality across setups

The agentic loop improves answer quality over one-shot RAG, however the gain is limited to multi-hop questions of the benchmark where retrieval is indeed needed. On MuSiQue, Setup B wins over Setup A on every model in every 2–4-hop bin when evaluated by LLM-as-a-judge, and in three out of four when evaluated by token-level F1. Haiku 4.5 is the exception, and its F1-judge inversion is the reason the judge results are reported. On 1-hop a single retrieval is

enough and the loop is not worth the additional cost. On MultiHop-RAG the models know the answer, therefore retrieval is not needed and Setup 0 is sufficient in most of the cases, depending also on the model (the difference is marginal for Qwen3.5 and GPT-4o mini, in cases of Haiku 4.5 and M2.7 the difference between Setup 0 and B is bigger with B having an advantage).

6.3 RQ2: cost, latency, and interaction depth

The agentic loop is expensive on every axis. On MuSiQue, Setup B consumes from 16 to 35 times more tokens than Setup A, iterates on average 5.94 times per query, and increases the latency per query from less than 1 second (Setup A) to 6–22 seconds (Setup B), depending on the non-reasoning model. M2.7 is the exception on every axis, as its reasoning tokens inflate even the no-retrieval baseline (Setup 0).

6.4 RQ3: when the agent pays off

Because the agent's cost exceeds that of one-shot RAG on every difficulty level, there is no *crossover* point where the agent becomes cheaper. Instead, its quality gain per token cost is the highest on questions of medium difficulty (2–3-hop), and decreases on 4-hop questions, where the 8-step boundary becomes a limitation. Therefore, the agent pays off the most on medium-difficulty questions, not on the simplest, where one-shot RAG is sufficient, and not on the most difficult, where the steps limit it. This evaluation is valid for MuSiQue. On MultiHop-RAG the trade-off does not make sense, as retrieval is not needed.

6.5 RQ4: iteration scaling and over-exploration

The number of iterations scales clearly with the complexity of the question for each of the models. The average number of steps rises from 3.5 on 1-hop to 7.9 on 4-hop, and often even this is not sufficient as the agents hit the step ceiling (67% on average, up to 96% for M2.7). Whether the agent keeps searching for supporting paragraphs after collecting the gold evidence is not a characteristic of the loop but of one model specifically, M2.7. After reaching full recall, three models stop the loop almost immediately and produce the answer (see Section 4). The exception is the reasoning model M2.7, which over-explores systematically, making on average 2.06 extra steps after full recall, in 75% of the runs that were self-terminated. Therefore, the wasted-token concern is justified, but concentrated in one model.

6.6 Further work

In this research we measure when agentic retrieval is justifiable and feasible; the next step is to predict, from the question alone, which setup to use - routing simple 1-hop queries to one-shot RAG, sending moderate 2–3-hop queries to the agent where it is most cost-effective, and flagging the hardest 4-hop queries, where the step cap and retrieval both bind, for a higher budget or a stronger retriever. Across the four models used for this research, one is a reasoning model and three are not. Running all models with both modes would show whether the agent's advantage is smaller when the model reasons on its own. This would, however, require a different choice of models, as the GPT-4o mini used for this research does not support reasoning. We attempted running Qwen3.5 with the reasoning turned on, however it was computationally infeasible on the available endpoint (≈400s/query, ≈weeks for $n = 400$), a faster endpoint would handle this task. Another direction for further work would be to evaluate the models on a domain-knowledge benchmark. In this research MuSiQue and MultiHop-RAG were used, which are based on knowledge from Wikipedia and English news articles [5, 6]. For MuSiQue, there is a chance that the model knows the answer to a question from its training knowledge. MultiHop-RAG was designed to mitigate this risk by using post-cutoff news, however, for the 2025–2026 models evaluated here this is no longer the case, and the leakage is in fact stronger (Section 4). A benchmark with genuinely post-cutoff or counterfactual content remains future work.

