Don't Lose Entities from Retrieval to Generation: Dual Entity Recovery RAG for multi-hop QA

Anonymous ACL submission

Abstract

Retrieval-augmented multi-hop question answering (QA) decomposes a query into sub-questions and decomposes the corpus into smaller retrieval units such as sentences. Both forms of decomposition improve the pipeline, but we show that both share the same vulnerability, the loss of entity information, and that this loss breaks the pipeline at two separate points. The first point is retrieval, where a sub-question loses the entity resolved at the previous hop, leaving the retriever with nothing to match against. The second point is harder to see, because retrieval still appears to succeed. Once a passage is split into sentences, an isolated sentence loses the context that grounds its pronouns, so even with the correct sentence in hand the LLM cannot tell which entity the sentence is about. We isolate this second point as a distinct failure mode that we call lost-in-generation, and a retrieval-controlled experiment shows that it degrades answers even when the gold evidence is fixed in the context. We then propose Dual Entity Recovery RAG (DER-RAG), which keeps the grounding entity explicit from retrieval through to generation with two lightweight components, a two-way query decomposition that carries the resolved entity across sub-questions and a subject entity prefix attached to each sentence at generation time. DER-RAG needs no graph construction, no corpus modification, and no fine-tuning, yet on three multi-hop QA benchmarks it matches or exceeds strong baselines, including graph-based methods that depend on costly offline structures.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) lets large language models (LLMs) answer knowledge-intensive questions by drawing on external corpora beyond their parametric knowledge. The setting is especially demanding for multi-hop QA (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022), where the evidence is scattered across documents, and RAG systems commonly address this with two forms of decomposition: a complex query is decomposed into a chain of sub-questions, and the corpus is decomposed into finer-grained retrieval units such as sentences.

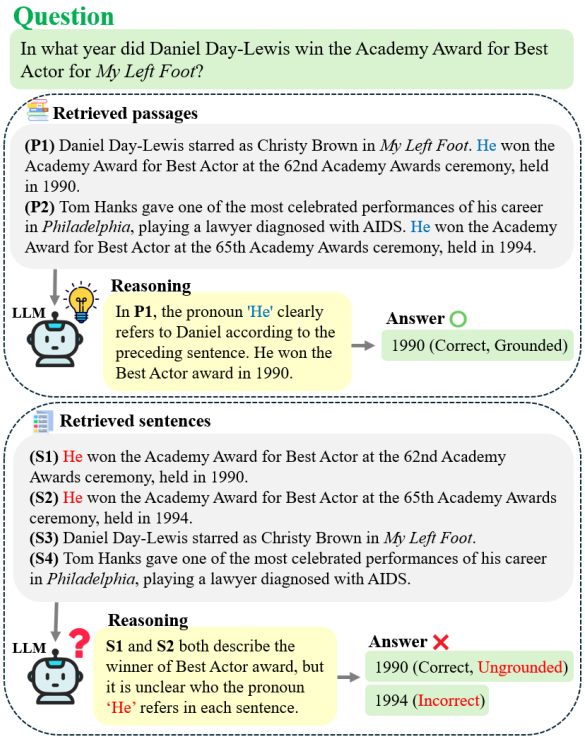

Figure 1: The lost-in-generation problem. Even when correct evidence is retrieved, sentence splitting leaves pronouns unresolvable, causing incorrect generation.

We observe that both forms of decomposition share an underlying vulnerability, namely the loss of **entity** information. Each decomposition step can remove a key entity that the surrounding context had made explicit, and this loss causes failures at two different stages of the RAG pipeline.

The first stage is retrieval. When a multi-hop question is broken into sub-questions, a later sub-question often loses the entity resolved in the preceding step and refers to it only through a vague

Sentence	He won the Academy Award for Best Actor at the 62nd Academy Awards, held in 1990.
Prefixed-sentence	Daniel Day-Lewis: He won the Academy Award for Best Actor at the 62nd Academy Awards, held in 1990.

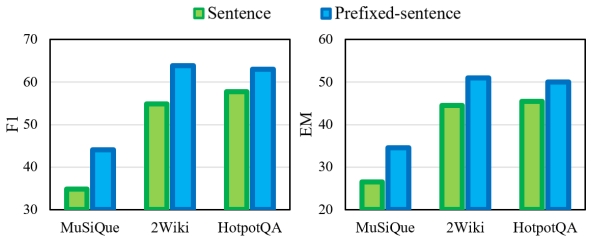

Figure 2: (*Top*) Example of Sentence and Prefixed-sentence. Daniel Day-Lewis is the subject entity prefix that resolves pronoun ambiguity. (*Bottom*) Generation performance of Sentence vs. Prefixed-sentence given the same retrieved sentence.

pronoun or a demonstrative, which degrades retrieval and disrupts the reasoning chain. This problem was identified by ChainRAG (Zhu et al., 2025) as *lost-in-retrieval* and addressed with a post-hoc step that rewrites a sub-question to recover the missing entity. ChainRAG reports an average entity recovery accuracy of 79.3%, which leaves clear room for improvement and motivates us to revisit the problem from a generation-time perspective.

The second stage is generation, and it is the failure mode we isolate. Splitting a passage into sentences improves retrieval granularity, but it also removes the context that disambiguates a sentence's pronouns. As illustrated in Figure 1, once a passage is split into individual sentences, a sentence such as "*He* won the Academy Award for Best Actor at the 62nd Academy Awards ceremony, held in 1990" becomes ambiguous, and the LLM can no longer tell who "He" refers to. This phenomenon has been studied as *decontextualization* by Choi et al. (2021), who address it by rewriting and reindexing the corpus. Our contribution is to isolate it as a distinct failure mode within the generation step of RAG, which we call *lost-in-generation*, where it persists after retrieval has succeeded and is invisible to retrieval-side metrics. It exposes an assumption RAG pipelines often make implicitly, that retrieving the correct evidence is enough, since a correctly retrieved sentence is not always usable and retrieval correctness does not ensure the evidence stays grounded once the LLM reads it.

This gap is not hypothetical. We run a controlled experiment in which the retrieved sentences are held fixed and the only variable is whether each sentence carries a subject entity prefix. Prepending the subject entity of the source passage improves exact match by +9.24, +9.08, and +5.18 on MuSiQue, 2Wiki, and HotpotQA (Figure 2), and since the evidence is identical in both conditions the gain isolates a generation-side failure rather than a retrieval one. We develop this into a full controlled study in Section 4.3.

Motivated by these findings, we propose **Dual Entity Recovery RAG (DER-RAG)**, a RAG framework built on NaiveRAG with both decomposition strategies and organized around a single principle, that the entity grounding a piece of evidence should remain explicit from retrieval through to generation. DER-RAG addresses the two stages of entity loss with one lightweight component each, requiring no graph construction, no corpus modification, and no fine-tuning. For lost-in-retrieval, a two-way query decomposition generates each sub-question so that the entity it depends on is already present, which removes the need for a post-hoc rewriting step. For lost-in-generation, an LLM extracts the subject entity of each passage offline, and this entity is prefixed to each retrieved sentence at generation stage. On MuSiQue, 2WikiMultiHopQA, and HotpotQA, DER-RAG matches or exceeds strong baselines on average, including graph-based methods that rely on offline graph construction. Our main contributions are as follows:

- We connect two entity-loss problems that prior work studied separately, lost-in-retrieval in the retrieval step and the loss of pronoun grounding in the generation step, and isolate **lost-in-generation** as a distinct failure mode where corpus decomposition discards the entity that grounds a sentence.
- We propose **DER-RAG**, a lightweight framework that addresses both forms of entity loss without graph construction, corpus modification or fine-tuning.
- We evaluate DER-RAG on three multi-hop QA benchmarks, where it matches or exceeds strong baselines including graph-based RAGs, with a controlled experiment and ablations isolating each component's contribution.

