The Impact of Query Decomposition and Cross-Encoder Reranking in Multi-Hop Retrieval-Augmented Generation

Hail Lim, Michigan State University, 132 Crossroads Ln., Troy, Michigan, 48083, United States

Corresponding author email: ianlim.dev@gmail.com

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a promising paradigm for open-domain question answering. However, standard single-hop retrieval often fails on complex, multi-hop queries where the answer requires synthesizing information from disparate documents. In this work, we propose an enhanced Multi-Hop RAG pipeline augmented with Cross-Encoder Reranking to address the challenges of reasoning across multiple documents. Our approach decomposes complex queries into self-contained sub-questions and employs a Cross-Encoder to rerank candidates at each retrieval step, mitigating the "semantic drift" inherent in dense vector search. We systematically evaluate our system against two baselines—Standard Single-Hop RAG and Decomposed Multi-Hop RAG—using a curated subset of the HotpotQA dataset. Experimental results demonstrate that our proposed method achieves superior accuracy (62%, a 20% gain over the single-hop baseline) by effectively filtering distractors. Furthermore, our ablation studies reveal a fundamental "Recall Ceiling" in dense retrieval, where blindly increasing the candidate pool yields diminishing returns. Based on these findings, we identify a "Wide Net, Tight Filter" strategy as the Pareto-optimal configuration for balancing reasoning accuracy with system latency.

Keywords

Large language models (LLM); Retrieval-augmented generation (RAG); Multi-hop question; Query decomposition; Reranking; HotpotQA.

Introduction

Retrieval-Augmented Generation (RAG) [1,2,3] has become the de facto standard for equipping Large Language Models (LLMs) with up-to-date, non-parametric knowledge. While standard RAG pipelines excel at "Single-Hop" queries—where the answer resides in a semantically similar passage—they often falter in realistic, complex scenarios. Many information-seeking tasks are inherently Multi-Hop [4,5,6], requiring the synthesis of facts dispersed across disparate documents connected by implicit reasoning chains rather than explicit keyword overlap [7,8,9,10]. In such cases, the semantic vector of the original question aligns poorly with the document containing the final answer, leading to "semantic drift".

Despite the proliferation of RAG frameworks, architectural best practices for these complex queries remain under-documented. Practitioners face a dilemma: Standard dense retrieval (Bi-Encoder) is efficient but lacks the reasoning depth to bridge semantic gaps; conversely, complex "Agentic" workflows involving iterative decomposition or heavy cross-encoder reranking promise higher accuracy but introduce significant latency [11,12,13,14]. Furthermore, simply retrieving more documents often yields diminishing returns due to the "Recall Ceiling" of dense embeddings—a fundamental limitation frequently overlooked in favor of generative optimizations [15,16,17,18].

Therefore, bridging the gap between naive RAG and robust multi-hop reasoning requires a rigorous empirical understanding of design trade-offs. Specifically: *When does the cost of query decomposition pay off? Is the precision gain from a cross-encoder worth the latency overhead? And at what point does retrieval recall become the insurmountable bottleneck?*

In this work, we study two such design decisions that are particularly salient in practice:

1. **Should we explicitly decompose complex questions?** Prompting LLMs to decompose questions into simpler sub-questions is theoretically sound but introduces sequential dependencies. It is not obvious under which conditions this complexity outperforms direct retrieval.
2. **Should we add a cross-encoder reranking stage?** Cross-encoders capture subtle logical relations by jointly encoding query-document pairs [], unlike interaction-light Bi-Encoders. For multi-hop RAG, this raises a critical trade-off between answer quality and system latency, especially when reranking is performed iteratively.

Our goal is to make these trade-offs concrete and measurable in a realistic, controlled setting. We build a modular RAG pipeline on top of **HotpotQA** [4] and systematically compare three baselines:

Baseline A: Single-hop RAG. Retrieves top-k passages for the original question.
Baseline B: Multi-hop RAG with Decomposition. Decomposes the question and retrieves evidence sequentially.
Baseline C: Multi-hop RAG with Cross-Encoder Reranking. Adds a reranking stage to filter distractors from a larger candidate pool.

Concretely, we investigate the efficacy and limitations of these architectures by addressing three interconnected dimensions. First, we examine Decomposition Effectiveness, quantifying how well decomposition mitigates semantic drift. Second, we analyze the Cost-Benefit Ratio of Reranking, determining if precision gains justify the computational latency. Finally, we explore System Boundaries, specifically characterizing the "Recall Ceiling"—the saturation point where increasing retrieval hyperparameters no longer yields accuracy improvements.