7 LIMITATIONS

Naive RAG as the only baseline. Setup A is deliberately the simplest RAG, without reranking, query rewriting, or hybrid retrieval. This was needed to isolate the agent's loop in Setup B, but it does not show how the agent would perform against an enhanced RAG system. Part of the agent's advantage might disappear if the baseline were stronger. This comparison (enhanced RAG vs agent) is studied by Ferrazzi et al. [1], but it is outside the scope of this work.

Agent steps capped. Setup B has a limit of 8 steps, due to token consumption. Averaged across all four models on 4-hop questions this limit is reached in approximately 67% of runs (M2.7: 96%, Qwen3.5: 53%, Haiku 4.5: 77%, GPT-4o mini: 43%), resulting in the model giving a forced answer. A test on 100 4-hop questions (Table B.8) shows that the cap matters only for the reasoning model: GPT-4o mini stays the same (judge 0.285/0.300/0.270 at caps 8/16/32), whereas M2.7 rises monotonically (0.300/0.380/0.475), at 8.6 times more tokens. The reported 4-hop score for Setup B for M2.7 is therefore a conservative lower bound. Its true ceiling is limited by the model's 64k context window, not the step cap.

Noise in gold answers. Some of the reference answers in MuSiQue are misaligned or phrased differently from the model's answer, so F1 is often lower than the actual correctness. A manual review of 48 (12 per model) low-F1 (<0.3) cases showed that 12 (25%), of which 3 were gold-answer errors and 9 format/alias mismatches, were not genuine model errors. The absolute F1 values should therefore be read as a conservative, lower bound estimate on accuracy. The LLM-as-a-judge metric largely addresses this by assessing correctness grounded in the supporting evidence rather than by literal match. On the same 48 cases, the judge's verdicts agreed with human judgement in 43 (89.6%) cases, which led to recovery of most answers that token-F1 had wrongly penalized.

Judge model capability. The LLM-as-a-judge uses DeepSeek-V4-Flash, which is a smaller and generally weaker model than the four system models it evaluates. While the choice of a different model family reduces self-preference bias [10], a weaker judge may fail to recognize correct answers that require stronger reasoning to assess.

ACKNOWLEDGMENTS

I sincerely thank my supervisor, Dr. Nacir Bouali, for his guidance and constructive feedback throughout this research.

REFERENCES

[1] Pietro Ferrazzi, Milica Cvjetićanin, Alessio Piraccini, and Davide Giannuzzi. 2026. Is Agentic RAG worth it? An experimental comparison of RAG approaches. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, Yunyao Li, Georg Rehm, and Mei Tu (Eds.). Association for Computational Linguistics, San Diego, California, USA, 56–75. https://aclanthology.org/2026.acl-industry.5/

[2] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Trans. Inf. Syst.* 43, 2, Article 42 (Jan. 2025), 55 pages. https://doi.org/10.1145/3703155

[3] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 9459–9474. https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf

[4] Fnu Neha and Deepshikha Bhati. 2025. Traditional RAG vs. Agentic RAG: A Comparative Study of Retrieval-Augmented Systems. In *2025 IEEE International Conference on Future Machine Learning and Data Science (FMLDS)*. 383–387. https://doi.org/10.1109/FMLDS67896.2025.00073

[5] Yixuan Tang and Yi Yang. 2024. MultiHop-RAG: Benchmarking Retrieval-Augmented Generation for Multi-Hop Queries. In *First Conference on Language Modeling*. https://openreview.net/forum?id=t4eB3zYWBK

[6] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554. https://doi.org/10.1162/tacl_a_00475

[7] 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)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 10014–10037. https://doi.org/10.18653/v1/2023.acl-long.557

[8] 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*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 2369–2380. https://doi.org/10.18653/v1/D18-1259

[9] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

[10] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 46595–46623. https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf

A AI DISCLOSURE

During the preparation of this work, I used Claude by Anthropic as a supplementary tool to support grammatical editing, improve clarity, adjust academic tone and assist with writing and refining Python code. Claude was not used to generate the substantive arguments, analysis, interpretation, or conclusions presented in this work. All AI-generated suggestions were critically reviewed, selectively incorporated, and edited by me. I take full responsibility for the accuracy, originality, and final content of this work.

