Bridging the Long-Tail Gap: Robust Retrieval-Augmented Relation Completion via Multi-Stage Paraphrase Infusion

Fahmida Alam, Mihai Surdeanu, Ellen Riloff
Department of Computer Science
University of Arizona, Tucson, AZ, USA
{fahmidaalam, msurdeanu, riloff}@arizona.edu

Abstract

Large language models (LLMs) struggle with relation completion (RC), both with and without retrieval-augmented generation (RAG), particularly when the required information is rare or sparsely represented. To address this, we propose a novel multi-stage paraphrase-guided *relation-completion RAG* framework, ***RC-RAG***, that systematically incorporates relation paraphrases across multiple stages. In particular, *RC-RAG*: (a) integrates paraphrases into retrieval to expand lexical coverage of the relation, (b) uses paraphrases to generate relation-aware summaries, and (c) leverages paraphrases during generation to guide reasoning for relation completion. Experiments with five LLMs on two benchmark datasets show that *RC-RAG* consistently outperforms several RAG baselines. In long-tail settings, the best-performing LLM augmented with *RC-RAG* improves by 50.4 Exact Match (EM) points over its standalone performance and surpasses two strong RAG baselines by 30.4 and 16.0 EM points, respectively, without requiring additional model training or fine-tuning.[1]

1 Introduction

Relation completion (RC) is an important task for large language models (LLMs), as it enables models to infer and reason about relations between entities. This capability supports a variety of applications, including knowledge graph completion and multi-hop reasoning, where LLMs are increasingly used as a central reasoning component (Wei et al., 2023; Yao et al., 2025). Although LLMs perform well on many tasks, they still struggle with relation completion. In particular, LLM performance deteriorates significantly when the required knowledge is rare or lies in the long tail of the data distribution. In such sparse settings, retrieval-augmented generation (RAG) (Guu et al., 2020; Lewis et al., 2021; Izacard et al., 2022b) is used to provide external evidence to LLMs. While RAG has shown strong performance across many NLP tasks, experimental results indicate that state-of-the-art RAG approaches still perform poorly for relation completion. This issue becomes even more severe when the underlying relations are rare or occur in long-tail settings.

Formally, given a triple $(e_1, r, ?)$, where e_1 denotes the head entity and r denotes the relation, the goal of relation completion (RC) is to predict the missing tail entity e_2 to complete the triple (e_1, r, e_2). We define a triple as long-tail[2] if the co-occurrence frequency of the entity pair (e_1, e_2) within the same sentence of a passage in the corpus is below a threshold x. Under this definition, most triples (e_1, r, e_2) have very limited evidence of the target relation in the corpus (see Figure 5 in Appendix C). For example, the pair (*Bruno Zevi*, *Sapienza University*) appears only three times in the corpus used in this work. When conditioned on the relation `educated at`, the available evidence becomes even more sparse. Moreover, relations can be expressed through diverse paraphrases. For instance, `educated at` may be expressed in text using the paraphrase `alma mater`. If such variations are not considered during retrieval, relevant evidence may not be retrieved.

In this paper, we propose a novel *relation-completion RAG* framework, ***RC-RAG***. The novelty of our framework lies in incorporating relation paraphrases across multiple stages. First, they are infused into a hybrid retrieval mechanism to expand lexical coverage of relation expressions. This enhances the system's capacity to retrieve relevant evidence even when explicit mentions of the relation are sparse. The same paraphrases are further

[1] Code and data are available at https://github.com/alamfahmida/RC-RAG.

[2] We acknowledge that several other scenarios may fall under the "long-tail" umbrella. We leave the development of a comprehensive taxonomy of long-tail relations for future work.

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 2805–2823
November 6-10, 2026 ©2026 Association for Computational Linguistics

leveraged to produce relation-aware summaries that reduce noise and guide the model toward relevant evidence. Finally, they are used during generation to focus the model's attention on relation-specific cues, enabling stronger reasoning for relation completion, without requiring any model fine-tuning.

This work studies RAG performance on relation completion (RC) across frequency regimes, from high-frequency to long-tail relations. Our analysis reveals how sparsity affects retrieval quality and motivates the design of methods to address this challenge.

All in all, our contributions are:

(1) We propose a novel RAG framework, ***RC-RAG***, for relation completion that incorporates relation paraphrases across multiple stages of the framework. These paraphrases expand lexical coverage during retrieval, help identify relevant information during evidence summarization, and guide reasoning during generation, addressing the challenge of relation sparsity. To our knowledge, we are the first to explore the impact of paraphrases for RC under RAG.

(2) We evaluate *RC-RAG* on two benchmark datasets (MALT and Wikidata5M) across five LLMs, demonstrating consistent improvements over strong state-of-the-art RAG baselines. Importantly, in the long-tail setting, our framework surpasses two competitive RAG baselines by 30.4 and 16.0 EM points, respectively. *RC-RAG* requires no model fine-tuning, making it broadly applicable across different LLMs.

(3) This work is the first to conduct a systematic frequency-aware analysis of RAG performance for relation completion. Our results show that existing methods degrade sharply in long-tail settings, while our framework remains robust, providing an initial step toward addressing long-tail challenges in relation completion.

(4) We introduce a reproducible evaluation framework in which RAG baselines and *RC-RAG* operate under identical settings, using the same retrieval corpus, benchmark datasets, and evaluation metrics to ensure fair comparison.

2 Related Work

Frequency Bias and Memorization in LLMs: LLMs have been shown to store large amounts of factual knowledge in their parametric memory, enabling strong performance on a wide range of knowledge-intensive tasks without explicit retrieval mechanisms (Petroni et al., 2019; Roberts et al., 2020). However, LLMs perform poorly in relation completion, particularly when the required information is sparse. Recent studies show that parametric knowledge in LLMs is unevenly distributed and influenced by the frequency with which facts appear in the pretraining data. In particular, LLMs are more likely to recall facts frequently observed during pretraining, while rare or infrequent facts are harder to extract (Carlini et al., 2023; Kandpal et al., 2023). This frequency-dependent memorization behavior poses a challenge in long-tail settings, where the required facts are rare or weakly represented in the data distribution. These limitations highlight the need for RAG frameworks that can retrieve relevant context even under sparse evidence conditions without requiring LLM fine-tuning.

RAG for Relation Completion: RAG-based methods augment LLMs with external non-parametric memory by retrieving relevant documents at inference time and conditioning generation on the retrieved evidence (Guu et al., 2020; Karpukhin et al., 2020; Lewis et al., 2021; Shuster et al., 2021; Asai et al., 2023; Xu et al., 2023; Jiang et al., 2023; Reichman and Heck, 2024). This approach has been shown to improve factual accuracy and robustness across tasks such as open-domain question answering, fact verification, and knowledge-intensive reasoning (Khattab and Zaharia, 2020; Izacard and Grave, 2021; Ram et al., 2023). However, the application of RAG to RC remains relatively underexplored, especially in long-tail settings. In practice, the information required to complete a relation may be sparse or weakly expressed in the corpus, making it difficult for retrieval mechanisms to identify sufficient contextual evidence. These challenges highlight the need for RAG frameworks that remain robust across varying evidence frequencies and support relation completion under sparse evidence conditions.

Architectural and Analytical Challenges of Current RAG Frameworks: Existing RAG frameworks are often architecturally complex and computationally expensive, relying on sophisticated machine learning components that require extensive fine-tuning, task-specific customization, and, in some cases, synthetic data generation for instruction tuning (Lin et al., 2024; Asai et al., 2023; Xu et al., 2023). Moreover, these methods operate as black-box systems, providing limited insight into model behavior. In particular, there is no system-

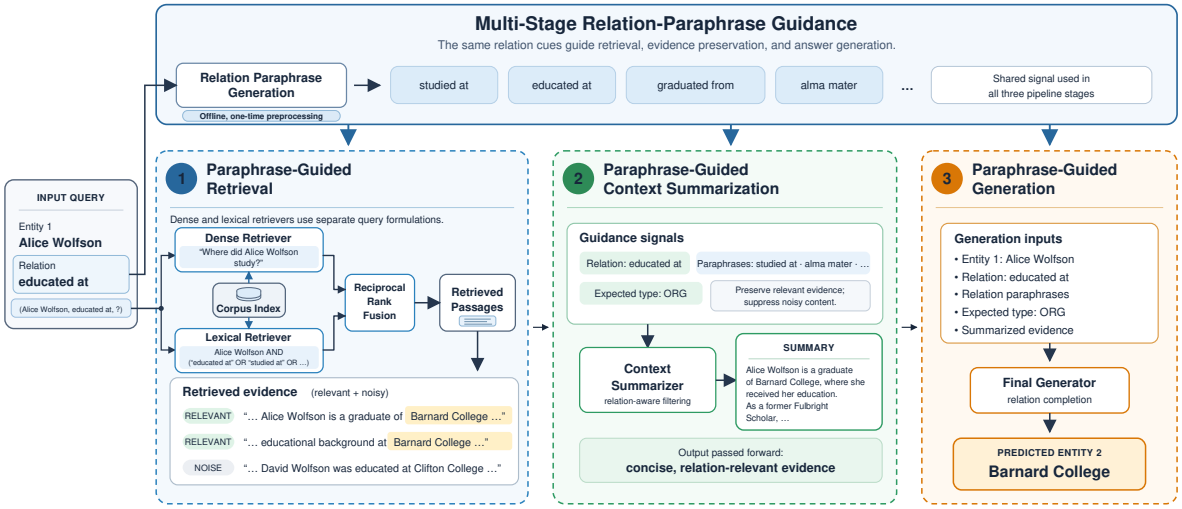


Figure 1: Overview of our multi-stage paraphrase-guided RAG framework, ***RC-RAG***, for relation completion. Given an entity–relation query, we generate relation paraphrases of the target relation and inject them across three stages: (1) paraphrase-guided hybrid retrieval to expand semantic coverage, (2) paraphrase-guided context summarization to extract relation-relevant information, and (3) paraphrase-guided generation to guide relation completion.

atic analysis of the sensitivity of RAG frameworks to data sparsity, making it difficult to diagnose failure modes or ensure robust performance across different frequency conditions. In this work, we take a first step toward addressing these challenges by studying retrieval performance under varying data frequency conditions.