2 Related Work

Retrieval-augmented generation and multi-hop QA. RAG (Lewis et al., 2020) grounds LLMs in external knowledge without parameter updates, and a broad line of work improves it through iterative

retrieval (Shao et al., 2023; Trivedi et al., 2023), reranking (Glass et al., 2022), and context compression (Wu et al., 2025). Multi-hop QA (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022) is difficult because the evidence is scattered across documents, and RAG systems commonly address this with two forms of decomposition. On the query side, a complex question is decomposed into sub-questions (Perez et al., 2020; Khot et al., 2022; Press et al., 2023), and methods such as IRCoT (Trivedi et al., 2023) interleave chain-of-thought reasoning with retrieval at each step. On the corpus side, the retrieval unit ranges from passages (Karpukhin et al., 2020) to sentences (Zhu et al., 2025) to long grouped units (Jiang et al., 2024). A prominent recent direction pursues multi-hop performance through elaborate offline structures: graph-based methods organise the corpus into hierarchical trees (Sarthi et al., 2024) or entity graphs (Edge et al., 2024; Gutiérrez et al., 2024; Li et al., 2024) and retrieve by traversing them, which is effective but requires graph construction whose cost grows with corpus size. ChainRAG (Zhu et al., 2025) takes a different view, identifying the lost-in-retrieval problem in which entity underspecification in a sub-question causes retrieval to fail regardless of how elaborate the downstream pipeline is. DER-RAG follows this view, asking where entity information is lost along the pipeline rather than where more structure can be added.

Decomposition depth and termination. A multi-step decomposition must decide how many sub-questions to generate. Much prior work on these benchmarks assumes two-hop questions and decomposes accordingly (Zhu et al., 2025), which is simple but does not generalise to other reasoning depths. DeepRAG (Guan et al., 2025) instead decides at each step whether to continue, and because this decision is hard to make from the query alone, it learns a termination policy through fine-tuning. DER-RAG occupies neither end: it predicts the decomposition depth from the question with a single prompted LLM call, which keeps the procedure general over reasoning depth while requiring no fine-tuning and no training data.

Retrieval unit and decontextualization. The granularity of the retrieval unit is a central design choice in RAG, and sentence-level retrieval improves precision over passage-level retrieval (Chen et al., 2024b) while being widely adopted (Zhu et al., 2025). A sentence taken out of its passage, however, may no longer be interpretable on its own, since its pronouns and definite descriptions lose their antecedents. Choi et al. (2021) study this phenomenon as *decontextualization*, rewriting a sentence to be interpretable out of context and re-indexing a corpus of such sentences. This line of work treats interpretability as a static property of a sentence, fixed once at indexing time. DER-RAG instead views referential failure as something a RAG pipeline produces dynamically, in that decomposition breaks the grounding at query time, after retrieval has already selected the evidence. We therefore address the same phenomenon in a different place and by a different means, identifying it as a distinct failure mode in the generation step of RAG, where it persists after retrieval has succeeded and is invisible to retrieval-side metrics, and we resolve it without rewriting or re-indexing the corpus. Rather than editing the indexed text, DER-RAG prepends the subject entity of the source passage at generation stage, which leaves the retrieval index unchanged and adds only a single LLM call per passage offline.

3 DER-RAG

Multi-hop RAG systems rely on two complementary forms of decomposition, in that a complex query is split into a chain of sub-questions and the corpus is split into fine-grained retrieval units such as sentences. Both forms help the pipeline, since decomposition simplifies each retrieval step and finer units improve retrieval precision, but both also share one side effect, in that each decomposition step can strip away the entity information the surrounding context made explicit. DER-RAG is built on this standard pipeline, NaiveRAG with both decomposition strategies, and is organized around a single principle, that the entity grounding a piece of evidence should remain explicit from retrieval through to generation.

Entity information is lost at two stages. The first is retrieval, where a later sub-question omits the entity resolved at the previous hop and refers to it only through a pronoun, leaving the retriever with no entity to match against (Section 3.1). The second is generation, where a sentence separated from its passage loses the context that licenses its pronouns, so the LLM cannot attribute the sentence to the correct entity even when it has been retrieved correctly (Section 3.2).

DER-RAG applies this principle with one

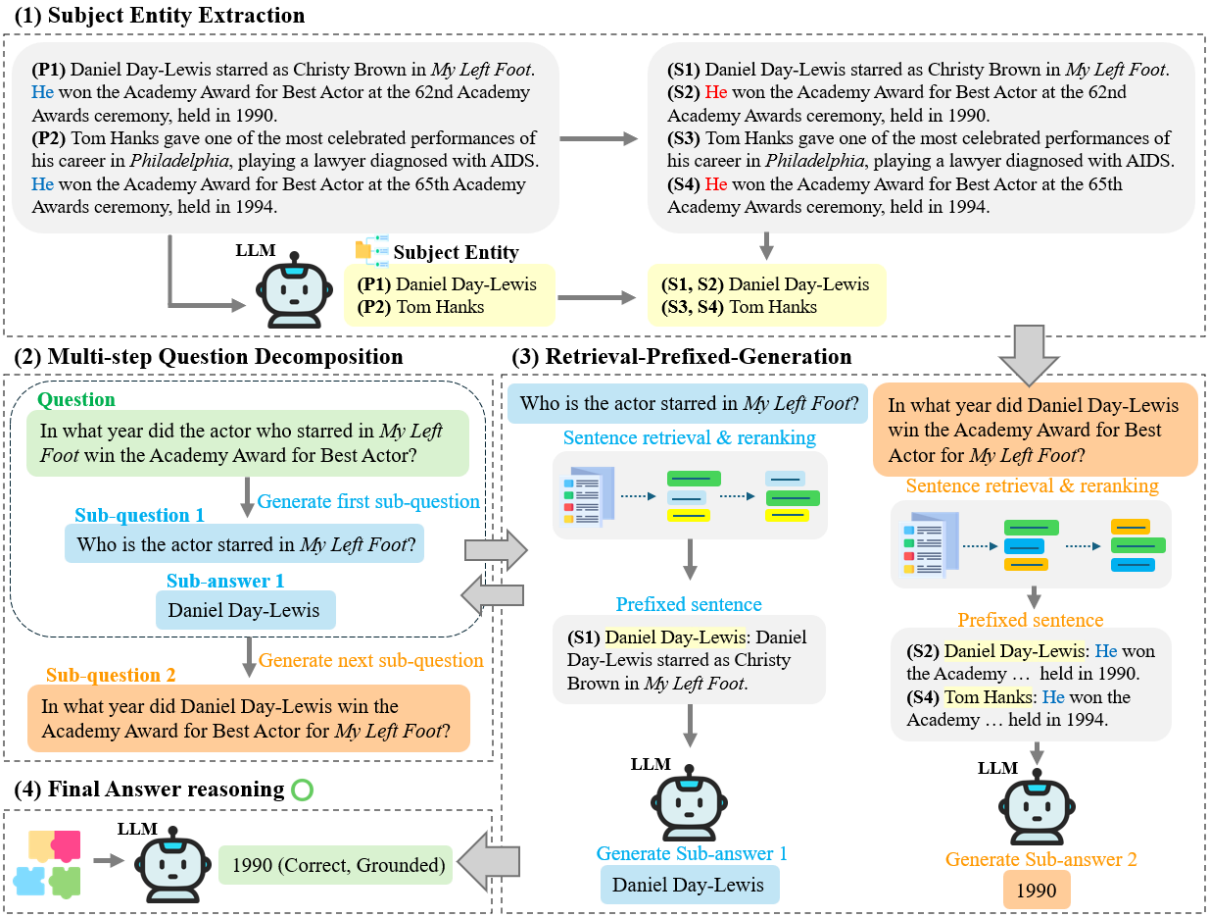

Figure 3: Overview of the DER-RAG pipeline. (1) **Two-way query decomposition**: for non-comparison questions, sub-questions are generated sequentially, each conditioned on the answer to the preceding step. (2) **Prefixed-sentence generation**: for each sub-question, supporting sentences are retrieved and prefixed with the subject entity of their source passage before being passed to the LLM. (3) **Final answer reasoning**: the LLM aggregates prefixed sentences and sub-answers to produce the final answer.

lightweight component at each stage. At the query side, two-way query decomposition generates each sub-question so that the entity it depends on is already present, removing the need for a separate post-hoc rewriting step. At the generation side, prefixed-sentence generation restores the entity that sentence splitting discarded, attaching the subject entity of the source passage to each retrieved sentence before the LLM reads it. The two components are two instantiations of the same principle.