B FULL PER-HOP RESULTS FROM MUSIQUE

Table B.1. Answer quality (token-level F1) by model, setup, and hop count. Bold marks the best setup per model and bin.

Model	Setup	1-hop	2-hop	3-hop	4-hop	All
MiniMax M2.7	0	0.386	0.220	0.140	0.066	0.203
	A	**0.701**	0.168	0.201	0.029	0.275
	B	0.637	**0.468**	**0.429**	**0.192**	**0.432**
Qwen3.5	0	0.347	0.158	0.150	0.093	0.187
	A	0.721	0.207	0.287	0.133	0.337
	B	**0.723**	**0.549**	**0.526**	**0.217**	**0.504**
Claude Haiku 4.5	0	0.411	**0.197**	0.127	0.051	0.196
	A	**0.688**	**0.197**	0.087	0.018	**0.248**
	B	0.402	0.161	**0.184**	**0.119**	0.216
GPT-4o mini	0	0.354	0.165	0.199	0.099	0.204
	A	0.669	0.193	0.191	0.090	0.286
	B	**0.703**	**0.460**	**0.444**	**0.232**	**0.460**

Table B.2. Answer quality under the grounded LLM-as-a-judge ($CORRECT = 1$, $PARTIAL = 0.5$, $INCORRECT = 0$), by model, setup, and hop count. Bold marks the best setup per model and bin.

Model	Setup	1-hop	2-hop	3-hop	4-hop	All
MiniMax M2.7	0	0.495	0.225	0.185	0.055	0.240
	A	0.790	0.225	0.215	0.060	0.323
	B	**0.865**	**0.610**	**0.535**	**0.300**	**0.578**
Qwen3.5	0	0.465	0.160	0.145	0.115	0.221
	A	0.800	0.230	0.295	0.130	0.364
	B	**0.830**	**0.605**	**0.550**	**0.250**	**0.559**
Claude Haiku 4.5	0	0.480	0.195	0.170	0.115	0.240
	A	0.760	0.230	0.085	0.020	0.274
	B	**0.865**	**0.565**	**0.600**	**0.325**	**0.589**
GPT-4o mini	0	0.435	0.135	0.200	0.115	0.221
	A	0.755	0.210	0.195	0.070	0.307
	B	**0.845**	**0.555**	**0.525**	**0.285**	**0.552**

Table B.3. Exact match (EM) by model, setup, and hop count. Bold marks the best setup per model and bin.

Model	Setup	1-hop	2-hop	3-hop	4-hop	All
MiniMax M2.7	0	0.350	0.130	0.080	0.040	0.150
	A	**0.660**	0.130	0.140	0.010	0.235
	B	0.560	**0.360**	**0.330**	**0.140**	**0.347**
Qwen3.5	0	0.300	0.050	0.040	0.020	0.102
	A	**0.640**	0.130	0.170	0.090	0.258
	B	0.630	**0.460**	**0.410**	**0.140**	**0.410**
Claude Haiku 4.5	0	0.360	0.100	0.060	0.020	0.135
	A	**0.600**	**0.140**	0.040	0.000	**0.195**
	B	0.320	0.070	**0.110**	**0.080**	0.145
GPT-4o mini	0	0.300	0.070	0.080	0.030	0.120
	A	**0.590**	0.130	0.110	0.050	0.220
	B	0.570	**0.310**	**0.300**	**0.170**	**0.338**

Table B.4. Mean total tokens (input + output) per query by model, setup, and hop count. The B/A row gives the agent's token cost as a multiple of one-shot RAG (ALL hops).