In summary, the contribution of this paper are three-fold:

1. We design and open-source a modular Multi-Hop RAG pipeline that cleanly separates indexing, decomposition, retrieval, and reranking.
2. We perform a systematic evaluation, revealing a performance hierarchy where our Reranking proposal achieves a 20% accuracy gain over the single-hop baseline.
3. We provide actionable engineering guidelines, identifying a "Wide Net, Tight Filter" strategy (large number pre-fetch candidates, small number of retrievals) as the Pareto-optimal configuration, while characterizing the fundamental "Recall Ceiling" on hard queries.

Background & Related Work

Large language models (LLMs) are rapidly transforming the landscape of artificial intelligence. With their expanding capabilities in language understanding and generation, LLMs have been widely adopted in knowledge-intensive scenarios, including everyday question answering, engineering workflows, and scientific research assistants. [illegible] However, even very large models have a fixed training cutoff and may lack access to up-to-date or domain-specific information, and it is often impractical or impossible to continually retrain them on all relevant data sources. This has created a growing demand for techniques that integrate LLMs with external knowledge in a scalable and controllable way.

1. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) has emerged as a standard solution to this challenge [21, 22, 23, 24, 25]. In RAG, a parametric language model is coupled with a non-parametric memory, typically a dense index over a text corpus, and generation is conditioned on documents retrieved from that index. Lewis et al [1]. introduced RAG as a general-purpose recipe for knowledge-intensive NLP tasks, showing that augmenting a seq2seq model with a learned neural retriever over Wikipedia improves performance and reduces hallucination compared to purely parametric models. Subsequent work and system reports have documented wide adoption of RAG in industrial question answering, document assistants, and domain-specific copilots. Conceptually, many early RAG systems follow a two-stage workflow very similar to the one we adopt in this paper:

1. A retrieval stage, which gathers context-relevant passages from an external knowledge base;
2. A generation stage, which conditions the LLM on the retrieved passages (e.g., by concatenating them into the prompt) to produce answers that are more accurate and grounded.

This architecture enables models to access information beyond their original training data without incurring the prohibitive cost of full retraining and allows practitioners to update knowledge by re-indexing documents rather than modifying model weights. Empirically, retrieval-augmented systems have been shown to reduce hallucinations by grounding responses in explicit evidence and to support stricter data-governance regimes, since sensitive documents can be controlled at the retrieval layer rather than embedded directly in model parameters.

2. Multi-Hop Question Answering and HotpotQA

While RAG has become the default for single-hop or factoid QA, multi-hop questions pose additional challenges. In multi-hop QA [4,5,6,27,28], a system must gather and integrate information from multiple documents to answer a question; solving the task requires both accurate retrieval and explicit or implicit reasoning across pieces of evidence.

HotpotQA [4] is a widely used benchmark for this setting. Yang et al. [4] introduced HotpotQA as a dataset of 113k Wikipedia-based questions requiring reasoning over multiple supporting documents, along with sentence-level supporting facts for strong supervision and explainable predictions. The dataset includes “comparison” questions and distractor paragraphs, making it a realistic and challenging environment for multi-hop retrieval and reasoning. HotpotQA has spurred a line of work on pipeline architectures that combine document selection, sentence selection, and reading comprehension models, as well as graph-based and neural-reasoning approaches to multi-document QA. In the era of LLMs and RAG, HotpotQA remains a natural testbed for evaluating whether retrieval-augmented LLMs can perform explicit multi-hop reasoning when provided with a large but noisy evidence pool.

3. LLM-Based Decomposition and Multi-Step Reasoning

The emergence of instruction-tuned LLMs has enabled new approaches to multi-step reasoning that rely heavily on prompting [7,8,9]. Chain-of-Thought (CoT) [10] prompting demonstrates that providing or eliciting intermediate reasoning steps can significantly improve performance on arithmetic, common sense, and symbolic reasoning tasks [23]. More broadly, recent work on tool-using agents and planner–executor architectures shows that LLMs can be prompted to decompose complex tasks into sub-tasks and call external tools such as search APIs in a loop, effectively implementing a form of decompose–solve–compose reasoning.