This design is lightweight by construction. Neither component requires graph construction, corpus modification or fine-tuning, and the only offline cost is a single LLM call per passage to extract its subject entity, which treats each passage independently and scales linearly with corpus size. Figure 3 illustrates the full pipeline.

3.1 Two-Way Query Decomposition

A sub-question generated during query decomposition can fail to retrieve relevant passages whenever it lacks the intermediate entity resolved at the previous hop. DER-RAG addresses this by routing each multi-hop question to one of two decomposition paths, so that the intermediate entity is always explicit in the sub-question that needs it.

Question type classification. Given a multi-hop question q, DER-RAG first classifies it as *comparison* or *non-comparison* using a single LLM call that returns a binary label. A question is labeled comparison when it explicitly names two or more entities and asks them to be evaluated against a shared criterion, as in "Which came first, X or Y?" or "Who is older, A or B?", while all other questions, including bridge and compositional questions, are labeled non-comparison. This distinction is highly reliable in practice (Appendix B), so classification

error contributes negligibly to the pipeline, and the classification prompt is given in Appendix I.

For comparison questions, both targets are already named in the original question, so DER-RAG generates all sub-questions in a single LLM call and each sub-question inherits a named entity directly from q. No sub-question therefore depends on an entity that has yet to be resolved, and this path also avoids the sequential cost of multi-step decomposition. For non-comparison questions, including bridge and compositional questions, the entity needed to anchor a later sub-question is not known until the preceding sub-question has been answered, so DER-RAG generates sub-questions one at a time. At step t, the LLM receives the original question q together with the history of previously generated sub-questions and their retrieved answers $\{(q_1, a_1), \ldots, (q_{t-1}, a_{t-1})\}$, and from these it generates the next sub-question q_t. Because q_t is generated after a_{t-1} has been retrieved, the intermediate entity resolved by a_{t-1} is available, and DER-RAG instructs the LLM to substitute this entity directly into q_t instead of referring to it with a pronoun. The decomposition prompts are given in Appendix I.

Termination. A multi-step decomposition must decide how many sub-questions to generate. Chain-RAG (Zhu et al., 2025) and much prior work assume two-hop questions, while DeepRAG (Guan et al., 2025) learns a termination policy through fine-tuning, at the cost of training data and additional compute. DER-RAG instead keeps this decision lightweight and train-free, decomposing questions of arbitrary reasoning depth without a learned policy. After the question type is classified, the LLM predicts, conditioned on the original question, how many sub-questions the decomposition will require, and because this prediction comes through a prompt instruction it needs no training data. We report the predicted-count distribution in Appendix E, and the strong results in Section 4.2 indicate that this simple predictor is enough to support accurate multi-hop reasoning.

3.2 Prefixed-Sentence Generation

Sentence-level retrieval improves precision, but a sentence read in isolation often loses the entity that its pronouns and definite descriptions refer to, so the LLM may fail to attribute a retrieved fact to the correct entity even when the sentence itself is relevant and correctly retrieved. DER-RAG resolves this by prefixing each retrieved sentence with the subject entity of its source passage before the sentence is passed to the LLM.

Subject entity. For each passage c in the corpus, we define its *subject entity* e_c as the central topic entity that the passage primarily describes. We obtain e_c with a single LLM call per passage, using the information-extraction prompt in Figure 6, which asks the LLM to return the one entity the passage is primarily about as a concise noun phrase. This step treats each passage independently, so it is trivially parallelizable. It is run once, offline, and cached as passage metadata, so it adds no cost to online inference, and the resulting e_c is shared by every sentence split from c.

When a corpus provides document titles, the title can be used directly as e_c. Titles are not always available, however, and even when present they need not coincide with the passage's true subject, since a passage titled after a single album in a discography is primarily about the artist rather than the album. LLM extraction therefore serves as the general mechanism, with titles as a readily available substitute. All DER-RAG experiments use extracted subject entities, and we compare against title-based prefixes in Appendix F.

Prefixing. For each sub-question q_t, DER-RAG retrieves the top-k sentences from the indexed corpus using dense retrieval followed by cross-encoder reranking, and retrieval operates on the original, unmodified sentences. At generation stage, each retrieved sentence s_i originating from passage c is prefixed with the subject entity e_c of that passage to form

$$\tilde{s}_i = e_c : s_i.$$

The prefixed context $[\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_k]$ is then passed to the LLM to generate the sub-answer a_t. Because each sentence now carries an explicit statement of the entity it describes, the LLM can attribute retrieved facts to the correct entity rather than to an unresolved pronoun. This intervention operates only on the generation stage, so its effect is in principle independent of retrieval quality, and we isolate this effect with a retrieval-controlled experiment in Section 4.3 while also ablating the prefix within the full pipeline in Section 4.4.

Because the prefix is added at generation stage, the retrieval index is left unchanged, so DER-RAG can be applied to an existing sentence-level RAG system as a drop-in modification.