3 Proposed Method

We propose *RC-RAG*, a novel RAG framework for relation completion that consists of three stages: (1) paraphrase-guided hybrid retrieval, (2) paraphrase-guided context summarization, and (3) paraphrase-guided generation. Figure 1 illustrates the framework design in detail.

Relation Paraphrases Generation: To capture lexical variations of relations, we construct a *relation paraphrase set* for each target relation. For a relation r, we define a paraphrase set $P_r = \{p_1, p_2, \ldots, p_k\}$, where each p_i represents a natural language expression conveying the same relational meaning as r. For example, for the relation $r =$ `founded by`, the paraphrase set may include expressions such as `established by`, `created by`, and `started by`. This step is performed offline as a one-time preprocessing procedure (Figure 1). The complete list is provided in Table 5 in Appendix A and is used across all three stages of *RC-RAG*. We generate this set for each relation using an LLM[3] with the prompt template shown in Figure 2 in Appendix A.

3.1 Paraphrase-Guided Hybrid Retrieval

In practice, relevant evidence in the RAG corpus may be expressed using linguistic variations that differ from the target relation, which can lead to retrieval failures. For example, when the relation r is `educated at`, the corpus may contain information expressed using paraphrases such as *alma mater*. This issue becomes more severe in long-tail scenarios, where the required evidence is scarce, and failing to capture such variations can significantly affect retrieval performance. To mitigate this issue, we incorporate relation paraphrases into the hybrid retriever, enabling the retriever to capture relevant evidence even when explicit mentions of the target relation are sparse. To operationalize paraphrase-guided hybrid retrieval, we leverage both lexical and dense retrieval signals, incorporating relation paraphrases into the lexical retriever to expand lexical coverage during retrieval.

3.1.1 Lexical Retrieval

Lexical retrieval captures surface-level variations of a relation through explicit textual matches. To leverage this property, we augment the retrieval query with relation paraphrases that provide linguistic variants of the target relation. Given a head

[3]Main experiments use `gpt-4o-mini-2024-07-18`; we also evaluate `Meta-Llama-3.1-8B-Instruct` (Appendix H).

entity e_1 and relation r, we construct lexical queries using the paraphrase set $P_r = \{p_1, \ldots, p_k\}$ and retrieve passages using a disjunction over conjunctions $(e_1 \wedge p_1) \vee \cdots \vee (e_1 \wedge p_k)$. Incorporating paraphrases into the retrieval query increases the likelihood of retrieving relevant passages even when explicit mentions of the relation are scarce, and helps disentangle cases where the dense retriever retrieves passages describing the target relation for a different entity. We implement lexical retrieval using an inverted index built with Elasticsearch,[4] ranking passages with the BM25 scoring function (Robertson and Zaragoza, 2009).

3.1.2 Dense Retrieval

Relevant evidence may also appear in implicit or semantically related formulations that do not share exact lexical matches with the target relation. To capture such cases, we complement lexical retrieval with dense retrieval, which models deeper semantic similarity between the query and corpus passages. Given an input (e_1, r), we construct a query using a relation-specific question-answering (QA) template (see Table 6 in Appendix B) that encodes the semantics of the target relation. These templates are automatically generated using an LLM.[5] For example, for the relation r = `founded by`, the query becomes *Who founded {entity}?*. The dense retriever operates over a Wikipedia passage corpus with precomputed Contriever embeddings[6] generated using the Contriever model trained on MS MARCO (Izacard et al., 2022a), which is widely used in state-of-the-art RAG systems (Asai et al., 2023; Xu et al., 2023). We retrieve the top-k passages using dense vector similarity.

3.1.3 Reciprocal Rank Fusion

We independently query the lexical and dense retrievers and combine their ranked outputs using Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), an unsupervised rank aggregation method that prioritizes items highly ranked by any retriever and has been shown effective for combining heterogeneous retrieval signals in RAG settings (Islam et al., 2025). Given the set of retrievers $\mathcal{R}$ and the rank $\mathrm{rank}_r(p)$ of passage p returned by retriever $r \in \mathcal{R}$, the RRF score is computed as:

$$\mathrm{RRF}(p) = \sum_{r \in \mathcal{R}} \frac{1}{c_{\mathrm{rrf}} + \mathrm{rank}_r(p)}. \quad (1)$$

In all experiments, we set $c_{\mathrm{rrf}} = 0$ to emphasize top-ranked passages and select the top-k passages based on the fused scores. We analyze how different values of c_{rrf} affect performance and justify our choice of $c_{\mathrm{rrf}} = 0$ in Appendix I.

3.2 Paraphrase-Guided Context Summarization

After retrieval, the collected evidence often contains substantial noise and irrelevant information. When the retrieved context is long or loosely related to the target relation, LLMs may become distracted by irrelevant details, which can negatively affect downstream reasoning. To mitigate this issue, we guide the summarization stage using relation paraphrases and entity-type constraints. Specifically, sentences containing relation paraphrases are assigned higher importance, as they are more likely to express the target relation. In addition, we prioritize evidence that mentions entities consistent with the expected type of the target entity e_2. These signals help the model focus on relationally relevant information while filtering out distracting context.

We use GPT-4o-mini[7] with the prompt shown in Figure 3 to generate a relation-aware summary of the retrieved passages. We also evaluate an open-source alternative to demonstrate that *RC-RAG* does not depend on proprietary models (see Appendix H). For each relation r, the expected type of the target entity e_2 is determined through a relation-to-type mapping, where each relation corresponds to a small and well-defined set of plausible entity types (e.g., `GPE` or `LOC` for `place of birth`). We use a manually constructed mapping in the main experiments, which is provided in Table 7 in Appendix B. We also evaluate an automatically generated LLM-based mapping in Appendix B, showing that comparable performance can be achieved without requiring a manually constructed mapping.

3.3 Paraphrase-Guided Generation

During the generation stage, relation paraphrases guide the reasoning process. Even after the relation-aware summarization stage, the summary may still contain multiple entities and contextual details that

[4] https://www.elastic.co/elasticsearch

[5] We use `gpt-4o-mini-2024-07-18` in the main experiments; `Meta-Llama-3.1-8B-Instruct` produces nearly identical performance (Appendix H).

[6] https://dl.fbaipublicfiles.com/contriever/embeddings/contriever-msmarco/wikipedia_embeddings.tar

[7] We use gpt-4o-mini-2024-07-18.

are not directly relevant to the target relation. To address this challenge, we incorporate paraphrases of the target relation to guide the generator LLM's attention toward relationally relevant sentences in the summary. This mechanism encourages the model to prioritize evidence aligned with the semantic intent of the relation, thereby improving its ability to identify the correct tail entity while reducing distractions from irrelevant context. We use five different LLMs (see Section 4.1) as RC generators to predict the target tail entity e_2 given the input pair (e_1, r), the relation-aware summary, and relation paraphrases that guide generation. The generation prompt is shown in Figure 4 in Appendix A.

4 Experimental Setup

4.1 Large Language Models for Generation

We evaluate five open-source LLMs as relation completion (RC) generators: Llama-2-7b-chat-hf, Llama-2-13b-chat-hf, Meta-Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Mixtral-8x7B-Instruct-v0.1. To ensure a fair comparison, all models use the same generation prompt. No task-specific fine-tuning is performed; each model is used in its original pretrained or instruction-tuned form. Additional implementation details, prompts, and hyperparameters are provided in Appendix A, along with the inference latency and API cost of *RC-RAG*.

4.2 RAG Settings

4.2.1 RAG Corpus

Following prior work (Asai et al., 2023; Karpukhin et al., 2020), we use the same Wikipedia passage corpus[8] for all retrieval components in our framework, as well as for all RAG baselines, to ensure a fair and consistent comparison.