Model	Setup	1-hop	2-hop	3-hop	4-hop	All
MiniMax M2.7	0	500	874	3261	5684	2580
	A	632	783	1099	1459	993
	B	16350	27947	35900	42038	30559
	B/A					**30.8×**
Qwen3.5	0	83	91	99	103	94
	A	487	506	503	519	504
	B	3041	8122	9501	15594	9064
	B/A					**18.0×**
Claude Haiku 4.5	0	83	109	174	239	151
	A	509	545	563	586	551
	B	5283	15778	22487	33043	19148
	B/A					**34.8×**
GPT-4o mini	0	75	81	88	92	84
	A	454	462	462	480	464
	B	4086	6675	7928	11517	7551
	B/A					**16.3×**

Table B.5. Mean latency (seconds) per query by model, setup, and hop count.

Model	Setup	1-hop	2-hop	3-hop	4-hop	All
MiniMax M2.7	0	17.8	18.6	78.1	266.8	95.3
	A	11.0	17.5	36.7	63.9	32.3
	B	13.0	23.3	30.7	40.2	26.8
Qwen3.5	0	0.3	0.3	0.3	0.3	0.3
	A	0.4	0.4	0.4	0.4	0.4
	B	3.8	8.2	10.1	14.3	9.1
Claude Haiku 4.5	0	0.6	0.8	1.3	1.9	1.2
	A	0.7	0.9	1.0	1.1	0.9
	B	5.8	17.7	26.2	37.9	21.9
GPT-4o mini	0	0.7	0.7	0.6	0.8	0.7
	A	0.6	0.6	0.7	0.7	0.7
	B	3.4	5.9	6.2	9.7	6.3

Table B.6. Supporting-paragraph recall of single-shot retrieval as a function of the window size k ($n = 150$ MuSiQue questions). Recall rises slowly with k and never closes the multi-hop gap: 4-hop full-recall is 0% at $k = 3$ and only 6% at $k = 50$.

k	2-hop	3-hop	4-hop
3	0.530	0.293	0.205
5	0.580	0.373	0.235
10	0.640	0.453	0.280
20	0.660	0.540	0.335
30	0.670	0.593	0.395
50	0.710	0.667	0.435

Table B.7. Approximate per-query API cost (USD) on MuSiQue for the two commercially accessed models. Rates current during the experiments: Claude Haiku 4.5 at $1.00 input / $5.00 output per million tokens; GPT-4o mini at $0.15 input / $0.60 output per million tokens. Qwen3.5 and MiniMax M2.7 ran on an institutional endpoint and are not included.

Model	Setup 0	Setup A	Setup B
GPT-4o mini	$0.0000143	$0.00007	$0.00117
Claude Haiku 4.5	$0.00043	$0.00072	$0.02150
Ratio (Haiku/GPT)	30×	10×	18×

Table B.8. Step-cap test on the 4-hop bin of MuSiQue ($n = 100$). M2.7 shows a strong monotonic gain with relaxed cap; GPT-4o mini is unaffected. Cap=32 for MiniMax M2.7 is bounded from above by its 64k context window rather than the step cap itself. "Mean calls" includes the forced final-answer call, so it can exceed the cap.

Model	Cap	Judge	Mean calls	Cap-hit %	Tokens (M)
MiniMax M2.7	8	0.300	8.84	96%	4.2
	16	0.380	15.77	73%	13.8
	32	0.475	23.87	32%	36.1
GPT-4o mini	8	0.285	6.80	43%	1.2
	16	0.300	9.54	34%	2.7
	32	0.270	16.13	37%	8.8

Table B.9. Paired bootstrap 95% confidence intervals on the mean Setup B − Setup A difference on MuSiQue (10,000 resamples, seed 20260625; positive favors the agent). A CI excluding zero is significant at $p < 0.05$ (sig); otherwise not significant (ns).