LLMs	Methods	MuSiQue		2Wiki		HotpotQA	
		F1	EM	F1	EM	F1	EM
GPT-4o-mini	Direct	$16.63_{\pm0.62}$	$10.33_{\pm0.29}$	$32.07_{\pm1.15}$	$26.50_{\pm1.00}$	$39.37_{\pm0.75}$	$29.50_{\pm0.50}$
	NaiveRAG (passage)	$35.30_{\pm0.42}$	$28.83_{\pm0.76}$	$51.99_{\pm0.56}$	$41.83_{\pm0.76}$	$58.99_{\pm0.34}$	$44.17_{\pm0.29}$
	NaiveRAG (sentence)	$44.39_{\pm0.54}$	$35.33_{\pm0.76}$	$57.48_{\pm0.81}$	$48.17_{\pm0.29}$	$61.36_{\pm0.71}$	$47.83_{\pm0.76}$
	ITER-RETGEN	$42.62_{\pm0.51}$	$32.17_{\pm0.76}$	$67.40_{\pm0.59}$	$52.33_{\pm0.76}$	$59.68_{\pm1.30}$	$48.33_{\pm1.26}$
	LongRAG	$47.02_{\pm1.49}$	$36.83_{\pm0.76}$	$64.86_{\pm1.68}$	$54.33_{\pm0.58}$	$\mathbf{66.59}_{\pm0.72}$	$\mathbf{52.17}_{\pm0.76}$
	HippoRAG w/ IRCoT	$44.29_{\pm0.47}$	$30.17_{\pm1.04}$	$63.07_{\pm0.99}$	$50.50_{\pm1.50}$	$59.05_{\pm0.30}$	$43.83_{\pm1.26}$
	ChainRAG (AnsInt)	$44.69_{\pm1.00}$	$33.67_{\pm1.04}$	$63.16_{\pm2.42}$	$53.67_{\pm2.02}$	$58.59_{\pm0.67}$	$45.67_{\pm1.04}$
	ChainRAG (CxtInt)	$51.17_{\pm1.20}$	$42.33_{\pm2.08}$	$62.15_{\pm2.13}$	$52.67_{\pm2.57}$	$62.84_{\pm0.57}$	$49.83_{\pm1.26}$
	DER-RAG (AnsInt)	$\underline{56.46}_{\pm0.18}$	$\underline{46.50}_{\pm0.50}$	$\mathbf{71.41}_{\pm0.22}$	$\mathbf{60.17}_{\pm0.29}$	$57.49_{\pm0.96}$	$44.17_{\pm0.76}$
	DER-RAG (CxtInt)	$\mathbf{57.13}_{\pm0.42}$	$\mathbf{48.83}_{\pm0.58}$	$\underline{70.67}_{\pm1.19}$	$\underline{59.33}_{\pm1.53}$	$\underline{63.36}_{\pm0.06}$	$\underline{50.67}_{\pm0.29}$
Qwen2.5-72B	Direct	$15.18_{\pm0.65}$	$7.67_{\pm0.29}$	$31.34_{\pm0.61}$	$26.17_{\pm0.58}$	$32.28_{\pm0.64}$	$24.50_{\pm0.50}$
	NaiveRAG (passage)	$23.31_{\pm0.33}$	$16.17_{\pm0.29}$	$48.91_{\pm0.43}$	$39.00_{\pm0.00}$	$37.99_{\pm0.49}$	$26.33_{\pm0.58}$
	NaiveRAG (sentence)	$41.64_{\pm0.22}$	$30.00_{\pm0.00}$	$63.08_{\pm0.60}$	$52.17_{\pm0.58}$	$56.98_{\pm0.10}$	$43.67_{\pm0.29}$
	ITER-RETGEN	$40.53_{\pm0.67}$	$30.83_{\pm0.76}$	$54.59_{\pm0.57}$	$45.17_{\pm0.76}$	$59.26_{\pm0.85}$	$46.33_{\pm0.58}$
	LongRAG	$40.69_{\pm0.40}$	$29.50_{\pm0.50}$	$63.74_{\pm1.17}$	$52.50_{\pm0.87}$	$60.10_{\pm0.25}$	$46.67_{\pm0.29}$
	HippoRAG w/ IRCoT	$44.27_{\pm0.29}$	$30.50_{\pm0.50}$	$64.46_{\pm0.53}$	$53.00_{\pm1.32}$	$55.42_{\pm0.77}$	$42.00_{\pm0.50}$
	ChainRAG (AnsInt)	$47.01_{\pm0.54}$	$36.67_{\pm0.58}$	$64.69_{\pm0.98}$	$54.50_{\pm1.32}$	$59.48_{\pm0.84}$	$45.50_{\pm0.87}$
	ChainRAG (CxtInt)	$48.38_{\pm0.55}$	$38.17_{\pm0.76}$	$64.62_{\pm0.99}$	$54.17_{\pm1.26}$	$\mathbf{63.22}_{\pm0.75}$	$\mathbf{50.67}_{\pm1.26}$
	DER-RAG (AnsInt)	$\mathbf{55.00}_{\pm0.54}$	$\mathbf{43.17}_{\pm0.58}$	$\underline{72.78}_{\pm0.35}$	$\mathbf{61.83}_{\pm0.29}$	$59.65_{\pm1.35}$	$46.00_{\pm1.80}$
	DER-RAG (CxtInt)	$\underline{52.75}_{\pm0.90}$	$\underline{40.33}_{\pm0.58}$	$\mathbf{73.15}_{\pm0.49}$	$\underline{60.67}_{\pm0.29}$	$\underline{62.78}_{\pm0.76}$	$\underline{48.50}_{\pm0.87}$

Table 1: Performance (%) on MuSiQue, 2Wiki, and HotpotQA (mean ± std over 3 runs). **Bold**: best per LLM block; underline: second best.

3.3 Final Answer Reasoning

After all sub-questions have been answered, DER-RAG produces the final answer using one of two integration strategies that follow Zhu et al. (2025), where ANSINT derives the final answer from the accumulated sub-questions and their answers and CXTINT derives it from the retrieved sentence context. Under both strategies the LLM reasons over multiple pieces of retrieved or accumulated evidence, so the same loss of entity grounding can recur at the final-answer stage. DER-RAG therefore applies subject entity prefixes at this stage as well, which keeps evidence grounded consistently from retrieval through to the final answer. The retrieval-controlled experiment in Section 4.3 covers both the sub-answer and the final-answer stage.

4 Experiments

4.1 Experimental Setup

We conduct experiments on MuSiQue (Trivedi et al., 2022), 2WikiMultiHopQA (Ho et al., 2020), and HotpotQA (Yang et al., 2018), following the 200-question data setting of Zhu et al. (2025), and we evaluate performance using F1 and exact match (EM). Dataset statistics are given in Appendix A.

For a fair comparison we fix the embedding model (text-embedding-3-small) and the cross-encoder reranker as BGE-Reranker-large (Chen et al., 2024a) across all systems. We compare DER-RAG against Direct (no retrieval), NaiveRAG (passage), NaiveRAG (sentence), ITER-RETGEN (Shao et al., 2023), LongRAG (Jiang et al., 2024), HippoRAG w/ IRCoT (Gutiérrez et al., 2024; Trivedi et al., 2023), and ChainRAG (Zhu et al., 2025). Main experiments use GPT-4o-mini (Achiam et al., 2023) and Qwen2.5-72B (Yang et al., 2024), and ablation studies use GPT-4o-mini unless otherwise noted. We report the two answer integration strategies of ChainRAG (Zhu et al., 2025), where ANSINT derives the final answer from the accumulated sub-question and its own answer, and CXTINT derives it from the retrieved sentence context.

4.2 Main Results

Table 1 reports results on all three benchmarks for both backbone LLMs. To ensure statistical reliability on the 200-question subsets, every experiment is repeated three times and reported as mean ± standard deviation. Averaged over the three benchmarks, DER-RAG matches or exceeds the strongest baselines under both backbone LLMs, and on every individual dataset the better of its two variants ranks first or second. The margins are clearest on MuSiQue and 2WikiMultiHopQA, while on

HotpotQA the methods fall within a narrow range, which is consistent with its predominantly two-hop questions, where the room for decomposition and entity recovery is smaller. Against the graph-based HippoRAG w/ IRCoT, DER-RAG performs better on all three datasets without any graph construction or offline preprocessing, which suggests that explicit query decomposition with entity-aware prefixing is a lightweight alternative to building offline structure. Among the baselines, sentence-level NaiveRAG improves over its passage-level counterpart on all datasets, which confirms the precision benefit of a finer retrieval unit, and LongRAG is strong on HotpotQA for GPT-4o-mini but does not carry this advantage to MuSiQue and 2Wiki.

Advanced RAG methods generally surpass NaiveRAG in mean performance but also show higher variance. ChainRAG in particular has large standard deviations, for example ± 2.42 F1 on 2Wiki for GPT-4o-mini, which we attribute to noise accumulating as the pipeline grows heavier, since more LLM calls and larger retrieved contexts across steps raise the chance of error propagation. DER-RAG instead reaches lower standard deviations than NaiveRAG in most settings, which indicates that grounding the reasoning in a compact retrieved context yields more consistent outputs, and this stability is clearest on MuSiQue and 2Wiki where the accuracy gains are also largest.