4.2.2 Top-k Retrieval Settings

We evaluate *RC-RAG* and all baselines under two retrieval settings based on the number of top-ranked passages used as context: Top-10 and Top-20. This allows us to compare the methods using the same amount of retrieved context and to assess the effect of increasing the amount of retrieved context.

4.3 Datasets

We evaluate our framework on two benchmark datasets: MALT (Chen et al., 2023) and Wikidata5M (Wang et al., 2021). MALT is specifically designed for long-tail knowledge base completion, so we evaluate it using a balanced global subset without additional frequency partitioning. In contrast, Wikidata5M spans a wide range of occurrence frequencies. Therefore, we report both global results and frequency-based analyses on a curated subset to assess robustness across frequency regimes, from high-frequency to long-tail settings. Our evaluation covers all eight relations in MALT and twelve relations in the Wikidata5M subset. The complete list of relations for both datasets is provided in Table 5 in Appendix A. Details on frequency-based partitioning, long-tail threshold selection, and dataset construction are provided in Appendix C.

4.4 Baselines

4.4.1 Parametric LLM Baselines (No Context)

We evaluate five LLMs to assess their performance in relation completion across frequency bins (high-frequency to long-tail) when relying solely on their parametric memory (see Section 4.1). We use the same generation prompt shown in Figure 4 in Appendix A, but remove both retrieved contextual evidence and relation paraphrases. This setting isolates the models' parametric factual knowledge and enables analysis of performance degradation across frequency partitions, highlighting the impact of relational sparsity in long-tail settings even for strong pretrained LLMs.

4.4.2 RAG Baselines

We compare *RC-RAG* against three RAG baselines: Vanilla RAG, SELF-RAG (Asai et al., 2023), and RECOMP (Xu et al., 2023).

Vanilla RAG. To assess the contribution of the components introduced in *RC-RAG*, we implement a standard retrieve-then-generate baseline in which retrieved passages are provided directly to the LLM, without evidence summarization or paraphrase guidance. This setup provides a direct comparison with conventional RAG and quantifies the gains of the proposed framework.

SELF-RAG and RECOMP. Since both methods are primarily designed and evaluated for question answering (QA), we cast the relation completion task as a QA problem to enable a fair comparison. Both baselines rely on sophisticated architectures that require substantial additional training beyond

[8] https://dl.fbaipublicfiles.com/dpr/wikipedia_split/psgs_w100.tsv.gz

the base generator, including task-specific components and large-scale supervision. SELF-RAG is released with LLaMA-2-7B and LLaMA-2-13B variants; we use the 7B variant in the main results and additionally evaluate the 13B variant as a model-size control. The architectural details of RECOMP and SELF-RAG, along with the SELF-RAG model-size comparison, are provided in Appendix D and Appendix D.1.1, respectively.

4.5 Evaluation Metrics

We evaluate relation completion using exact match (EM) and approximate match (AM) accuracy. EM requires an exact match to the gold tail entity; however, unlike MALT, Wikidata5M provides only a single canonical surface form, penalizing semantically correct variants (e.g., *Chevy Chase, Maryland* vs. *Chevy Chase*). To mitigate surface-form mismatches, we adopt AM based on word-level Jaccard similarity with a 60% threshold. We select this threshold based on a manual analysis of 50 Wikidata5M prediction–gold answer pairs. A lower threshold of 33% was overly permissive, with 24 of 25 cases accepted by the metric being incorrect, while a higher threshold of 80% was overly strict, rejecting all 24 semantically correct cases examined. This analysis supports our choice of the 60% threshold. At this threshold, *Chevy Chase, Maryland* is counted as correct for the gold answer *Chevy Chase* (2/3 similarity), while *Orlando, FL* is counted as incorrect for *Jacksonville, FL* (1/3 similarity).

5 Results and Analysis

5.1 Overall Performance Across Datasets

We evaluate *RC-RAG* across five generator LLMs and compare it with Vanilla RAG, SELF-RAG, RECOMP, and no-context baselines.

Tables 1 and 2 report the results on MALT and Wikidata5M, respectively. We draw the following key observations:

(1) *RC-RAG* consistently outperforms RAG baselines. *RC-RAG* strongly outperforms Vanilla RAG, RECOMP, and SELF-RAG across datasets under Top-10 and Top-20 retrieval. Notably, these gains persist even with the smallest LLM in our evaluation, Llama-2-7b-chat-hf. The performance margin increases with stronger LLMs. This consistency across all five LLMs indicates that *RC-RAG* is not dependent on a particular model or retrieval setting.

Context Length	Model	EM (%)	AM (%)
Baselines			
No-context	Llama-2-7B-chat-hf	13.6	17.0
	Llama-2-13B-chat-hf	16.0	19.4
	Meta-Llama-3.1-8B-Instruct	22.8	27.7
	Mistral-7B-Instruct-v0.3	16.1	20.1
	Mixtral-8x7B-Instruct-v0.1	34.7	38.4
Top-10	Vanilla RAG	51.9	54.0
	RECOMP	37.1	39.0
	SELF-RAG	54.2	57.3
Top-20	Vanilla RAG	51.5	53.4
	RECOMP	31.9	34.6
	SELF-RAG	52.8	56.0
RC-RAG (Ours)			
Top-10	Llama-2-7B-chat-hf	58.6	60.9
	Llama-2-13B-chat-hf	58.9	61.1
	Meta-Llama-3.1-8B-Instruct	59.3	61.3
	Mistral-7B-Instruct-v0.3	60.6	63.2
	Mixtral-8x7B-Instruct-v0.1	61.0	63.3
Top-20	Llama-2-7B-chat-hf	59.3	61.5
	Llama-2-13B-chat-hf	60.5	62.9
	Meta-Llama-3.1-8B-Instruct	60.3	62.6
	Mistral-7B-Instruct-v0.3	**62.7**	**65.1**
	Mixtral-8x7B-Instruct-v0.1	62.2	64.8

Table 1: Overall performance of *RC-RAG* and baselines on MALT under Top-10 and Top-20 retrieval settings.

(2) *RC-RAG* substantially improves LLM performance over no-context baselines. *RC-RAG* improves performance over the corresponding no-context LLM baselines on both datasets. Even the strongest no-context generator, Mixtral-8x7B-Instruct-v0.1, performs poorly on its own, but achieves substantially higher performance when augmented with *RC-RAG*. This highlights the limitations of relying solely on parametric knowledge for relation completion, particularly when relational evidence is sparse.

(3) *RC-RAG* benefits from additional retrieved context, while the RAG baselines do not. Across both datasets, every *RC-RAG* configuration improves when retrieval depth increases from Top-10 to Top-20. On MALT, all three RAG baselines degrade with additional context. On Wikidata5M, RECOMP and SELF-RAG also degrade, while Vanilla RAG shows only a small improvement. These results suggest that *RC-RAG* is better able to leverage additional retrieved evidence without being substantially affected by the noise introduced by deeper retrieval.

Context Length	Model	EM (%)	AM (%)
Baselines			
No-context	Llama-2-7B-chat-hf	21.0	23.4
	Llama-2-13B-chat-hf	21.9	23.7
	Meta-Llama-3.1-8B-Instruct	30.0	32.6
	Mistral-7B-Instruct-v0.3	25.7	27.5
	Mixtral-8x7B-Instruct-v0.1	37.3	40.7
Top-10	Vanilla RAG	49.2	52.5
	RECOMP	47.5	50.9
	SELF-RAG	52.1	55.3
Top-20	Vanilla RAG	49.9	52.9
	RECOMP	39.9	42.7
	SELF-RAG	50.3	53.3
RC-RAG (Ours)			
Top-10	Llama-2-7B-chat-hf	59.1	62.4
	Llama-2-13B-chat-hf	61.7	65.0
	Meta-Llama-3.1-8B-Instruct	61.1	64.6
	Mistral-7B-Instruct-v0.3	63.8	67.5
	Mixtral-8x7B-Instruct-v0.1	64.5	68.7
Top-20	Llama-2-7B-chat-hf	60.8	64.0
	Llama-2-13B-chat-hf	62.8	65.7
	Meta-Llama-3.1-8B-Instruct	62.0	65.3
	Mistral-7B-Instruct-v0.3	**65.5**	**69.1**
	Mixtral-8x7B-Instruct-v0.1	**65.5**	68.6

Table 2: Overall performance of *RC-RAG* and baselines on Wikidata5M under Top-10 and Top-20 retrieval settings.

Open-Source Evaluation. We additionally evaluate a fully open-source *RC-RAG* pipeline for paraphrase generation, QA-template generation, entity-type mapping, and summarization. The results show that *RC-RAG* remains effective without relying on proprietary LLMs; detailed results are provided in Appendix H.