		Judge $B - A$		F1 $B - A$	
Model	Hop	Mean	95% CI	Mean	95% CI
MiniMax M2.7	1	+0.075	[−0.010, +0.160] ns	−0.064	[−0.156, +0.030] ns
	2	+0.385	[+0.275, +0.490]	+0.300	[+0.215, +0.388]
	3	+0.320	[+0.210, +0.430]	+0.228	[+0.128, +0.332]
	4	+0.240	[+0.150, +0.330]	+0.163	[+0.092, +0.238]
	All	+0.255	[+0.205, +0.306]	+0.157	[+0.111, +0.204]
Qwen3.5	1	+0.030	[−0.040, +0.105] ns	+0.003	[−0.074, +0.080] ns
	2	+0.375	[+0.275, +0.475]	+0.342	[+0.248, +0.436]
	3	+0.255	[+0.150, +0.360]	+0.239	[+0.138, +0.340]
	4	+0.120	[+0.025, +0.215]	+0.084	[+0.005, +0.164]
	All	+0.195	[+0.148, +0.245]	+0.167	[+0.122, +0.214]
Claude Haiku 4.5	1	+0.105	[+0.035, +0.180]	−0.286	[−0.376, −0.198]
	2	+0.335	[+0.240, +0.435]	−0.037	[−0.110, +0.034] ns
	3	+0.515	[+0.410, +0.615]	+0.097	[+0.025, +0.171]
	4	+0.305	[+0.220, +0.395]	+0.100	[+0.051, +0.156]
	All	+0.315	[+0.269, +0.364]	−0.031	[−0.071, +0.009] ns
GPT-4o mini	1	+0.090	[+0.015, +0.165]	+0.034	[−0.046, +0.115] ns
	2	+0.345	[+0.235, +0.455]	+0.268	[+0.178, +0.360]
	3	+0.330	[+0.220, +0.435]	+0.253	[+0.156, +0.349]
	4	+0.215	[+0.130, +0.305]	+0.142	[+0.067, +0.222]
	All	+0.245	[+0.196, +0.294]	+0.174	[+0.130, +0.218]

C FULL PER-DOC RESULTS FROM MULTIHOP-RAG

The complexity axis on MultiHop-RAG is the number of evidence documents assigned to each question (2, 3, or 4), not a hop count; see Section 3 for the distinction.

Table C.10. Answer quality (token-level F1) on MultiHop-RAG by model, setup, and evidence-document count. Bold marks the best setup per model and bin.

Model	Setup	2-doc	3-doc	4-doc	All
MiniMax M2.7	0	0.241	0.522	0.746	0.503
	A	0.190	0.267	0.550	0.336
	B	**0.286**	**0.577**	**0.975**	**0.613**
Qwen3.5	0	**0.745**	0.785	0.920	0.817
	A	0.250	0.417	0.840	0.502
	B	0.692	**0.861**	**0.978**	**0.844**
Claude Haiku 4.5	0	**0.181**	**0.490**	**0.953**	**0.541**
	A	0.052	0.292	0.714	0.352
	B	0.093	0.187	0.226	0.169
GPT-4o mini	0	**0.587**	**0.685**	0.970	**0.747**
	A	0.284	0.473	0.943	0.567
	B	0.292	0.502	**0.973**	0.589

Table C.11. Answer quality under the grounded LLM-as-a-judge (*CORRECT* = 1, *PARTIAL* = 0.5, *INCORRECT* = 0) on MultiHop-RAG by model, setup, and evidence-document count. Bold marks the best setup per model and bin.