In the context of multi-hop QA and RAG, these ideas have inspired methods that decompose a complex question into a sequence of sub-questions, retrieve and answer each sub-question separately, and then synthesize a final answer from the intermediate results. Such decomposition is attractive because it (i) makes retrieval queries more focused, potentially improving recall, and (ii) yields interpretable intermediate steps that can be inspected or debugged. However, decomposition also introduces new failure modes: if

the decomposition is incorrect or misaligned with the underlying reasoning structure, downstream retrieval and answering may be led astray.

Our multi-hop baselines are directly motivated by this line of work. We use an LLM to generate a sequence of sub-questions from the original query, then apply RAG per sub-question, and finally synthesize a final answer. This allows us to empirically study when decomposition improves performance over a single-hop RAG baseline on HotpotQA-style questions.

Methods

In this study, we developed and evaluated three distinct RAG pipelines to investigate the efficacy of multi-hop reasoning and cross-encoder reranking in answering complex, bridge-type questions. All pipelines were implemented using the LangChain framework, utilizing OpenAI's gpt-3.5-turbo as the generator and text-embedding-ada-002 for dense retrieval. The document corpus consists of paragraphs from the HotpotQA dataset, indexed using a FAISS vector store [26].

1. Baseline 1: Single-Hop RAG (Standard Baseline)

The Single-Hop RAG serves as the naive baseline, representing standard industry practices where retrieval is performed once based on the original user query. Given a complex user query Q, this pipeline performs a direct dense retrieval against the vector index V. The system computes the cosine similarity between the query embedding $E(Q)$ and document embeddings $E(d_i)$ for all documents in the corpus. The top-k documents with the highest similarity scores are retrieved:

$$\mathcal{D}_{retrieved} = \text{Top-}k\left(\cos(E(Q), E(d_i))\right)$$

In our experiments, we set $k = 5$. These retrieved documents are concatenated into a context string C and passed to the LLM along with the original query to generate the final answer A.

Limitation. This baseline assumes that the answer to a complex question resides within a single retrieval step's semantic vicinity. However, for multi-hop questions (e.g., "Who is the director of the film starring X?"), the necessary information for the second hop ("film starring X") may not share high semantic overlap with the original query, often leading to retrieval failure.

2. Baseline 2: Multi-Hop RAG via Query Decomposition

To address the limitations of single-hop retrieval, we implemented a Multi-Hop RAG pipeline that mimics the human reasoning process by decomposing complex questions into a sequence of simpler, self-contained sub-questions. We explain the workflow below.

Query Decomposition Mechanism. We employ a "Decompose-then-Execute" strategy. The core component is a decomposition chain driven by a specifically engineered prompt. The prompt instructs the LLM to break down the input question Q into a list of sub-questions $\{q_1, \ldots, q_n\}$. Crucially, to prevent information loss during independent retrieval steps, we enforce a Context Retention Constraint in our decomposition prompt. This ensures that sub-questions are self-contained and do not rely on vague pronouns (e.g., resolving "When was he born?" to "When was the creator of The Simpsons born?"). The decomposition prompt template is defined as follows:

You are an expert at query decomposition for retrieval systems. Break down the complex question into a series of simple sub-questions. IMPORTANT: Each sub-question must be SELF-CONTAINED. Do NOT use vague pronouns without referring to the original context.

Iterative Retrieval and Synthesis. The pipeline executes sequentially. *(1) Decomposition:* The LLM generates the list of sub-questions. *(2) Sequential Retrieval:* For each sub-question q_j, the system performs

an independent dense retrieval to obtain a specific context set D_j. *(3) Synthesis:* All retrieved contexts from all steps are aggregated. A final synthesis prompt guides the LLM to reason across these disjoint pieces of information to construct the final answer.

3. Proposal (Baseline 3): Multi-Hop RAG with Cross-Encoder Reranking

While Baseline 2 improves reasoning, dense retrieval (Bi-Encoder) often suffers from "semantic drift," retrieving documents that are semantically similar but factually irrelevant (distractors). To mitigate this, we propose our primary method: a Multi-Hop pipeline augmented with a Cross-Encoder Reranker.