5.2 Frequency-Based Performance Analysis on Wikidata5M

Table 3 presents a frequency-based evaluation of our RAG framework, *RC-RAG*, and all baselines on the Wikidata5M dataset. For this analysis, we partition the data into three frequency regimes: high-frequency, mid-frequency, and long-tail. The definitions and partitioning procedure are described in Appendix C.

(1) Existing RAG methods degrade sharply as relation frequency decreases, while *RC-RAG* remains robust. As shown in Table 3, Vanilla RAG, RECOMP, and SELF-RAG show substantial performance degradation from high-frequency to long-tail instances under both Top-10 and Top-20. In contrast, *RC-RAG* shows a much smaller decline as relation frequency decreases, even with the smallest LLM in our evaluation, and this robustness persists across stronger LLMs. These results indicate that *RC-RAG* is substantially more robust to relational sparsity than existing RAG methods.

(2) *RC-RAG* outperforms all RAG baselines across frequency regimes. Across all LLMs and both retrieval settings, *RC-RAG* consistently and substantially outperforms Vanilla RAG, RECOMP, and SELF-RAG in the high-, mid-, and long-tail regimes. The performance gains of *RC-RAG* over the baselines increase as relation frequency decreases. In the long-tail setting, the gain is particularly large, showing that *RC-RAG* remains robust under sparse relational evidence.

(3) No-context LLM performance degrades sharply as relation frequency decreases. As shown in Table 3, all no-context baselines decline sharply as relation frequency decreases, with particularly severe degradation in the long-tail regime. This pattern persists even for Mixtral-8x7B-Instruct-v0.1, the strongest no-context generator in our evaluation, indicating that increasing model capability alone does not overcome sparse relational evidence. In contrast, augmenting the same LLM with *RC-RAG* substantially mitigates this degradation and maintains strong long-tail performance. These results highlight the importance of retrieved relational evidence for robust long-tail relation completion.

(4) *RC-RAG* benefits from deeper retrieval in sparse frequency regimes, unlike existing RAG methods. As shown in Table 3, increasing retrieval from Top-10 to Top-20 improves *RC-RAG* across all generator LLMs in both the mid-frequency and long-tail subsets. In contrast, RECOMP and SELF-RAG consistently lose performance with deeper retrieval across all frequency regimes, while Vanilla RAG shows only small and mixed changes. These results indicate that *RC-RAG* can effectively leverage additional retrieved evidence when relational evidence is sparse, while existing RAG methods are more susceptible to noise introduced by deeper retrieval.

(5) High-frequency instances are more affected by ambiguity and retrieval noise. Interestingly, the mid-frequency bucket slightly outperforms the high-frequency bucket in Table 3. To investigate this pattern, we conduct two analyses. First, we

Context Length	Model	Frequency					
		High-frequency		Mid-frequency		Long-tail	
		EM (%)	AM (%)	EM (%)	AM (%)	EM (%)	AM (%)
Baselines							
No-context	Llama-2-7B-chat-hf	38.8	40.4	16.4	18.6	7.80	11.2
	Llama-2-13B-chat-hf	41.8	43.6	17.4	18.6	6.60	8.80
	Meta-Llama-3.1-8B-Instruct	51.0	52.6	28.0	30.8	11.0	14.4
	Mistral-7B-Instruct-v0.3	47.2	49.0	21.6	22.6	8.20	10.8
	Mixtral-8x7B-Instruct-v0.1	57.2	61.0	38.4	41.8	16.2	19.2
Top-10	Vanilla RAG	53.4	56.2	51.6	54.6	42.6	46.8
	RECOMP	52.2	56.6	49.4	51.4	40.8	44.8
	SELF-RAG	60.8	63.6	52.6	54.8	43.0	47.6
Top-20	Vanilla RAG	55.0	57.6	51.0	53.8	43.6	47.4
	RECOMP	49.4	53.0	42.0	43.4	28.2	31.8
	SELF-RAG	57.4	60.0	50.8	52.8	42.6	47.2
RC-RAG (ours)							
Top-10	Llama-2-7B-chat-hf	60.8	64.0	63.8	65.6	52.8	57.6
	Llama-2-13B-chat-hf	63.8	67.0	66.8	69.0	54.4	59.0
	Meta-Llama-3.1-8B-Instruct	63.2	67.2	66.0	67.6	54.0	59.0
	Mistral-7B-Instruct-v0.3	67.4	71.2	67.8	69.8	56.2	61.4
	Mixtral-8x7B-Instruct-v0.1	67.4	**71.4**	69.2	71.0	56.8	61.8
Top-20	Llama-2-7B-chat-hf	62.2	65.2	65.2	67.4	55.0	59.4
	Llama-2-13B-chat-hf	64.2	67.0	68.4	70.4	55.8	59.8
	Meta-Llama-3.1-8B-Instruct	62.4	65.6	68.2	70.0	55.4	60.2
	Mistral-7B-Instruct-v0.3	**67.8**	**71.4**	70.2	72.4	**58.6**	**63.4**
	Mixtral-8x7B-Instruct-v0.1	**67.8**	71.0	**70.8**	**72.6**	57.8	62.2

Table 3: Frequency-wise performance of *RC-RAG* and baselines on Wikidata5M across high-, mid-, and long-tail regimes.

perform a finer-grained frequency analysis by partitioning the dataset into 10 global frequency bins. The results show that performance within both the high- and mid-frequency regimes is not uniform: the first high-frequency bin achieves the highest EM overall, while performance declines in the extreme high-frequency bins due to increased ambiguity and heterogeneous contextual evidence. This indicates that the apparent mid $>$ high pattern is primarily driven by degradation in the highest-frequency bins rather than a consistent advantage of mid-frequency instances. This suggests that the effect arises from data characteristics rather than systematic model behavior.

Second, we analyze the noise level of retrieved evidence in the high-frequency bin. A manual inspection of retrieved passages shows that high-frequency queries contain fewer passages explicitly expressing the target relation than mid-frequency queries, indicating higher retrieval noise. Detailed bin-level analysis, noise analysis, and qualitative examples are provided in Appendix F.

5.3 Ablation Study

We perform two ablation studies on Wikidata5M: (i) a component-wise ablation and (ii) a frequency-wise ablation (see Appendix E.1 for the latter).

5.3.1 Component-Wise Ablation Study

Table 4 shows a component-wise ablation on Wikidata5M using Mistral-7B-Instruct-v0.3 (chosen over Mixtral-8x7B-Instruct-v0.1 for time efficiency) under Top-10[9]. Each variant removes one component while keeping others fixed.

Ablation of Hybrid Retrieval. We analyze the contribution of hybrid retrieval by isolating the lexical and dense retrievers while keeping all other components fixed. *Lexical-only* disables dense retrieval but retains paraphrase infusion across the

[9]Top-20 shows similar trends.

Context Length	Mistral-7B-Instruct-v0.3	EM (%)	AM (%)
	Our RAG (RC-RAG)		
Top-10	Full RAG framework	63.8	**67.5**
	Ablation Study		
Top-10	Dense-only retrieval	61.7	65.2
	Lexical-only retrieval	62.6	66.0
	No relation paraphrases in any stage	61.2	64.9
	No relation paraphrases in summarization	62.7	66.0
	No relation paraphrases in retrieval	63.5	67.3
	No relation paraphrases in generation	**64.0**	**67.5**

Table 4: Component-wise ablation results for Mistral-7B-Instruct-v0.3 on Wikidata5M.

framework, whereas *dense-only* removes lexical retrieval and eliminates paraphrase-guided retrieval while preserving the use of paraphrases in later stages. Removing either retriever degrades performance compared to the full framework, confirming their complementarity. Dense-only performs worse than lexical-only, highlighting the importance of our paraphrase-guided retrieval.

Stage-wise Paraphrase-Guidance Ablations. We evaluate the role of paraphrase-guidance at each stage. To do so, we remove paraphrase-guidance from each of the three stages of *RC-RAG* individually and also perform a global ablation that removes paraphrase-guidance from all modules. Removing paraphrase guidance from retrieval, summarization, or all stages reduces performance, whereas the generation-stage ablation remains comparable to the full framework. The global ablation causes the largest degradation, highlighting the overall value of paraphrase guidance.

To examine whether these trends generalize across generators, we repeat the component-wise ablation with Llama-2-7b-chat-hf. The overall pattern remains similar; see Appendix E.2.

Question 1: *How does paraphrase guidance improve retrieval?*
To better understand why paraphrases of the target relation improve retrieval performance, we analyze retrieval outputs by comparing lexical retrieval using only the relation name with retrieval augmented with relation paraphrases. Incorporating paraphrases into lexical queries enables the retrieval of substantially more passages that explicitly express the target relation, increasing relation-aligned retrieval from 19.5% to 29.5% (+10 percentage points). Detailed analysis is provided in Appendix G. We further illustrate this effect with a qualitative comparison between Vanilla RAG and *RC-RAG* on a long-tail example in Appendix G.1.