We report DER-RAG under both answer integration strategies, ANSINT and CXTINT, because entity loss can arise both when a sub-question is answered and when the final answer is generated. DER-RAG matches or exceeds strong baselines under both strategies on the large majority of datasets and LLMs, which indicates that subject entity prefixing addresses the failure mode across both points of answer synthesis.

4.3 Lost-in-Generation under Controlled Retrieval

This experiment tests the central claim that subject entity prefixing improves generation independently of retrieval quality. To remove retrieval as a confound, we bypass the retriever and build the context of each question from all passages annotated for it, where the gold passages contain the answer and the remaining passages serve as distractors. The correct evidence is therefore guaranteed to be in the context in every condition. On this fixed context we compare three ways of presenting the evidence to the LLM, the full passage, the passage split into sentences, and the split sentences with a subject entity prefix. The split sentences are reordered by the same reranker, applied identically to all three evidence formats. Since the underlying passages are identical in all three conditions, any difference in answer quality reflects how the evidence is presented rather than what was retrieved. The fixed context contains 8.6 to 11.1 passages per question on average, with details given in Appendix A.

Evidence Form	MuSiQue		2Wiki		HotpotQA	
	F1	EM	F1	EM	F1	EM
Passage	**49.18**	**39.5**	**65.12**	**53.5**	64.70	**51.5**
Sentence	46.24	36.0	62.88	52.0	62.02	48.5
Prefixed sentence	47.85	35.5	64.55	**53.5**	**65.24**	50.5

Table 2: Lost-in-generation under controlled retrieval. The context is fixed to the gold plus distractor passages, with no retriever run over the corpus.

QD Strategy	MuSiQue		2Wiki		HotpotQA	
	F1	EM	F1	EM	F1	EM
One-step QD	39.41	31.50	59.19	49.00	**65.65**	**52.50**
Post-hoc	45.43	35.50	60.00	49.50	65.28	51.00
Two-way QD	**49.87**	**41.00**	**64.43**	**53.00**	63.63	51.00

Table 3: Comparison of query decomposition strategies on a NaiveRAG base. Post-hoc follows ChainRAG (Zhu et al., 2025), and Two-way QD is our method.

Table 2 reports the result. Splitting a passage into sentences lowers F1 on all three datasets, by 2.2 to 2.9 points. Because the retriever is bypassed and the underlying passages are unchanged, this drop cannot come from retrieval, which makes it a direct measurement of lost-in-generation, in that splitting alone, by detaching a sentence from the context that resolves its pronouns, degrades generation. Prefixing the subject entity recovers most of this loss, improving over the unmodified sentence by 1.6 to 3.2 points and bringing it back to roughly the full-passage level.

This also resolves an apparent tension with the ablation in Section 4.4, where the sentence outperforms the passage under ordinary retrieval. The two orderings are consistent once retrieval and generation are separated, since a finer unit helps retrieval (Chen et al., 2024b) but hurts generation, and the plain sentence carries both effects, so its retrieval gain is partly cancelled by lost-in-generation. The prefixed sentence keeps the retrieval advantage of a fine unit, since the index is unchanged, while repairing the generation side.

Retrieval Unit	MuSiQue		2Wiki		HotpotQA	
	F1	EM	F1	EM	F1	EM
Passage	34.89	28.0	51.51	41.0	58.74	44.0
Sentence	44.68	**36.0**	56.97	48.0	60.54	47.0
Prefixed sentence	**47.72**	35.5	**66.47**	**55.5**	**64.51**	**52.5**

Table 4: Retrieval unit ablation on a NaiveRAG base. Prefixed sentence is the unit used by DER-RAG.

4.4 Ablation Study

DER-RAG modifies two parts of the standard pipeline, the query decomposition strategy and the retrieval unit, so we ablate each on a common NaiveRAG base. All ablation runs use the same embedding model, reranker, and evaluation setup as the main results, so the numbers are directly comparable with Table 1.

We first ablate the query decomposition strategy. Table 3 compares three strategies on the same NaiveRAG base, where one-step QD generates all sub-questions in advance without access to intermediate answers, ChainRAG (Zhu et al., 2025) adds a post-hoc rewriting step that recovers omitted entities after generation, and two-way QD supplies the intermediate entity while each sub-question is generated. On MuSiQue and 2Wiki, two-way QD improves F1 by 4.4 points over post-hoc rewriting and by 10.5 and 5.2 over one-step QD, which shows that supplying the entity during generation beats recovering it afterward. On HotpotQA the three strategies perform within a narrow range, with two-way QD slightly behind, consistent with its predominantly two-hop questions where decomposition strategy matters little. The advantage of two-way QD is therefore clearest on datasets that require deeper or less uniform reasoning chains.

We next ablate the retrieval unit. Table 4 compares the full passage, the unmodified sentence, and the prefixed sentence used by DER-RAG on the same NaiveRAG base. The unmodified sentence improves over the passage, confirming the precision benefit of a finer unit, and prefixing then adds a further gain over the unmodified sentence, by 3.0, 9.5, and 4.0 F1 points on MuSiQue, 2Wiki, and HotpotQA. Since the prefixed and unmodified sentences are retrieved identically and differ only by the prefix, this gain isolates entity grounding at generation time rather than any retrieval effect, and the comparison against the passage condition shows it comes from restoring the missing entity, not from supplying more context.

	MuSiQue	2Wiki	HotpotQA
ChainRAG	8.98	7.52	8.00
DER-RAG	9.63	8.00	9.11

Table 5: Comparison of LLM calls (GPT-4o-mini).

	MuSiQue	2Wiki	HotpotQA
ChainRAG	103.84	67.24	75.36
DER-RAG	33.10	24.94	30.34

Table 6: Comparison of retrieved sentences N.

4.5 Online Efficiency

We compare DER-RAG and ChainRAG on online inference cost along two dimensions, the number of LLM calls (Table 5) and the number of retrieved context sentences (Table 6). DER-RAG issues a comparable number of LLM calls to ChainRAG, slightly more on each dataset, since two-way query decomposition follows a short chain of sub-questions. It operates, however, on far fewer retrieved sentences, around 25 to 33 per question against 67 to 104 for ChainRAG, because subject entity prefixing restores entity grounding directly rather than retrieving additional sentences to supply the missing context. DER-RAG therefore reaches its accuracy with a substantially smaller generation context at a similar number of LLM calls.

5 Conclusion

We revisited entity loss in multi-hop RAG and showed that it breaks the pipeline at two stages. Lost-in-retrieval, identified in prior work, occurs when a sub-question drops the entity resolved at the previous hop. Lost-in-generation, which we isolate as a distinct failure mode in the generation step, occurs when sentence splitting detaches a sentence from the context that grounds its pronouns, and a retrieval-controlled experiment shows that it degrades answers even when the gold evidence is fixed. We proposed DER-RAG, which keeps the grounding entity explicit across both stages with two lightweight components and requires no graph construction, no corpus modification, and no fine-tuning. Across three multi-hop QA benchmarks, DER-RAG matches or exceeds strong baselines, including graph-based methods, which suggests that addressing entity loss directly is a worthwhile alternative to building heavier retrieval structures.

6 Limitations

Two limitations of DER-RAG follow from its core design. First, DER-RAG assigns a single subject entity to each passage and reuses it for every sentence and every query. Our analysis in Appendix F shows that this is often too rigid, since the most useful subject entity is not a fixed expression but varies in granularity, and which form helps most can depend on the query being answered. A passage-level entity also cannot resolve every coreference chain within a passage, because some sentences refer to entities other than the main subject. Extracting finer, query-aware entities is a natural next step. Second, DER-RAG decides the decomposition depth once, right after the question type classification. This keeps the method train-free and is enough to support accurate reasoning on the benchmarks we study, but a policy that revisits the decision at every step could adapt more finely to questions whose reasoning depth is hard to predict in advance. Making such a per-step policy train-free, rather than relying on fine-tuning as current approaches do, is a natural direction for future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024a. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 4(5).