Model	Setup	2-doc	3-doc	4-doc	All
MiniMax M2.7	0	0.260	0.540	0.740	0.513
	A	0.190	0.280	0.550	0.340
	B	**0.640**	**0.745**	**0.990**	**0.792**
Qwen3.5	0	**0.750**	0.795	0.925	0.823
	A	0.270	0.430	0.840	0.513
	B	0.720	**0.885**	**0.970**	**0.858**
Claude Haiku 4.5	0	0.180	0.510	0.950	0.547
	A	0.050	0.300	0.710	0.353
	B	**0.595**	**0.705**	**0.960**	**0.753**
GPT-4o mini	0	0.580	0.720	0.970	0.757
	A	0.280	0.490	0.940	0.570
	B	**0.660**	**0.725**	**0.975**	**0.787**

D ADDITIONAL PLOTS

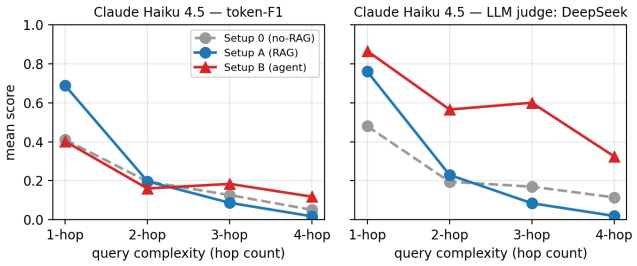

Fig. D.1. Token-level F1 (left) and LLM-as-a-judge score (right) for Claude Haiku 4.5 by setup and hop count on MuSiQue. Setup B ranks last under F1 overall but first under the judge. This motivates reporting both an overlap metric and an evidence-grounded judge.

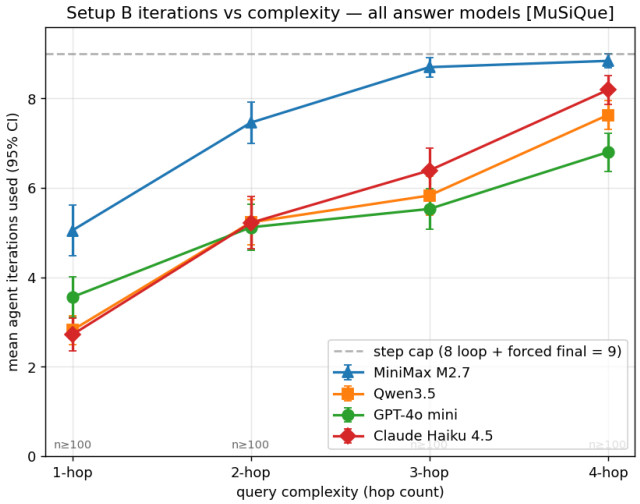

Fig. D.2. Tool-calling iterations used by the Setup B agent per query complexity. The loop is capped at 8 iterations; the agent then generates the final answer with one additional model call. M2.7 almost always hits the ceiling on 3- and 4-hop queries, consistent with its reasoning nature.

E PROMPTS

E.1 Setup 0 — No-retrieval baseline

User prompt and system message. Setup 0 has no abstention instruction: the model is allowed to guess, so F1/EM captures parametric leakage. In case of MHR questions, the persona was changed slightly, by adding *...about recent news.*

```
You are a research assistant answering a multi-hop question.
Answer the question with the shortest possible answer
(a name, date, number, or short phrase). Do not explain.
Do not provide context. Output only the answer itself.

Question: {query}
Answer:
```

E.2 Setup A — One-shot RAG

User prompt and system message. `{context}` is the top-3 retrieved chunks, each prefixed `[1]`, `[2]`, `[3]`. In case of MHR questions, the persona was changed slightly, by adding *...about recent news.*

```
You are a research assistant answering a multi-hop question.
Answer the question using only the context below. If the
context does not contain the answer, say you don't know.
Give the shortest possible answer (a name, date, number,
or short phrase). Do not explain. Output only the answer
itself.

Context:
{context}

Question: {query}
Answer:
```

E.3 Setup B — Agent (ReAct loop)

Two execution paths: native tool-calling (M2.7, GPT-4o mini, Haiku 4.5) and a prompted `TOOL_CALL:` fallback (Qwen3.5, whose endpoint lacks native tool-calling support). The structure is identical across paths and benchmarks; only the corpus noun differs.