Question 2: *Does increasing the number of relation paraphrases improve performance?*
We analyze the impact of paraphrase coverage by varying the number of relation paraphrases. On the Wikidata5M dataset with LLaMA-2-7B-chat, increasing the average number from $\sim$7 to $\sim$9 improves EM from 59.1 to 60.27. While the gain is modest, it is consistent, indicating that greater relation-specific paraphrase coverage contributes positively to performance and supports our design.

6 Conclusion

We present ***RC-RAG***, a relation-completion RAG framework that incorporates relation paraphrases across multiple stages to improve performance under sparse relational evidence. By expanding lexical coverage during hybrid retrieval and leveraging paraphrases in relation-aware summarization and generation, our approach enables more effective and robust retrieval and reasoning without requiring model fine-tuning. Experiments across multiple LLMs and benchmark datasets show that *RC-RAG* consistently outperforms strong RAG baselines and significantly improves LLM performance over standalone models, with particularly large gains in long-tail settings.

Limitations

Our evaluation covers all eight relations in MALT and twelve relations in Wikidata5M; extending the analysis to a broader and more diverse relation set remains an important direction for future work. We use deterministic decoding with temperature 0, so we do not evaluate variability under stochastic decoding. LLM-generated relation paraphrases may require additional verification for highly specialized or domain-specific relations, although the increasing availability of domain-specialized LLMs may reduce this burden by providing more appropriate domain-specific terminology. Finally, our study focuses on relation completion in English, and it remains an open question whether the observed long-tail robustness generalizes to other languages.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to

retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. *Preprint*, arXiv:2202.07646.

Lihu Chen, Simon Razniewski, and Gerhard Weikum. 2023. Knowledge base completion for long-tail entities. In *Proceedings of the First Workshop on Matching From Unstructured and Structured Data (MATCHING 2023)*, pages 99–108, Toronto, ON, Canada. Association for Computational Linguistics.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *Preprint*, arXiv:2002.08909.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *ArXiv*, abs/1506.03340.

Md Asiful Islam, Alice Saebom Kwak, Derek Bambauer, Clayton T Morrison, and Mihai Surdeanu. 2025. A framework to retrieve relevant laws for will execution. In *Proceedings of the Natural Legal Language Processing Workshop 2025*, pages 338–350, Suzhou, China. Association for Computational Linguistics.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022a. Unsupervised dense information retrieval with contrastive learning. *Preprint*, arXiv:2112.09118.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022b. Atlas: Few-shot learning with retrieval augmented language models. *Preprint*, arXiv:2208.03299.

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

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. *Preprint*, arXiv:2211.08411.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. *Preprint*, arXiv:2004.12832.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Rich James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. 2024. Radit: Retrieval-augmented dual instruction tuning. *Preprint*, arXiv:2310.01352.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *Preprint*, arXiv:2302.00083.

Benjamin Reichman and Larry Heck. 2024. Dense passage retrieval: Is it retrieving? *Preprint*, arXiv:2402.11035.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *Preprint*, arXiv:2104.07567.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. KICGPT: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683, Singapore. Association for Computational Linguistics.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2025. Exploring large language models for knowledge graph completion. *Preprint*, arXiv:2308.13916.

A Large Language Models and Implementation Details

A.1 Implementation Details: Paraphrase Generation

We use an LLM[10] to generate relation paraphrases. The model is prompted in a zero-shot manner to act as a paraphrase generator for a given target relation. Figure 2 illustrates the prompt used for paraphrase generation. Given a target relation, the prompt instructs the LLM to produce diverse linguistic expressions that convey the same relational meaning in natural text. Table 5 presents the full set of paraphrases for each relation across datasets.

Practical Considerations. The relation paraphrase generation step requires less than one minute to generate and manually verify paraphrases for each relation. The LLM effectively produces relevant paraphrases for the target relations in our experiments. We do not impose a fixed number of paraphrases per relation, as different relations naturally exhibit varying degrees of linguistic diversity. In practice, this process produces between 4 and 11 relation-specific paraphrases, with an average of 7 per relation across the two datasets. The set can be iteratively expanded by adding new paraphrases. Our experiments show that increasing paraphrase coverage leads to improved performance.

[10]We use `gpt-4o-mini-2024-07-18`.

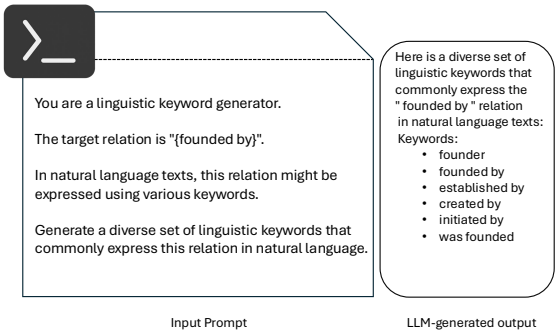


Figure 2: Example prompt used for relation paraphrase generation. The figure shows the input prompt and LLM-generated paraphrases for the `founded by` relation.

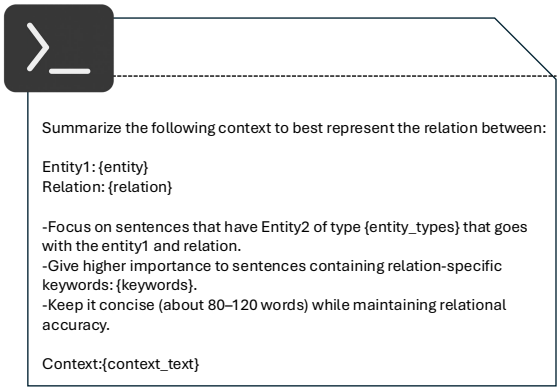


Figure 3: Prompt template used for relation-aware summarization with paraphrase guidance.

A.2 Implementation Details: Summarization Module

We use an LLM[11] to generate relation-aware evidence summaries, following the methodology described in Section 3.2. For *RC-RAG*, we use the prompt template shown in Figure 3. No additional fine-tuning is performed for the summarization model.

A.3 Implementation Details: Generation Module

In the generation module of our proposed RAG architecture (Figure 1), we perform relation completion using five different large language models: Llama-2-7b-chat-hf[12], Llama-2-13b-chat-

[11]We use `gpt-4o-mini-2024-07-18`.

[12]`https://huggingface.co/meta-llama/Llama-2-7b-chat-hf`

Relation	Relation and Its Paraphrases
founded by	founder, founded by, established by, created by, initiated by, was founded
performer	performed by, performer, singer, musician, artist, actor, actress
composer	composed by, composer, written by, music by, score by
place of birth	born, born in, birthplace, was born, place of birth
place of death	died, died in, passed away, place of death
employer	employed by, works for, worked for, job at, employer
educated at	studied at, educated at, graduated from, alma mater
residence	resides in, lives in, home in, residence, dwelling in
spouse★	spouse, married to, wife, husband, partner, wedded to, marriage to, life partner, spousal, their spouse
country of citizenship★	citizenship, citizen of, nationality, holds citizenship, country of origin, national of, belongs to, citizen status, legal nationality, national citizen, country of citizenship
shares border with★	borders, bordering, shares border, neighboring, adjacent to, common border, bordered by, touches, land border, nearby, shares border with
producer★	produced by, producer, film producer, music producer, executive producer, production by, producing, album producer, produced the, production credit

Table 5: List of all relations in our MALT and Wikidata5M datasets and their associated relation paraphrases. Relations marked with ★ denote additional relations included only in the Wikidata5M subset.

hf[13], Meta-Llama-3.1-8B-Instruct[14], Mistral-7B-Instruct-v0.3[15], and Mixtral-8x7B-Instruct-v0.1[16]. We use the same generation prompt, shown in Figure 4, across all evaluated LLMs to ensure a fair and consistent comparison. No task-specific fine-tuning is performed; all models are used in their original pretrained or instruction-tuned form. We use chat- or instruction-tuned variants for all LLMs when available, as the generation prompt requires models to follow structured instructions, such as attending to relation paraphrases and conditioning on summarized contextual evidence.

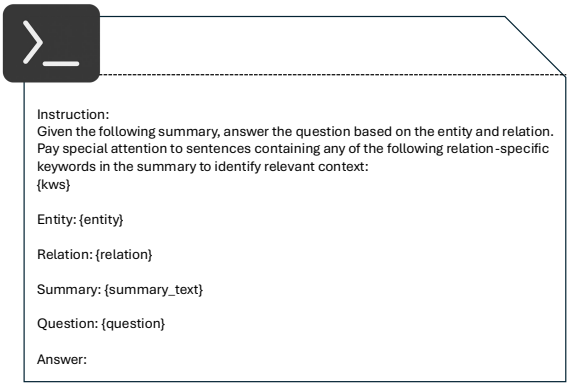


Figure 4: Prompt template used for paraphrase-guided generation. We use the same prompt for all five LLMs in our experiments. Here, `kws` refers to the relation paraphrases incorporated during inference.