Reranking Mathematical Formulation. In the standard retrieval stage (Bi-Encoder), the relevance score s_{bi} is the dot product of independent vectors: $s_{bi}(q,d) = \varphi(q)^T\psi(d)$, where and ψ are separate encoding networks (or the same network). This is computationally efficient but lacks deep interaction between query and document tokens. In our proposal, we introduce a re-ranking stage using a Cross-Encoder model M_{cross}, specifically ms-marco-MiniLM-L-6-v2. Unlike the Bi-Encoder, the Cross-Encoder takes the concatenated pair of the query and document as a single input, allowing full self-attention across both sequences:

where *CLS* is the classification token output used to predict the relevance probability.

$$s_{cross}(q,d) = \text{Sigmoid}\left(\mathcal{M}_{cross}([CLS] \circ q \circ [SEP] \circ d)\right)$$

Pipeline Implementation. The enhanced retrieval process for each sub-question q_j operates as follows. *(1) Recall Expansion:* We first retrieve a larger set of candidates C (where $|C| = K_{candidates} = 15$) using the standard vector search. *(2) Scoring:* Each candidate document $d \in C$ is paired with the sub-question q_j and scored using the Cross-Encoder to obtain $s_{cross}(q_j, d)$. *(3) Reranking & Pruning:* Documents are sorted by s_{cross} in descending order. We retain the top-K' documents (where $K' = 7$) to form the final refined

$$\mathcal{D}_{final} = \{d \in \mathcal{C} \mid \text{Rank}(s_{cross}(q_j, d)) \leq K'\}$$

context.

This reranking step acts as a semantic filter, effectively removing high-similarity distractors (e.g., documents sharing keywords but different entities) and ensuring that the subsequent generation step is grounded in high-precision evidence.

Experiment Setup

To rigorously evaluate the proposed Multi-Hop RAG with Reranking against standard baselines, we designed a comprehensive experimental workflow encompassing knowledge base construction, automated test set curation, and LLM-based evaluation. The implementation details are available in our open-source repository, specifically within *config.py* and *rag_engine.py*.

1. Dataset and Knowledge Base Construction. We utilized the **HotpotQA** dataset (distractor setting), a benchmark specifically designed for multi-hop reasoning. To simulate a realistic, resource-constrained RAG environment, we constructed a local knowledge base using a dense subset of the training split (e.g. train[:1000]). The document processing pipeline (*data_loader.py* and *vector_store.py*) involved the following steps:

(1) *Chunking:* Raw documents were split into chunks of 256 tokens with a 100-token overlap to preserve contextual continuity across boundaries.
(2) *Indexing:* We generated dense vector embeddings using OpenAI's `text-embedding-ada-002`. These embeddings were indexed using FAISS (Facebook AI Similarity Search) to enable efficient similarity retrieval.

This process will output files *index.faiss* and *index.pkl* for future reference so that we do not need to generate them in each experiment.

2. Automated Test Set Curation. To ensure the evaluation focused on reasoning capabilities rather than simple keyword matching, we did not random sample questions. Instead, we implemented a targeted curation script (*generate_test_set.py*). We filtered the dataset to select questions meeting two specific criteria: *(1) Type Constraint* (type="bridge"). This ensures the question requires connecting multiple distinct pieces of information (e.g., finding a director, then finding their birth year). (2) *Difficulty Constraint:* (level="medium"). We selected medium-difficulty questions to balance the need for multi-hop reasoning while avoiding obscure entities that might suffer from embedding recall failures in a small index. This process yielded a curated test set where the ground truth answers are guaranteed to be present in the indexed corpus, isolating the retrieval and reasoning performance of the models.

3. Experimental Baselines. We compared three distinct RAG configurations using the same underlying Generator (`gpt-3.5-turbo`) and Vector Store. The construction of these baselines are elaborated in the Methods section. Here we focus on the setup parameters.

Baseline 1: Single-Hop RAG. A naive approach performing a single vector search ($K = 8$) followed by answer generation. This serves as the lower bound for performance.

Baseline 2: Multi-Hop RAG. A decompose-then-execute pipeline where the complex query is broken into sub-questions. Each sub-question triggers a standard vector search.

Baseline 3 (Proposal): Multi-Hop RAG with Reranking. It augments the multi-hop pipeline with a “Recall-then-Precision” strategy: *(1) Recall.* It fetches a larger pool of candidates ($K_{candidate} = 30$). *(2) Rerank:* A Cross-Encoder (`ms-marco-MiniLM-L-12-v2`) scores the relevance of each candidate. *(3) Safety Fusion:* To prevent over-pruning, we force the inclusion of the top-1 vector search result alongside the highest-scoring reranked documents before generation.

4. Evaluation Protocol: LLM-as-a-Judge

Given the generative nature of the answers, exact string matching is insufficient for accuracy calculation. We implemented an automated evaluation pipeline (*evaluator.py*) utilizing **LLM-as-a-Judge** [19,20]. To ensure reliability and mitigate the "lazy judge" phenomenon often seen in smaller models, we employed GPT-4o as the evaluator. The judge operates with a Chain-of-Thought prompt that enforces a strict verification protocol that extracts key entities (names, dates, locations) from the Ground Truth, verifies their presence in the Generated Answer, and penalizes vague answers or hallucinations.