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024b. Dense x retrieval: What retrieval granularity should we use? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15159–15177.

Eunsol Choi, Jennimaria Palomaki, Matthew Lamm, Tom Kwiatkowski, Dipanjan Das, and Michael Collins. 2021. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics*, 9:447–461.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieve step by step for large language models. *arXiv preprint arXiv:2502.01142*.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Ziyan Jiang, Xueguang Ma, and Wenhu Chen. 2024. Longrag: Enhancing retrieval-augmented generation with long-context llms. *arXiv preprint arXiv:2406.15319*.

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

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, and 1 others. 2024. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12758–12786.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. 2020. Unsupervised question decomposition for question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8864–8880.

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

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. 2024. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pages 32628–32649.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 10014–10037.

Chenyuan Wu, Ninglu Shao, Zheng Liu, Shitao Xiao, Chaozhuo Li, Chen Zhang, Senzhang Wang, and Defu Lian. 2025. Lighter and better: Towards flexible context adaptation for retrieval augmented generation. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 271–280.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, and 25 others. 2024. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Rongzhi Zhu, Xiangyu Liu, Zequn Sun, Yiwei Wang, and Wei Hu. 2025. Mitigating lost-in-retrieval problems in retrieval augmented multi-hop question answering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22362–22375.

A Dataset Statistics and Implementation Details

Table 7 presents statistics for the three benchmarks used in our evaluation. Each dataset comprises 200 questions, and the corpus associated with each question consists of multiple passages of varying length. MuSiQue and HotpotQA tend to have longer passages, with 43.69 and 47.93 sentences per passage on average, while 2WikiMultiHopQA passages are shorter at 22.75 sentences on average. This difference in passage density is relevant to lost-in-generation, since denser passages produce more pronoun-bearing sentences once they are split, which increases the need for subject entity prefixing. For a fair comparison across all methods we fix the embedding model (text-embedding-3-small) and the cross-encoder reranker (BGE-Reranker-large) throughout. Sentence splitting uses the "en_core_web_sm" model from spaCy library, and all methods operate on the same preprocessed corpus. For ITER-RETGEN we follow the original settings and report results at the third iteration, where performance plateaus. LongRAG and HippoRAG are evaluated with their publicly released code under default configurations, with only the backbone LLM and embedding model replaced for a unified comparison, where LongRAG applies word-count-based chunking with a chunk size of 200 and HippoRAG operates on unsegmented passages. For DER-RAG, subject entities are extracted offline in a single preprocessing pass and cached, which adds no latency to online inference. All ablation experiments use GPT-4o-mini unless otherwise noted.

	MuSiQue	2Wiki	HotpotQA
No. of Samples	200	200	200
No. of Passages	2,219	1,986	1,722
Avg. Sents	43.69	22.75	47.93

Table 7: Dataset statistics. Avg. Sents: average number of sentences per passage.

For the controlled experiment in Section 4.3, the context of each question is fixed to its gold passages together with distractor passages drawn from the same pool, so that the correct evidence is always present. The combined passages are then split into sentences and ordered by the same reranker, and the prefixed-sentence condition additionally prepends the subject entity of each source passage. Table 8 reports the resulting number of passages per question, which is used in Section 4.3. It averages 8.6 to 11.1 across the three datasets and varies from as few as 2 to as many as 20, so the fixed context still contains a substantial amount of non-supporting material.

All datasets (MuSiQue, 2WikiMultiHopQA, HotpotQA) and models used in this work are publicly available and were used in accordance with their respective licenses and intended research use.

	MuSiQue	2Wiki	HotpotQA
Mean	11.09	9.93	8.61
Min	3	6	2
Max	20	10	10

Table 8: Number of passages (gold plus distractor) in the fixed context used for the controlled experiment in Section 4.3.

B Question Type Classification Details

Table 9 reports the accuracy of the LLM-based binary classifier that routes each question to its decomposition path. The horizontal A/B labels denote the ground-truth question types (actual labels), while the vertical A/B labels denote the classifier's predicted question types (predicted labels). The classifier reaches over 99% accuracy on all three datasets, with 100% on MuSiQue, 99.5% on 2WikiMultiHopQA, and 99% on HotpotQA. The few misclassified cases are predominantly bridge questions that name two entities in their surface form, which leads the classifier to label them as comparison questions. Given this near-perfect accuracy, classification error contributes negligibly to overall pipeline performance. The classification prompt is shown in Figure 4.

	MuSiQue		2Wiki		HotpotQA	
	A	B	A	B	A	B
A	0	0	82	1	42	2
B	0	200	0	117	0	156
EM	100		99.5		99	

Table 9: Question type classification results using GPT-4o-mini. A: comparison type, B: non-comparison type. Diagonal cells indicate correct classifications.

Classify Comparative Question

You are a classifier that determines whether a question is Comparative or Not.

A question is Comparative if and only if:

1. It involves TWO or more explicitly named entities (not derived via reasoning).
2. The final answer requires a direct comparison between those entities.
3. The answer is one of: yes/no, one of the named entities, or a shared/differing attribute.

A question is Not Comparative if:

1. One entity must be derived first before the comparison can be made. (bridge)
2. Multiple entities/items appear in the question but serve as clues, not comparison targets.

Example: {few-shots}

Figure 4: Prompt for classifying whether a question is comparative or not.

C Portability of Two-Way Query Decomposition

The query decomposition ablation in Section 4.4 measures two-way QD on a NaiveRAG base. Here we additionally test whether the same decomposition transfers to a different pipeline, by replacing the post-hoc rewriting step of ChainRAG (Zhu et al., 2025) with two-way QD while leaving the rest of ChainRAG unchanged. Table 10 reports the result.

As shown in Table 10, replacing post-hoc rewriting with two-way QD consistently improves or matches performance across all three datasets, demonstrating that the proposed decomposition strategy is not tied to a specific pipeline and can serve as a portable drop-in replacement within existing RAG frameworks.

D Effect of Two-Way Routing

DER-RAG routes comparison questions to one-step decomposition and all other questions to multi-step decomposition. This experiment isolates the contribution of that routing by comparing the full two-way design against a variant that removes the routing and applies multi-step decomposition to every question, including comparison questions. All other components, the retriever, the prefixing, and the generation setup, are held fixed, so any difference reflects only the routing.

Table 11 compares the full two-way routing design against a variant that applies multi-step decomposition to every question without routing. The results vary across datasets depending on the proportion of comparison-type questions (see Table 9 for the distribution of question types per dataset).

On Musique, which contains no comparison questions, removing the routing yields virtually no change in performance, confirming that the routing mechanism has no adverse effect when comparison questions are absent. On 2Wiki, which has the highest proportion of comparison questions (82 out of 200), the no-routing variant shows a notable performance drop, demonstrating the effectiveness of the two-way routing in redirecting comparison questions to the more appropriate one-step decomposition path. On HotpotQA, the performance gap between the two variants is relatively small. We attribute this to the comparatively lower accuracy on comparison-type questions in HotpotQA, which diminishes the marginal benefit of routing. Taken together, these findings validate the utility of the two-way routing strategy: it provides meaningful gains precisely in settings where comparison questions are frequent and answerable.