A.4 LLM Hyperparameters

For relation-aware summarization, we use GPT-4o mini with a temperature of 0.0; all other decoding parameters are left at their API defaults. For relation completion in the generation stage with open-source LLMs, we use greedy decoding (no sampling) with a temperature of 0.0 and a maximum of 50 generated tokens. Input prompts are truncated to a maximum length of 2048 tokens.

A.5 Computational Cost

We measure the per-query latency of the full *RC-RAG* pipeline using LLaMA-2-7B-chat as the generation model. Retrieval takes approximately 0.78 s, summarization 0.53 s, and generation

[13] https://huggingface.co/meta-llama/Llama-2-13b-chat-hf

[14] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[15] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

[16] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

0.25 s, resulting in an end-to-end latency of approximately 1.56 s per query. These components account for approximately 50%, 34%, and 16% of the total latency, respectively. The summarization step is implemented through an external API, and its latency is therefore partly influenced by API response time.

For comparison, Vanilla RAG, which uses Contriever retrieval followed directly by generation without paraphrase guidance or summarization, requires approximately 0.45 s per query. While *RC-RAG* introduces additional inference latency, this overhead is accompanied by substantial performance gains, particularly in the long-tail setting.

When GPT-4o-mini is used for summarization, the API cost is approximately $0.00032 per query under Top-10 and $0.00050 under Top-20. This external API cost can be avoided using the fully open-source configuration described in Appendix H.

B Relation-Specific QA Templates and Entity Type Constraints

B.1 Relation-Specific Templates

Table 6 presents the list of relation-specific question–answering templates used for dense retrieval in our framework. The same templates are also used in *RC-RAG* during generation (see Figure 4 in Appendix A) and for all state-of-the-art RAG baselines (Asai et al., 2023; Xu et al., 2023) to ensure a fair comparison.

Relation	QA Template
place of birth	Where was {entity} born?
place of death	Where did {entity} die?
founded by	Who founded {entity}?
composer	Who composed {entity}?
performer	Who performed {entity}?
employer	Who is {entity} employed by?
educated at	Where did {entity} study?
residence	Where does {entity} live?
spouse★	Who is the spouse of {entity}?
country of citizenship★	What country is {entity} a citizen of?
shares border with★	Which place does {entity} share a border with?
producer★	Who is the producer of {entity}?

Table 6: Relation-specific question–answering templates. Relations marked with ★ are included only in the Wikidata5M subset.

B.2 Expected Tail Entity Type

Table 7 shows the full list of expected tail entity types used during relation-aware summarization.

Relation	Expected Tail Entity Type
place of birth	GPE, LOC
place of death	GPE, LOC
founded by	PERSON
composer	PERSON
performer	PERSON, ORG
employer	ORG
educated at	ORG
residence	GPE, LOC
spouse★	PERSON
country of citizenship★	GPE, LOC
shares border with★	GPE, LOC
producer★	PERSON

Table 7: Relation-specific expected tail entity types used to guide relation-aware summarization. Relations marked with ★ denote additional relations included only in the Wikidata5M subset.

Automatic Entity-Type Mapping. We additionally evaluate whether the expected tail entity types can be generated automatically. Using zero-shot Meta-Llama-3.1-8B-Instruct to construct the relation-to-type mapping results in only a 0.3 EM drop when Llama-2-7b-chat-hf is used as the generator, demonstrating that *RC-RAG* does not rely on manually specified entity-type mappings and that this component can be automated.

C Dataset Details

We evaluate our framework on two datasets, MALT (Chen et al., 2023) and Wikidata5M (Wang et al., 2021). For MALT, we perform evaluation on a balanced global subset without further partitioning, as the dataset is explicitly designed to emphasize long-tail relations. In contrast, for Wikidata5M, we construct frequency-based partitions to analyze model performance under varying levels of relational sparsity.

C.1 MALT

The MALT dataset[17] (Multi-token, Ambiguous, Long-Tailed facts) was developed to benchmark knowledge base completion (KBC) methods specifically for long-tail relations, which often have lim-

[17] https://www.mpi-inf.mpg.de/de/departments/databases-and-information-systems/research/knowledge-base-recall/lm4kbc/malt

ited facts in knowledge bases such as Wikidata. To construct MALT, Chen et al. (2023) selected head entities from three broad categories, `business`, `music composition`, and `human`, which frequently involve long-tail entities. The dataset includes facts from eight different relations such as `founded by`, `performer`, capturing various relation structures. The full list of relations is given in Table 5. The dataset statistically contains 87% of the sample facts with long-tail head entities, where a long-tail entity is defined as one having at most 13 triples in Wikidata. Additionally, 65.3% of the dataset contains multi-word entities, and 58.6% of the facts involve ambiguous entities. Moreover, in the MALT dataset, each tail entity tends to have multiple valid paraphrases for the same answer. For example, for the tail entity *Harvard University*, the gold annotations include multiple acceptable variations, such as *Harvard University*, *Harvard*, *Harvard Graduate School*, and *University of Harvard*, etc. Moreover, for most cases, relations such as `educated at` may have multiple correct tail entities, and the dataset explicitly accounts for this variability.

For the relation completion task, we construct a balanced subset of the original MALT dataset. To avoid bias toward or dominance by specific relations, we sample an equal number of instances per relation, with 125 examples for each relation, resulting in a total of $1,000$ triples (e_1, r, e_2).

C.2 Wikidata5M

Wikidata5M[18] is a knowledge graph (KG) benchmark dataset containing approximately 5 million entities, 20 million triples, 822 relations, and aligned entity descriptions from Wikipedia. The dataset provides both transductive and inductive evaluation settings. Since we do not perform any model training, we construct our subset from the transductive training split. We select the same eight relations included in the MALT dataset and additionally include four more relations to increase relational diversity, resulting in a total of twelve relations in our Wikidata5M subset. This relation selection is motivated by the distribution of relations in Wikidata5M, which is highly skewed, with a small number of common relations dominating the dataset. For example, the most frequent relation, `instance of`, accounts for approximately 20% of all triples, while the majority of relations individually contribute less than 1% of the data. Moreover, many relations do not contain sufficient examples to support our frequency-based partitioning across the high-, mid-, and long-tail regimes. Several relations are also broad or heterogeneous, introducing ambiguity and making relation completion evaluation less reliable. Following MALT, a long-tail knowledge base completion benchmark that focuses on eight relations, we include all eight MALT relations and add four additional relations in Wikidata5M to increase relational diversity while maintaining reliable frequency-based evaluation. The resulting subset contains 1,500 triples (e_1, r, e_2).

Frequency-Based Partitioning. We partition instances based on the co-occurrence frequency of the entity pair (e_1, e_2) within the same sentence of the same passage in the indexed RAG corpus. If an entity pair (e_1, e_2) is already rare, conditioning on the relation r further reduces its effective frequency in the corpus. Let $count(e_1, e_2)$ denote this frequency. We define three partitions: **long-tail** if $count(e_1, e_2) < x$, **mid-frequency** if $x \leq count(e_1, e_2) < 4x$, and **high-frequency** if $count(e_1, e_2) \geq 4x$, with each partition containing 500 examples.

Prior work shows that facts appearing fewer than five times in pretraining data are difficult for LLMs to extract (Carlini et al., 2023). We note that this notion of frequency differs from ours, which measures entity-pair co-occurrence in the indexed RAG corpus. We use $x = 5$ as an initial reference point and further validate this threshold empirically on our corpus, whose frequency distribution is highly skewed toward low-frequency entity pairs (Figure 5). Within each partition, we apply approximately balanced sampling across the 12 relations by randomly allocating 41 or 42 examples per relation. In the high-frequency partition, some relations (e.g., *residence*, *composer*, *educated at*, and *producer*) contain insufficient examples; in such cases, remaining instances are randomly sampled from other relations to maintain 500 examples per partition. This partitioning enables a fine-grained analysis of how RAG methods behave as relational evidence becomes increasingly sparse.

Empirical Analysis for Long-Tail Threshold Selection While selecting the threshold value, we conducted an additional analysis of the entity-pair co-occurrence frequencies in the indexed corpus used for partitioning. The analysis shows (shown in Figure 5) that the distribution is highly skewed toward low-frequency entity pairs, with many oc-

[18] https://deepgraphlearning.github.io/project/wikidata5m

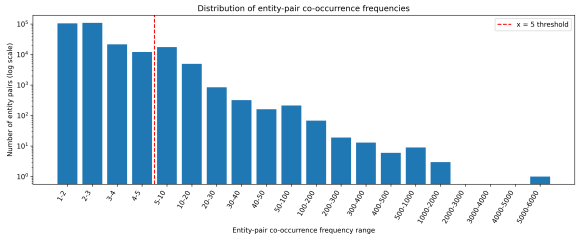


Figure 5: Histogram of entity-pair co-occurrence frequencies in the indexed RAG corpus.

curring fewer than five times and the number of instances steadily decreasing at higher frequencies. This confirms the long-tailed nature of the data and provides additional empirical motivation for our choice of the threshold. In particular, most examples lie in a data-sparse regime, and conditioning on a relation further reduces effective frequency. Therefore, using $x = 5$ provides an intuitive boundary separating extremely sparse instances from more frequent ones. This choice is also consistent with prior findings showing that facts appearing fewer than 5 times during training are difficult for LLMs to extract.