The system assigns a binary score (Correct/Incorrect) to each output. Final accuracy is reported as the percentage of correct answers across the test set. All experimental parameters, including model names and retrieval hyperparameters, are logged automatically to *config.py* to ensure reproducibility.

Results & Discussion

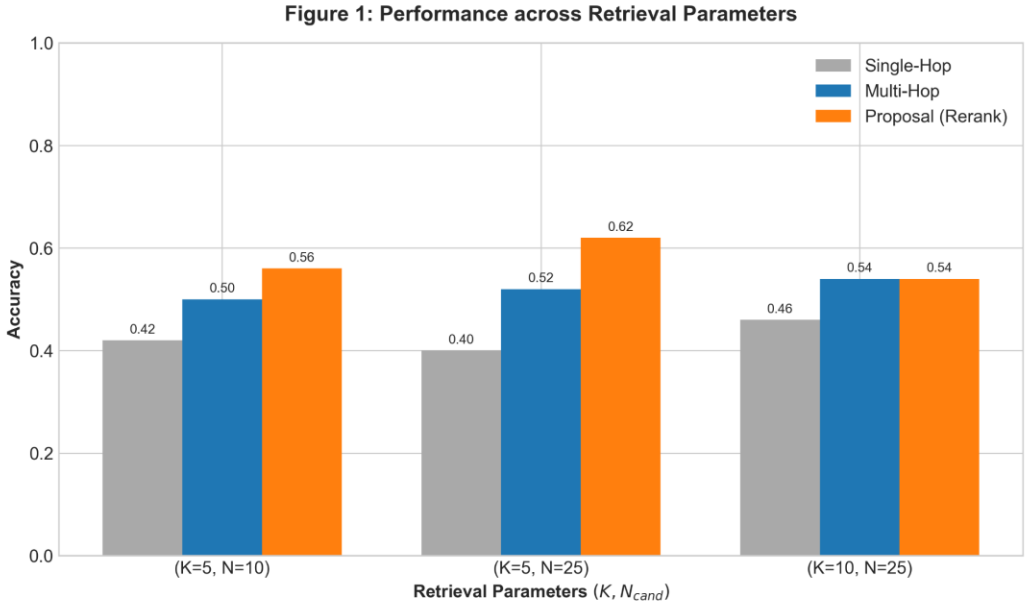
In this section, we provide a comprehensive analysis of the experimental results. We evaluate the proposed Multi-Hop RAG with Reranking against the baselines across three critical dimensions: sensitivity to retrieval hyperparameters, computational efficiency trade-offs, and robustness across varying difficulty levels.

1. Performance under Varying Retrieval Constraints

We first investigate how the three RAG pipelines behave under different retrieval configurations. Specifically, we manipulate the Recall depth (N_{cand}, the number of candidates initially fetched) and the Precision scope (K, the final context window size passed to the LLM). As illustrated in **Figure 1**, a consistent performance

hierarchy emerges in the optimal setting ($\boldsymbol{N_{cand} = 25, K = 5}$): the Proposal (Multi-Hop + Reranking) significantly outperforms the Multi-Hop baseline, which in turn surpasses the Single-Hop baseline. This confirms that complex bridge questions suffer from a semantic gap that cannot be bridged by a single vector search step.

A more nuanced phenomenon becomes apparent when we perform an ablation study on the hyperparameter constraints.



a. The "Context Saturation" Effect ($\boldsymbol{N_{cand} = 25, K = 10}$). When we relax the final context constraint by increasing K to 10, we observe that the performance gap between the Proposal and the Multi-Hop baseline narrows. This convergence suggests that GPT-3.5 possesses a latent capacity to perform "internal reranking"; if the vector search manages to place the correct document anywhere within the top 10 results, the LLM can often locate it without explicit external reranking. This indicates that the architectural value of the Cross-Encoder is most pronounced in resource-constrained environments (small K), where it acts as a critical filter to maximize the information density of the context window.