E Decomposition Depth

This section reports analyses of decomposition depth on the multi-step decomposition. DER-RAG predicts the number of sub-questions after question type classification, rather than fixing it in advance (Section 3.1). We record the predicted count for every question and report its distribution per dataset, which shows that, although the procedure is not restricted to a fixed depth, it terminates within a small number of steps for the large majority of questions on these benchmarks. Table 12 shows the distribution of predicted sub-question counts across the three datasets. Questions assigned zero sub-questions are comparison-type questions that the two-way routing directs to one-step decomposition; they are excluded from multi-step QD entirely. Among the remaining questions, the vast majority are assigned exactly two sub-questions across all datasets, with only a small fraction receiving one, confirming that the benchmarks are dominated by two-hop reasoning chains. No question is assigned three or more sub-questions, indicating that the predicted depth aligns well with the actual reasoning requirements of these benchmarks and that

Method	QD		MuSiQue		2Wiki		HotpotQA	
			F1	EM	F1	EM	F1	EM
ChainRAG	Post-hoc rewriting	AnsInt	50.54	37.0	62.55	52.0	60.73	46.0
		CxtInt	47.87	38.5	56.54	50.5	64.59	50.0
	Two-way	AnsInt	54.00	43.5	**68.57**	**58.0**	61.20	46.5
		CxtInt	**54.05**	**45.5**	63.41	53.5	**66.14**	**52.0**

Table 10: Performance comparison of replacing ChainRAG's post-hoc rewriting with our two-way QD.

Methods		MuSiQue		2Wiki		HotpotQA	
		F1	EM	F1	EM	F1	EM
Ours	AnsInt	56.25	46	**71.66**	**60**	58.02	45
	CxtInt	**56.76**	**48.5**	72.04	61	63.41	51
Ours (w/o routing)	AnsInt	56.14	46.5	67.01	55.5	59.47	46
	CxtInt	57.41	**48.5**	68.06	56.5	**65.86**	**52**

Table 11: Effect of Two-Way Routing (k=10)

	MuSiQue	2Wiki	HotpotQA
0	0	82	42
1	13	1	12
2	187	117	146
Total	200	200	200

Table 12: Distribution of predicted sub-question counts per dataset. Depth 0 indicates comparison-type questions routed to one-step decomposition, which are excluded from multi-step QD.

the procedure terminates compactly without over-decomposing.

F Title vs. Subject Entity

Table 13 reports the similarity between LLM-extracted subject entities and original passage titles, measured by token-level F1 and exact match. F1 scores of 82 to 86 across all three datasets indicate that the extracted entities closely match the passage titles in most cases, and the gap between F1 and EM reflects partial matches where the LLM produces a slightly different surface form of the correct entity, such as "Daniel Day-Lewis" against "Daniel Day Lewis".

	MuSiQue	2Wiki	HotpotQA
F1	86.06	82.31	82.76
EM	66.97	57.30	57.38

Table 13: Similarity of subject entities extracted by GPT-4o-mini to the original passage titles.

Table 14 directly compares the two prefix strategies in the full DER-RAG pipeline at k=10. Subject entity prefixing is comparable to or better than title prefixing across most settings, and the gain is largest on 2WikiMultiHopQA under CxtInt, where subject entity prefixing improves F1 from 68.84 to 72.04. This is consistent with our case study, in which LLM-extracted entities more precisely identify the central topic of a passage, particularly when the title refers to a sub-entity rather than the primary subject. Title prefixing is marginally higher on MuSiQue under AnsInt, which suggests that the effect of prefix precision varies with both the dataset and the answer integration strategy. We observe the same pattern on a NaiveRAG base, where title and subject entity prefixes yield comparable performance across the three datasets, which indicates that the choice between the two prefixes has a limited effect on accuracy and that extracted subject entities can be used as the general mechanism without a performance cost.

Prefix		MuSiQue		2Wiki		HotpotQA	
		F1	EM	F1	EM	F1	EM
Title	AnsInt	**56.82**	45.5	71.56	**61.0**	56.98	43.5
	CxtInt	55.48	46.0	68.84	57.5	62.98	49.0
Subject	AnsInt	56.25	46.0	71.66	60.0	58.02	45.0
	CxtInt	56.76	**48.5**	**72.04**	**61.0**	**63.41**	**51.0**

Table 14: Comparison of prefix strategies (title vs. subject entity) in DER-RAG (k=10).

To understand these results we conducted a hu-

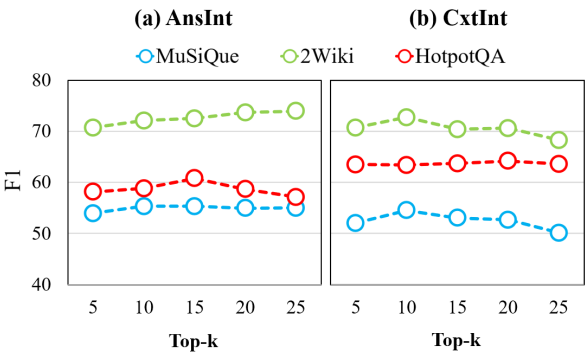

Figure 5: F1 as a function of Top-k, for AnsInt (left) and CxtInt (right) across all three datasets.

man evaluation that categorises each passage into three cases, defined in Table 15, where each case was identified by human annotators. Table 16 reports the resulting distribution. Case 1, in which only the title is correct, accounts for fewer than 3% of passages on every dataset, which confirms that this failure mode is relatively rare.

Two further patterns emerged from the evaluation. First, as Case 3 shows, the extracted entity sometimes corrects an inaccurate title, since when a title refers to a sub-entity such as a single album in a discography passage, the LLM identifies the artist as the primary subject and yields a more informative prefix than the title. Second, even in Case 2, where both the title and the subject entity are judged correct, extraction often provides a complementary advantage, because many titles are abbreviated forms such as a shortened place name or a partial person name, whereas the LLM tends to produce a fuller surface form that is better suited as a prefix. This suggests that extracted entities can be a better prefix even when the title is technically accurate.

These observations also point to a deeper question, namely that the most appropriate subject entity for a passage is not a single fixed expression but can take several valid forms that differ in specificity, and which form is most effective as a prefix may further depend on the query being answered. One direction for future work is therefore to identify a small set of candidate entities at varying granularity per passage and select the most query-relevant one at retrieval time.

G Top-k Study

Figure 5 plots F1 as a function of the number of retrieved sentences k, swept from 5 to 25. Across both integration types and all three datasets, performance remains relatively stable throughout the range. A notable exception is 2Wiki under AnsInt, where F1 shows a gradual increase as k grows, suggesting that broader retrieval coverage benefits this dataset's multi-hop structure. Overall, however, beyond k=10, larger values tend to yield diminishing returns or a slight decline, an effect that is more pronounced under CxtInt. We attribute this to the fact that CxtInt concatenates all retrieved contexts from each sub-question and passes them jointly to the generation stage, causing the input to grow substantially with k and introducing greater context noise. We therefore adopt k=10 as the default in all experiments.

H Case Study

Table 17 traces a multi-hop QA example through three RAG methods to show how the two entity-loss problems arise and how DER-RAG resolves them.

For NaiveRAG w/ QD, the first sub-question correctly identifies the bridging nursery rhyme, but because one-step QD generates all sub-questions at once without access to intermediate answers, the second sub-question is formed without substituting the resolved entity, which leaves the retriever unable to identify the correct context and produces an incorrect answer. For ChainRAG, post-hoc rewriting recovers the missing entity and issues a well-formed second sub-question, which fixes retrieval, but the retrieved sentences all read "It has a Roud Folk Song Index number of . . . " with no indication of which nursery rhyme each sentence describes, so the LLM cannot resolve the implicit subject and again answers incorrectly. For DER-RAG, the second sub-question is the same well-formed one as in ChainRAG, and each retrieved sentence is then prefixed with the subject entity of its passage, so the LLM identifies the sentence belonging to the target nursery rhyme and produces the correct answer. This example shows that lost-in-retrieval and lost-in-generation are independent failure modes, since even when retrieval succeeds the absence of entity grounding in the generated context can still lead to a wrong answer, and fixing only one problem is insufficient while the other remains.