D Details of Baseline RAG Methods

D.1 SELF-RAG

We include SELF-RAG (Asai et al., 2023) as a strong state-of-the-art RAG baseline. SELF-RAG integrates retrieval, generation, and self-critique through extensive model training by extending a pretrained language model (LM) with reflection tokens, which are learned as part of standard next-token generation. The training framework follows a multi-stage process. First, a separate critic model based on LLaMA-2-7B is trained using supervision generated by prompting GPT-4 to annotate retrieval decisions and critique signals, which are then distilled into an in-house model. This critic is used offline to insert reflection tokens into training examples, enabling the generator to learn retrieval control, passage relevance assessment, factual grounding, and critique of intermediate generations. Subsequently, the final generator LMs (LLaMA-2-7B or LLaMA-2-13B) are trained on over 150K instruction-following instances augmented with retrieved passages and reflection tokens, using a conventional language modeling objective together with reinforcement learning-based optimization. This allows the generator to autonomously produce reflection tokens at inference time without relying on the critic. At inference time, SELF-RAG performs multi-stage retrieval, parallel passage processing, and critique-guided decoding with segment-level beam search. For retrieval, SELF-RAG uses the off-the-shelf Contriever-MS MARCO retriever (Izacard et al., 2022a) and retrieves up to ten passages per query; we use the same retriever configuration in our experiments for a fair comparison. We evaluate both released SELF-RAG variants, based on LLaMA-2-7B and LLaMA-2-13B, on both datasets. We use the 7B variant in the main results and additionally evaluate the 13B variant in the model-size comparison.

D.1.1 Model-Size Comparison

To examine whether the performance differences between SELF-RAG and *RC-RAG* are attributable to generator size, we additionally evaluate the released SELF-RAG LLaMA-2-13B variant. We then compare both methods at the 7B and 13B scales under the Top-10 setting. The results on Wikidata5M and MALT are reported in Tables 8 and 9, respectively.

Increasing the SELF-RAG model size from 7B to 13B improves SELF-RAG performance on both datasets, but *RC-RAG* strongly outperforms SELF-RAG-13B overall. On Wikidata5M, *RC-RAG* with LLaMA-2-13B achieves 61.7 EM compared with 57.3 for SELF-RAG-13B, with particularly large gains in the mid-frequency and long-tail subsets. Even *RC-RAG* with LLaMA-2-7B outperforms SELF-RAG-13B in these two subsets. On MALT, *RC-RAG* with LLaMA-2-7B also exceeds SELF-RAG-13B (58.6 vs. 55.8 EM). These results indicate that the gains of *RC-RAG* are not explained by model size alone.

D.2 RECOMP

We include RECOMP (Xu et al., 2023) as another RAG baseline that improves generation by compressing retrieved passages before they are consumed by the generator. The method proposes two compression variants: an extractive compressor that selects salient spans from retrieved passages and an abstractive compressor that generates query-conditioned summaries. In our experiments, we use the abstractive summarization variant of RECOMP as a baseline, since our proposed framework also incorporates a summarization stage, enabling a fair

Model	Overall		High-frequency		Mid-frequency		Long-tail	
	EM	AM	EM	AM	EM	AM	EM	AM
SELF-RAG LLaMA-2-7B	52.1	55.3	60.8	63.6	52.6	54.8	43.0	47.6
SELF-RAG LLaMA-2-13B	57.3	60.3	**64.8**	**68.0**	59.2	61.4	48.0	51.4
RC-RAG LLaMA-2-7B	59.1	62.4	60.8	64.0	63.8	65.6	52.8	57.6
RC-RAG LLaMA-2-13B	**61.7**	**65.0**	63.8	67.0	**66.8**	**69.0**	**54.4**	**59.0**

Table 8: Performance comparison of SELF-RAG and *RC-RAG* with 7B and 13B LLaMA-2 variants on Wikidata5M under Top-10.

Model	EM	AM
SELF-RAG LLaMA-2-7B	54.2	57.3
SELF-RAG LLaMA-2-13B	55.8	59.1
RC-RAG LLaMA-2-7B	58.6	60.9
RC-RAG LLaMA-2-13B	**58.9**	**61.1**

Table 9: Performance comparison of SELF-RAG and *RC-RAG* with 7B and 13B LLaMA-2 variants on MALT under Top-10.

comparison.

RECOMP first retrieves passages using a dense retriever; for fair comparison, we use the Contriever-MS MARCO retriever (Izacard et al., 2022a), consistent with SELF-RAG and our dense retrieval module. The abstractive compressor is implemented as an encoder—decoder sequence-to-sequence model initialized from T5 and pre-trained on a summarization dataset (Hermann et al., 2015). Its query-focused summarization ability is distilled from an extreme-scale language model by generating a large synthetic training dataset, filtering the generated summaries for quality, and training the encoder–decoder model on the filtered data. Training instances are constructed by pairing each retrieved passage with the input query and supervising the compressor to generate summaries that preserve answer-relevant evidence while discarding irrelevant content. The compressed summaries from multiple retrieved passages are concatenated and provided as context to the generator, reducing noise and effective context length. The compressor and generator are trained jointly using supervised learning.

E Additional Ablation Analyses

E.1 Frequency-Wise Ablation Study

Frequency-Wise Ablation Analysis on Wikidata5M. Table 10 analyzes the contribution of individual components across three frequency partitions. The full RAG framework consistently achieves strong performance across all regimes. Notably, in the long-tail setting, the lexical-only retrieval variant remains competitive even without dense retrieval, highlighting the effectiveness of relation paraphrase guidance in both retrieval and downstream stages. In contrast, dense-only retrieval and removing paraphrase guidance from summarization result in clear performance degradation in the long-tail regime.

Interestingly, removing paraphrase guidance at the generation stage slightly improves performance in some cases by allowing the model to output multiple valid answers derived from the relation-aware summaries, particularly for relations that naturally support multiple correct entities (e.g., *shares border with*, *educated at*, *producer*). In contrast, paraphrase guidance during generation encourages the model to produce a single, specific answer. Unlike MALT, Wikidata5M contains incomplete gold annotations for such relations; consequently, correct but non-matching predictions are penalized. Removing paraphrase guidance during generation, which allows multiple answers to be produced, therefore increases the likelihood of matching the annotated gold labels, explaining the observed effect.

E.2 Cross-Model Component Ablation

To examine whether the component-wise ablation trends generalize across generators, we repeat the analysis using Llama-2-7b-chat-hf. As shown in Table 11, the overall trends are largely consistent with the Mistral results, while some stage-specific variation is observed across models.

F Detailed Analysis of Mid- vs High-Frequency Performance.

To better understand this phenomenon, we conduct a finer-grained analysis by dividing the Wikidata5M dataset into 10 equal-sized global bins sorted by entity-pair frequency. Examining the high-frequency bucket at this granularity reveals that bin 7 (shown in Figure 6) achieves the high-

Context Length	(Mistral-7B-Instruct-v0.3)	Frequency					
		High-frequency		Mid-frequency		Long-tail	
		EM (%)	AM (%)	EM (%)	AM (%)	EM (%)	AM (%)
Our RAG (RC-RAG)							
Top-10	Full RAG framework	**67.4**	**71.2**	67.8	69.8	56.2	61.4
Ablation Study							
Top-10	Lexical-only retrieval	66.0	68.6	65.2	68.4	**56.6**	61.0
	Dense-only retrieval	65.8	69.6	65.2	68.2	54.2	57.8
	No relation paraphrases in retrieval	66.2	70.2	67.8	70.0	56.6	**61.8**
	No relation paraphrases in summarization	66.4	69.6	65.6	67.6	56.0	60.8
	No relation paraphrases in generation	66.8	70.4	**69.0**	**71.0**	56.2	61.2
	No relation paraphrases in any stage	63.6	67.4	65.4	67.8	54.8	59.4

Table 10: Ablation study analyzing the contribution of individual components of our RAG framework on the Wikidata5M dataset, using Mistral-7B-Instruct-v0.3 as the underlying language model. Results are reported across high-frequency, mid-frequency, and long-tail partitions, highlighting the impact of our multi-stage paraphrase-guided RAG framework.

Configuration	**EM**	**AM**
Full RC-RAG Framework	59.1	62.4
Dense-only retrieval	57.6	60.6
Lexical-only retrieval	58.1	61.5
No relation paraphrases in any stage	56.8	59.7
No relation paraphrases in summarization	60.7	63.6
No relation paraphrases in retrieval	58.6	61.8
No relation paraphrases in generation	59.1	62.4

Table 11: Component-wise ablation results with Llama-2-7b-chat-hf on Wikidata5M under Top-10.