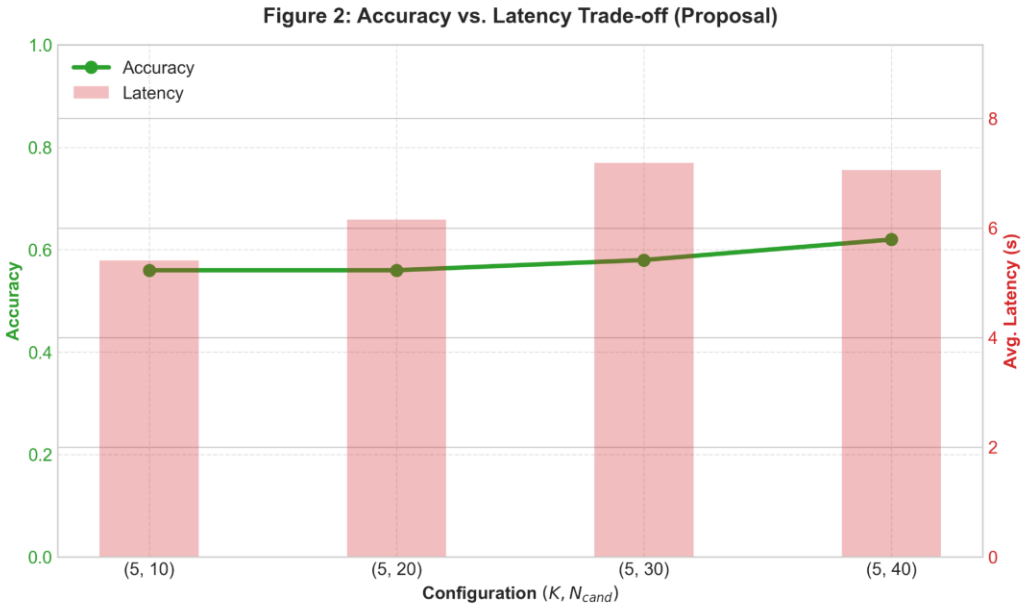
b. The "Recall Bottleneck" Effect ($\boldsymbol{N_{cand} = 10, K = 5}$). Conversely, when we restrict the initial fetch size to 10, the Proposal's performance degrades, failing to significantly outperform the baseline. This reveals a "Recall Bottleneck." Due to semantic drift, the correct document often falls outside the top-10 boundary in the vector space. If the initial fetch size is too small, the relevant document is pruned before the Reranker even has a chance to score it. This empirically validates our design choice of a "Wide Net, Tight Filter" strategy ($N >> K$), ensuring that the Reranker has a sufficient candidate pool to exercise its discriminative power.

2. The Efficiency Trade-off: Diminishing Returns of Recall

A key engineering challenge in deploying RAG systems is balancing recall quality with system latency. **Figure 2** visualizes this trade-off for the Proposal model, plotting accuracy against average query latency as the initial fetch size (N_{cand}) increases from 10 to 40, while keeping the retrieval window (K) fixed.

Contrary to the expectation that a wider search radius would yield proportional performance gains, the data reveals a distinct asymptotic trend in accuracy that contrasts sharply with the linear increase in latency. As depicted in the figure, inference time grows linearly with N_{cand} due to the computational overhead of the Cross-Encoder scoring every additional candidate pair. However, the accuracy curve exhibits an early saturation pattern. Increasing the fetch size from 10 to 20 yields modest improvements or even worse performance, and extending beyond 30 results in statistically negligible marginal gains. This behavior indicates a fundamental "Recall Ceiling" inherent to the dense embedding model. For this specific domain, the retrieval outcome tends to be binary: either the relevant document is semantically close enough to appear in the top tier of results, or it suffers from severe semantic drift and falls outside the retrieval horizon entirely. There is little evidence of a "middle ground" where correct documents are consistently buried between rank 30 and 40. Consequently, blindly increasing the prefetch buffer beyond N_{cand} introduces significant computational waste without tangible benefits. Based on this cost-benefit analysis, we identify $N_{cand} = 20$ as the Pareto-optimal configuration, balancing robust recall with acceptable system latency.

3. Robustness and Failure Mode Analysis



Finally, we assessed the generalization capabilities of the models by stratifying the test set into "Medium" and "Hard" difficulty levels. **Figure 3** depicts the performance degradation observed across all models when transitioning to harder questions.