I Prompts

DER-RAG uses an LLM with task-specific prompts at three points: subject entity extraction, question type classification, and sub-question generation.

Passage	Check	Analysis
Case 1		
Countess Albina du Boisrouvray is a former journalist and film producer who has become a global philanthropist. . . She is the founder of FXB International, a non-governmental organization established in memory of her son.	✓ title × subject	Title correctly points to the person the passage is about, but subject captures the organization she founded — a related but distinct entity.
Case 2		
Thordis Markusdottir, known as Stokkseyrar-Disa (1668–1728), was an Icelandic magician (Galdrmaster). She is known for her alleged magical powers and is the subject of at least ten different folk sagas.	✓ title ✓ subject	The passage opens by giving her real name and immediately states she is "known as" the title — both names refer to the same person.
Case 3		
Aziza Mustafa Zadeh (born 1969) is an Azerbaijani singer, pianist, and composer who plays a fusion of jazz and mugham. . . Discography: Aziza Mustafa Zadeh (1991), Always (1993), Dance of Fire (1995), Jazziza (1997). . .	× title ✓ subject	The title is just one album listed in the discography section, subject correctly identifies the artist the passage describes.

Table 15: Case study of subject entity extraction (EM < 1).

	MuSiQue	2Wiki	HotpotQA
Case 1	53 (2.4%)	46 (2.3%)	43 (2.5%)
Case 2	2,155 (97.1%)	1,915 (96.4%)	1,665 (96.7%)
Case 3	11 (0.5%)	25 (1.3%)	14 (0.8%)
Total	2,219	1,986	1,722

Table 16: Distribution of subject entity extraction cases per dataset. Case 1: title correct, subject incorrect. Case 2: both title and subject correct. Case 3: title incorrect, subject correct (subject outperforms title).

All prompts include few-shot demonstrations, omitted here for brevity, and are applied without any task-specific fine-tuning. The subject entity extraction prompt is shown in Figure 6 and the classification prompt in Figure 4.

Figures 7 to 9 present the sub-question generation prompts. Figure 7 is the one-step decomposition prompt for comparison questions, which splits the question into two parallel sub-questions, one per named entity. Figure 8 is the prompt for the first sub-question in the multi-step chain for non-comparison questions. Figure 9 is the prompt for the final sub-question, which substitutes the entity resolved at the preceding step directly into the sub-question wording.

J Use of AI Assistants

AI assistants were used for writing assistance, including grammar checking, sentence-level editing, and improving the clarity and conciseness of the text. All technical content, experimental results, and claims in this paper are the authors' own.

Extract subject-entity

You are an information extraction assistant.

Your task is to identify the single primary subject entity of a given paragraph. The subject entity is the main object, person, place, concept, or event that the paragraph is primarily describing.

Rules:
- Output only the entity name, nothing else.
- Use a concise noun phrase (e.g., *examples*).
- Do not output sentences, explanations, or punctuation.
- If multiple entities are present, choose the most central one.

Figure 6: Prompt for extracting subject entity of a paragraph.

Question: What was the Roud Folk Song Index of the nursery rhyme inspiring *What Are Little Girls Made Of?*			
Method	**Sub-question 1**	**Sub-question 2**	**Answer**
NaiveRAG w/ QD	**Q:** What is the nursery rhyme that inspired *'What Are Little Girls Made Of'?* **Context:** ...the title is taken from the nursery rhyme, "What Are Little Boys Made Of?" **Answer:** "What Are Little Boys Made Of?"	**Q:** What is the Roud Folk Song Index of this nursery rhyme? **Context:** • The nursery rhyme has a Roud Folk Song Index number of 19800. • It has a Roud Folk Song Index number of 7734. • It has a Roud Folk Song Index number of 12986. • It has a Roud Folk Song Index number of 821. **Answer:** 19800	19800
ChainRAG	**Q:** What is the nursery rhyme inspiring *What Are Little Girls Made Of?* **Context:** ...the title is taken from the nursery rhyme, "What Are Little Boys Made Of?" **Answer:** What Are Little Boys Made Of?	**Q:** What was the Roud Folk Song Index of What Are Little Boys Made Of? **Context:** • It has a Roud Folk Song Index number of 19800. • It has a Roud Folk Song Index number of 7734. • It has a Roud Folk Song Index number of 12986. • It has a Roud Folk Song Index number of 821. **Answer:** 12986	12986
DER-RAG (ours)	**Q:** What is the nursery rhyme inspiring *What Are Little Girls Made Of?* **Context:** ...the title is taken from the nursery rhyme, "What Are Little Boys Made Of?" **Answer:** What Are Little Boys Made Of?	**Q:** What was the Roud Folk Song Index of What Are Little Boys Made Of? **Context:** • Tweedledum and Tweedledee: ... Song Index number of 19800. • Three Little Kittens: ... Song Index number of 7734. • A Wise Old Owl: ... Song Index number of 12986. • What Are Little Boys Made Of?: ... Song Index number of 821. **Answer:** 821	821

Table 17: RAG process comparison on a HotpotQA dataset. Blue = correct/relevant; red = incorrect; green = subject entities prefixed to retrieved sentences.

Single-step QD

You are a helpful AI assistant that decomposes a comparative question.

Guidelines:

1. Output exactly 2 sub-questions as a JSON array (and nothing else).
2. Each sub-question should extract the relevant attribute or fact for one of the two compared entities.
3. Stay close to the original question wording by reusing it.
4. If the original question specifies what type of entity is being compared (e.g., film, book, person, song), always include that type word in each sub-question.

Example: {few-shots}

Figure 7: Prompt for decomposing a comparative question into two sub-questions.

Multi-step QD (First Sub-question)

You are a helpful AI assistant that generates only one sub-question for multi-hop reasoning.

Goal: Produce the single best first sub-question.

What is a sub-question:

- Ask for the missing link (an intermediate entity/value).
- Do not ask for the final answer to the original question.
- Stay close to the original question wording by reusing key phrases/entities.

Hard rules:

1. Output exactly one sub-question.
2. Avoid vague or generic prompts (e.g., "Who is A?") unless the original question truly requires identifying A.
3. Do not use pronouns or demonstratives that rely on previous answers (forbidden: it, this, that, they, these, those, he, she, his, her, its, their).
4. Make the subject explicit using entities already in the original question.

Example: {few-shots}

Figure 8: Prompt for generating the first sub-question for non-comparative question.

Multi-step QD (Last Sub-question)

You are a helpful AI assistant that generates the last sub-question for the original question.

Input:

- Original question: user's final goal
- Current sub-question: question just answered
- Current answer: answer to the current sub-question

Task:

- Using the current answer as a known fact, generate exactly one final sub-question that directly leads to the answer of the original question.
- This sub-question must be the last step; its answer should directly resolve the original question.

Hard rules:

1. Output only one question in JSON (no lists, no explanations).
2. Do not use pronouns or demonstratives that require context (forbidden: it, this, that, they, these, those, he, she, his, her, its, their).
3. Substitute the relevant entity from the current answer directly into the sub-question.
4. Stay close to the original question wording by reusing it.
5. If the original question contains family relations (e.g., "maternal grandfather", "father-in-law"), determine which relation step has been resolved by the current sub-question, then ask only the remaining next step.

Example: {few-shots}

Figure 9: Prompt for generating the last sub-question for non-comparative question.