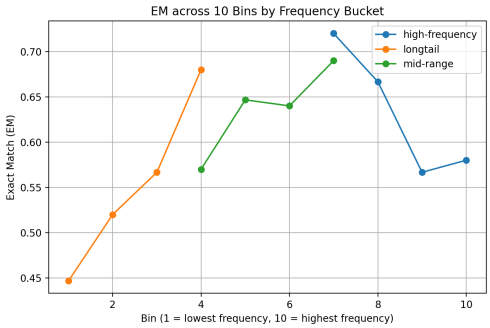


Figure 6: Exact Match (EM) across 10 global frequency bins on Wikidata5M.

est EM (0.72), outperforming all mid-frequency bins, while bin 8 (0.67) performs comparably to the strongest mid-frequency bin (0.69). However, performance drops substantially in bins 9 and 10.

These findings indicate that the apparent mid $>$ high pattern is not due to a consistent advantage of mid-frequency instances or uniformly poor performance in the high-frequency bucket. Instead, performance varies across subranges, with degradation occurring primarily in the highest-frequency bins. This downward trend at extreme frequency levels is likely related to increased entity ambiguity and heterogeneous contextual evidence. Highly frequent entities tend to appear in more diverse contexts, often involving multiple plausible candidate entities, surface-form variations, and polysemous usages. As a result, although such entities have abundant evidence, the effective signal-to-noise ratio may decrease, leading to lower prediction accuracy.

Noise Level Analysis in the High-Frequency Bin. The observed performance drop in the highest-frequency bins further motivates an analysis comparing the noise level of retrieved passages in high- vs. mid-frequency buckets. To investigate this phenomenon, we manually analyze 20 instances from each bucket and count how many of the top-10 retrieved passages explicitly express the target relation. On average, mid-frequency instances contain 2.65 relevant passages per query, whereas high-frequency instances contain only 1.9. We define retrieval noise as the proportion of the top-10 retrieved passages that do not explicitly express the target relation (i.e., $1 - \text{relevant_passages}/10$). This corresponds to an estimated noise rate of 73.5% for mid-frequency instances and 81% for high-frequency instances.

A closer inspection reveals several factors contributing to the higher noise in the high-frequency bucket. First, entity ambiguity is more common among frequent entities. For example, in the high-

frequency bin we observe ambiguous names such as *Storni*, which may refer to either *Alfonsina Storni* or *Segundo Rosa Storni*. Retrieved passages often contain evidence for both entities, and the model tends to select the more dominant or frequently mentioned one. In such cases, the prediction is not necessarily factually incorrect but reflects name ambiguity. Similarly, instances such as *Agis II* or *Mustafa II* illustrate how retrieval may surface passages about closely related or more prominent entities (e.g., *Mustafa I*), leading the model to favor a more popular or ambiguous variant that does not exactly match the gold entity.

Second, surface-form variations introduce additional mismatches (e.g., *W E B Du Bois* vs. *W. E. B. Du Bois* or *chi lites* vs. *Chi-Lites*), which may reduce exact lexical alignment despite semantic equivalence. Third, some entities exhibit strong polysemy across domains. For example, *Tete* may refer to a song, a footballer, or a rodent species; highly frequent entities with multiple meanings tend to retrieve passages about the more popular interpretation rather than the gold target.

These observations suggest that high-frequency entities often retrieve abundant but heterogeneous information, increasing ambiguity and distractor signals. As a result, although such entities are frequent in the corpus, the effective signal-to-noise ratio for the target relation can be lower than in mid-frequency instances, helping explain the observed mid $>$ high pattern.

G Why Infusing Relation Paraphrases Improves Retrieval

To better understand why lexical variants of the target relation improve retrieval performance, we analyze the retrieved passages under two lexical retrieval settings: (1) queries using only the canonical relation name and (2) queries augmented with relation paraphrases. We randomly sample 20 instances and examine the top-10 retrieved passages for each instance, resulting in 200 passages in total. For each passage, we manually determine whether it explicitly expresses the target relation.

Across the 20 sampled instances, relation-only lexical retrieval retrieves 39 passages (19.5%; 1.95 per instance on average) that explicitly express the target relation. In contrast, lexical retrieval using relation paraphrases retrieves 59 such passages (29.5%; 2.95 per instance on average), corresponding to a +10 percentage point increase in relation-aligned retrieval.

This improvement suggests that many relations are expressed in natural text through diverse linguistic forms that do not necessarily contain the canonical relation label. For example, relations such as `performer` or `employer` are often conveyed through verbs or prepositional phrases (e.g., `performed by`, `worked at`, `served as`), which may not be captured when querying only the relation name. By incorporating relation paraphrases, lexical retrieval better aligns with these natural language expressions, resulting in a higher proportion of retrieved passages that explicitly express the target relation.

G.1 Qualitative Case Study of Paraphrase-Guided Retrieval

To provide a concrete example of how paraphrase-guided query expansion improves retrieval in long-tail cases, we compare *RC-RAG* with Vanilla RAG. We consider a representative example with the entity *I Remember Me* (a song) and the relation `producer`, whose gold answer is *Ryan Tedder*. We examine the retrieved passages produced by *RC-RAG* and compare them with those from Vanilla RAG, which uses Contriever retrieval followed directly by LLM generation without summarization or paraphrase guidance. In Vanilla RAG, the dense retriever is distracted by lexically similar but irrelevant titles such as *Remember Me (TV series)* and *Remember Me (video game)*, leading the generator to predict the incorrect answer *Chris Carey*.

In contrast, *RC-RAG* expands the query with relation-aware cues such as *music producer*, *production credit*, and *album producer*, which surface the correct passage for *I Remember Me (song)*. The retrieved evidence explicitly states that the song was written by Ryan Tedder, who also produced the track, allowing *RC-RAG* to recover the correct answer *Ryan Tedder*. This example illustrates how relation-aware query expansion improves evidence retrieval and supports more reliable downstream generation in the long-tail setting.

H Open-Source *RC-RAG* Variants

To evaluate whether *RC-RAG* depends on proprietary LLMs, we conduct additional experiments on Wikidata5M using LLaMA-3.1-8B-Instruct. We first use LLaMA-3.1-8B-Instruct for paraphrase generation, QA-template generation, and entity-type mapping while retaining GPT-4o-mini for

Configuration				Overall		High-frequency		Mid-frequency		Long-tail	
Method	**Paraphrase/Template**	**Type Mapping**	**Summarizer**	**EM**	**AM**	**EM**	**AM**	**EM**	**AM**	**EM**	**AM**
RECOMP	–	–	–	47.5	50.9	52.2	56.6	49.4	51.4	40.8	44.8
SELF-RAG	–	–	–	52.1	55.3	60.8	63.6	52.6	54.8	43.0	47.6
RC-RAG	LLaMA-3.1-8B	LLaMA-3.1-8B	LLaMA-3.1-8B	55.2	58.1	57.8	60.8	57.2	59.4	50.6	54.2
RC-RAG	LLaMA-3.1-8B	LLaMA-3.1-8B	GPT-4o-mini	55.6	59.1	58.2	61.8	60.6	63.2	48.0	52.2
RC-RAG	GPT-4o-mini	Manual	GPT-4o-mini	59.1	62.4	60.8	64.0	63.8	65.6	52.8	57.6

Table 12: Overall and frequency-wise performance of *RC-RAG* and RAG baselines on Wikidata5M under the Top-10 setting. For *RC-RAG*, we report the models used for paraphrase and QA-template generation and summarization, along with the entity-type mapping strategy. Across these variants, *RC-RAG* maintains strong overall and long-tail performance, demonstrating that the framework is not dependent on a specific LLM.

Context	Model	c_{rrf}	EM (%)	AM (%)
Top-10	Llama-2-7b-chat-hf	0	**58.6**	**60.9**
		10	57.9	59.1
		20	58.2	60.2
		30	57.8	59.9
		40	57.8	60.0
		50	57.8	59.9
		60	57.8	59.9

Table 13: Performance of *RC-RAG* with different RRF constants on the MALT dataset.

summarization. We then evaluate a fully open-source pipeline in which LLaMA-3.1-8B-Instruct is used for all four components, including summarization.

Table 12 shows that even the fully open-source pipeline outperforms RECOMP and SELF-RAG overall and retains strong performance in the long-tail setting. These results show that *RC-RAG* remains effective across different choices of LLMs and does not depend on GPT-4o-mini or another proprietary model, supporting the model-agnostic nature of the framework.

I Analysis of the RRF Constant

To justify our choice of $c_{\text{rrf}} = 0$, we analyze the impact of the RRF constant c_{rrf} on the MALT dataset, with results reported in Table 13. The results show that setting $c_{\text{rrf}} = 0$ yields the best performance (EM: 58.6, AM: 60.9), while increasing c_{rrf} to higher values, including the standard setting of $c_{\text{rrf}} = 60$, keeps performance stable, albeit marginally lower. Therefore, we use $c_{\text{rrf}} = 0$ in our experiments to prioritize top-ranked relevant passages.