System prompt — native path (MuSiQue).

```
You are a research assistant answering a multi-hop question.
You have two tools: semantic_search to find relevant
paragraphs by meaning, and list_files to browse paragraph
titles by a filter substring. Work step by step: find one
fact, then use it to search for the next. When you have
gathered enough information, stop calling tools and reply
with ONLY the final short answer (a name, date, number, or
short phrase). Do not explain and do not add context.
```

System prompt — native path (MultiHop-RAG).. Identical, with "passages from news articles" and "browse article titles":

```
You are a research assistant answering a multi-hop question
about recent news. You have two tools: semantic_search to
find relevant passages from news articles by meaning, and
list_files to browse article titles by a filter substring.
Work step by step: find one fact, then use it to search for
the next. When you have gathered enough information, stop
calling tools and reply with ONLY the final short answer
(a name, date, number, or short phrase). Do not explain and
do not add context.
```

System prompt — prompted path (MuSiQue).

```
You are a research assistant answering a multi-hop question.
You have two tools:
- semantic_search: find relevant paragraphs by meaning.
  arguments: {"query": string, "k": integer (optional)}
- list_files: browse paragraph titles by a filter substring.
  arguments: {"filter": string}

To use a tool, reply with exactly one line and nothing else:
TOOL_CALL: {"name": "<tool>", "arguments": {...}}

After you receive the tool result, you may call another tool
or answer. Work step by step: find one fact, then use it to
search for the next. When you have gathered enough
information, reply with ONLY the final short answer (a name,
date, number, or short phrase). Do not explain, do not add
context, and do not write TOOL_CALL on that final turn.
```

The MultiHop-RAG prompted variant is identical, with "passages from news articles" and "article titles".

Other messages. Initial user message (both paths):

```
Question: {question}
```

Tool-result feedback (prompted path; native path uses a `role:"tool"` message instead):

```
TOOL_RESULT:
{formatted_result}
```

Step-bound forcing message, sent on reaching $MAX_STEPS = 8$:

```
You have used all your search steps. Based on what you
found, reply now with ONLY the final short answer.
```

Tool descriptions (native path).
`semantic_search`:

```
Search the document index for paragraphs relevant to a
natural-language query. Returns a list of paragraphs sorted
by semantic similarity (most similar first). Use this to
find information by meaning. Search step by step: find one
fact, then use it to search for the next.
  params: query (string, required)
          k (integer, defaults to 3 if omitted)
```

`list_files`:

```
List the ids and titles of paragraphs available in the
index. The index is large, so always pass 'filter': only
entries whose id or title contain that substring
(case-insensitive) are returned. Use this to browse what
is available before searching.
  param: filter (string, optional)
```

E.4 LLM-as-a-judge

Single user prompt, no system message. `{ref}` appends aliases ("also acceptable: …") when present. Empty predictions are scored INCORRECT deterministically without a model call; MultiHop-RAG null queries are skipped and scored by the abstention metric instead.

```
You are grading an answer to a multi-hop question. You are
given supporting evidence that contains the facts needed to
answer. Judge the predicted answer ONLY against this evidence
and the reference answer. Do not use your own outside
knowledge.

Question: {question}

Supporting evidence:
{evidence}

Predicted answer: {predicted}
Reference (gold) answer: {ref}

Decide whether the predicted answer is:
- CORRECT: it conveys the same fact as the reference answer.
  Different wording, formatting, or extra correct detail is
  fine.
- PARTIAL: it captures part of the required answer but is
  incomplete, or names a specific instance of what the
  reference describes.
- INCORRECT: it is wrong, contradicts the evidence, or fails
  to answer.

Respond in exactly this format:
VERDICT: <CORRECT|PARTIAL|INCORRECT>
JUSTIFICATION: <one sentence>
```