It is notable that while our Proposal maintains its lead in the Medium category, the performance advantage over the Multi-Hop baseline diminishes in the Hard category. This result points to a critical "Garbage In, Garbage Out" bottleneck in the RAG pipeline. Hard questions in HotpotQA often involve obscure entities or highly indirect relationships that challenge the initial query decomposition step. If the first-hop retrieval fails to recall the necessary "bridge" document—a scenario significantly more probable with Hard questions—the Reranker receives a candidate list consisting entirely of noise. Since a Reranker is a discriminative model rather than a generative one, it cannot recover information that was never retrieved. This finding highlights that while Cross-Encoder Reranking is a powerful tool for filtering distractors and improving precision, it cannot compensate for fundamental failures in the initial Recall stage. Future optimizations for hard queries should thus prioritize upstream improvements, such as hybrid search (BM25

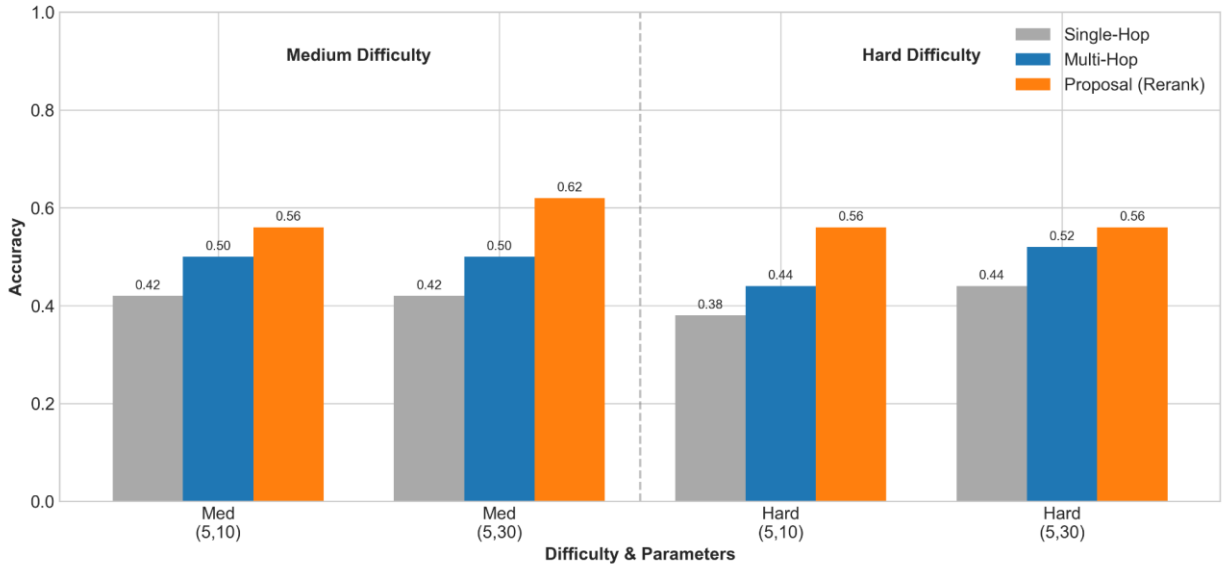
[] + Dense) or enhanced query reformulation strategies, rather than simply scaling the downstream reranking component.

Conclusion & Outlook

In this work, we presented a robust Multi-Hop RAG pipeline augmented with Cross-Encoder Reranking to tackle complex, bridge-type questions. By systematically evaluating our approach against standard baselines on the HotpotQA dataset, we demonstrated that query decomposition is a prerequisite for multi-step reasoning, while reranking serves as a critical precision amplifier in constrained retrieval environments.

Our experiments established a clear hierarchy of competence, with the proposed method consistently outperforming both Single-Hop and Standard Multi-Hop baselines, achieving a peak accuracy of roughly

Figure 3: Robustness Analysis (Medium vs. Hard Difficulty)



60% on medium-difficulty bridge questions. We empirically validated a "Wide Net, Tight Filter" strategy, showing that a large initial fetch size ($N_{cand} = 30$) coupled with a Cross-Encoder Reranker effectively mitigates semantic drift by filtering out high-confidence distractors that often confuse standard vector search. Furthermore, our efficiency analysis revealed a distinct "Recall Ceiling," where blindly increasing the candidate pool beyond a certain threshold yield diminishing returns. This indicates that the primary failure mode for hard questions is not ranking error, but the fundamental failure of the embedding model to retrieve the correct document in the first place.

Looking forward, the performance degradation observed on "Hard" questions highlights the necessity for upstream improvements beyond simple reranking. Future research should prioritize integrating sparse retrieval (BM25 []) with dense embeddings to improve recall for rare entities often missed by semantic vector search. Additionally, implementing advanced query rewriting techniques such as HyDE [] could generate more diverse search queries, increasing the probability of retrieving the correct document during the initial fetch stage. Finally, exploring end-to-end optimization by fine-tuning the embedding model specifically for multi-hop reasoning tasks offers a promising avenue to align the vector space more closely with the logical structure of bridge questions.

References

[1] Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks." Advances in neural information processing systems 33 (2020): 9459-9474.

[2] Gao, Yunfan, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. "Retrieval-augmented generation for large language models: A survey." arXiv preprint arXiv:2312.10997 2, no. 1 (2023).

[3] Guu, Kelvin, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. "Retrieval augmented language model pre-training." In International conference on machine learning, pp. 3929-3938. PMLR, 2020.

[4] Yang, Zhilin, 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, pp. 2369-2380. 2018.

[5] Ho, Xanh, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. "Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps." arXiv preprint arXiv:2011.01060 (2020).

[6] Fang, Yuwei, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. "Hierarchical graph network for multi-hop question answering." In Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP), pp. 8823-8838. 2020.

[7] Fu, Ruiliu, Han Wang, Xuejun Zhang, Jun Zhou, and Yonghong Yan. "Decomposing complex questions makes multi-hop QA easier and more interpretable." arXiv preprint arXiv:2110.13472 (2021).

[8] Zhou, Denny, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans et al. "Least-to-most prompting enables complex reasoning in large language models." arXiv preprint arXiv:2205.10625 (2022).

[9] Press, Ofir, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. "Measuring and narrowing the compositionality gap in language models." In Findings of the Association for Computational Linguistics: EMNLP 2023, pp. 5687-5711. 2023.

[10] Wei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.

[11] Karpukhin, Vladimir, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. "Dense Passage Retrieval for Open-Domain Question Answering." In EMNLP (1), pp. 6769-6781. 2020.

[12] Nguyen, Tri, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. "Ms marco: A human-generated machine reading comprehension dataset." (2016).

[13] Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." arXiv preprint arXiv:1901.04085 (2019).

[14] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

[15] Liu, Nelson F., Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. "Lost in the middle: How language models use long contexts." Transactions of the Association for Computational Linguistics 12 (2024): 157-173.

[16] Mallen, Alex, 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), pp. 9802-9822. 2023.

[17] Gao, Luyu, Xueguang Ma, Jimmy Lin, and Jamie Callan. "Precise zero-shot dense retrieval without relevance labels." In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1762-1777. 2023.

[18] Shen, Tao, Guodong Long, Xiubo Geng, Chongyang Tao, Tianyi Zhou, and Daxin Jiang. "Large language models are strong zero-shot retriever." arXiv preprint arXiv:2304.14233 (2023).

[19] Liu, Yang, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. "G-eval: NLG evaluation using gpt-4 with better human alignment." arXiv preprint arXiv:2303.16634 (2023).

[20] Zhu, Lianghui, Xinggang Wang, and Xinlong Wang. "Judgelm: Fine-tuned large language models are scalable judges." arXiv preprint arXiv:2310.17631 (2023).

[21] Hoffmann, Jordan, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas et al. "Training compute-optimal large language models." arXiv preprint arXiv:2203.15556 (2022).

[22] Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." arXiv preprint arXiv:2001.08361 (2020).

[23] Topsakal, Oguzhan, and Tahir Cetin Akinci. "Creating large language model applications utilizing langchain: A primer on developing llm apps fast." In International conference on applied engineering and natural sciences, vol. 1, no. 1, pp. 1050-1056. 2023.

[24] Jin, Jiajie, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. "Flashrag: A modular toolkit for efficient retrieval-augmented generation research." In Companion Proceedings of the ACM on Web Conference 2025, pp. 737-740. 2025.

[25] Kwon, Woosuk, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. "Efficient memory management for large language model serving with pagedattention." In Proceedings of the 29th symposium on operating systems principles, pp. 611-626. 2023.

[26] Douze, Matthijs, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. "The faiss library." IEEE Transactions on Big Data (2025).

[27] Borgeaud, Sebastian, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche et al. "Improving language models by retrieving from trillions of tokens." In International conference on machine learning, pp. 2206-2240. PMLR, 2022.

[28] Jiang, Zhengbao, Frank F. 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, pp. 7969-7992. 2023.

Authors

Hail Lim graduated from Michigan State University in Spring 2025 with a degree in Computer Science. His work focuses on AI, computer vision, and data engineering, including projects in RAG systems, YOLO-based wildlife monitoring, and robotics. He plans to pursue graduate studies in data science.