IndexRAG: Index-Time Reasoning for Multi-Hop Retrieval-Augmented Generation

Zhenghua Bao, Yi Shi
Continuum AI
research@orcarouter.ai

Abstract

Multi-hop question answering (QA) requires reasoning across multiple documents, yet existing retrieval-augmented generation (RAG) approaches address this either through graph-based methods requiring additional online processing or iterative multi-step reasoning. We present IndexRAG, a novel approach that shifts cross-document reasoning from online inference to offline indexing. IndexRAG identifies bridge entities shared across documents and generates bridging facts as independently retrievable units, requiring no additional training or fine-tuning. Experiments on three widely-used multi-hop QA benchmarks (HotpotQA, 2WikiMultiHopQA, MuSiQue) show that IndexRAG improves F1 over Naive RAG by 4.6 points on average, while requiring only single-pass retrieval and a single LLM call at inference time. When combined with IRCoT, IndexRAG achieves the best average performance among all evaluated methods, including graph-based baselines such as HippoRAG2 and FastGraphRAG, while relying on a flat vector index. Our code is available at https://github.com/Continuum-AI-Corp/IndexRAG.

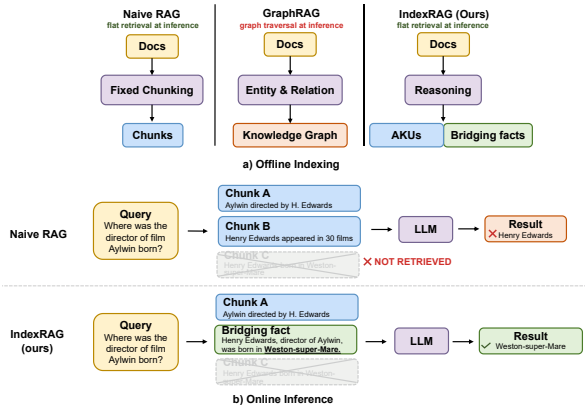


Figure 1: (a) Comparison of different RAG approaches during offline indexing. IndexRAG shifts reasoning to index time, generating bridging facts stored alongside AKUs in a unified vector store. (b) Example from 2WikiMultiHopQA (Ho et al., 2020). The query **"Where was the director of the film Aylwin born?"** requires two-hop reasoning. With naive RAG, Chunk C containing the director's birthplace is not retrieved, causing the LLM to output the director's name instead. With IndexRAG, a bridging fact supplying the director's birthplace is retrieved directly, completing the reasoning chain. The panel is illustrative, with the retrieved units reported in Table 6 and analyzed in Section 5.2.1.

1 Introduction

Large language models (LLMs), built on the Transformer architecture (Vaswani et al., 2017), have achieved remarkable performance across a wide range of tasks, spanning language (Raffel et al., 2020; Brown et al., 2020), vision (Dosovitskiy et al., 2021; Radford et al., 2021), and speech (Zhang et al., 2023; Défossez et al., 2024). However, their reliance on static parametric knowledge limits their ability to incorporate domain-specific or up-to-date information, often leading to hallucinations (Huang et al., 2025). Retrieval-Augmented Generation (RAG) mitigates this by grounding generation in an external knowledge base, allowing the model to retrieve relevant context before responding (Lewis et al., 2020).

While effective for single-hop questions, conventional RAG pipelines retrieve passages independently and struggle when answering a question requires synthesizing information across multiple documents. Multi-hop QA requires reasoning over multiple pieces of evidence spread across different documents to reach the correct answer (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022). Graph-based RAG systems (Edge et al., 2025; Gutiérrez et al., 2024; Gutiérrez et al., 2025; You et al., 2025) address this by constructing knowledge graphs that explicitly represent inter-document relationships. However, these methods often require multi-step online processing, such as query-time entity extraction, graph traversal, and multiple LLM calls,

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 3491–3506
November 6-10, 2026 ©2026 Association for Computational Linguistics

	Single-pass retrieval	Cross-doc reasoning	Single LLM call	Training-free	Index-time reasoning
Naive RAG	✓	×	✓	✓	×
HippoRAG2	×	✓	×	✓	×
IRCoT	×	✓	×	✓	×
IndexRAG (Ours)	✓	✓	✓	✓	✓

Table 1: Qualitative comparison of different RAG approaches. IndexRAG is the only method that achieves cross-document reasoning with single-pass retrieval and a single LLM call at inference time.

which increases inference cost and latency. Iterative approaches such as IRCoT (Trivedi et al., 2023) decompose complex queries through multiple rounds of retrieval and generation, resulting in higher inference cost and slower response times.

To address these limitations, we propose IndexRAG, a novel approach that shifts cross-document reasoning from online inference to offline indexing. We observe that these connections do not depend on the query. When two documents mention the same entity, the link between them is already fixed by what the documents say. This makes it possible to precompute these reasoning connections at indexing time, transforming per-query reasoning into an offline step. We refer to this paradigm as *index-time reasoning*. During indexing, we extract atomic knowledge units (AKUs) and entities from each document, and identify entities that appear across multiple documents. We then prompt an LLM to generate *bridging facts* that capture cross-document reasoning by linking related evidence from different sources. Both AKUs and bridging facts are stored in a unified vector store. Figure 1 sets this against existing pipelines and traces one query through both. At inference time, a single retrieval pass and a single LLM call suffice, with no graph traversal or repeated retrieval rounds. Table 1 summarizes how IndexRAG differs from existing approaches.

Our main contributions are as follows:

- We propose **IndexRAG**, a novel approach that shifts cross-document reasoning from online inference to offline indexing.
- We introduce *bridging facts*, a new type of retrieval unit that encodes cross-document reasoning as independently retrievable entries in a flat vector store, along with a balanced context selection mechanism to control their proportion at retrieval time.
- We propose a training-free framework that is agnostic to the underlying retrieval strategy and compatible with iterative methods such as IRCoT, requiring no fine-tuning of the embedding model or LLM.
- Experiments on three multi-hop QA benchmarks show IndexRAG outperforms all single-call baselines on average and, when combined with IRCoT, achieves the best average performance among all evaluated methods.

2 Related Work

2.1 Retrieval-Augmented Generation

To mitigate hallucinations arising from the static parametric knowledge of LLMs, RAG augments generation with external retrieval (Lewis et al., 2020). In a typical RAG pipeline, documents are encoded into a vector store, and relevant passages are retrieved based on embedding similarity at inference time. Recent research has focused on improving retrieval representations. Karpukhin et al. (2020) replace sparse retrieval with learned dense representations, while Gao et al. (2023) prompt the LLM to generate a hypothetical answer and use it as the retrieval query to better align with target passages. Other work investigates how documents should be segmented for indexing. Chen et al. (2024) show that finer-grained units such as propositions yield stronger retrieval than fixed-size chunks. While propositions decompose individual passages into finer units, they do not capture reasoning across documents. RAPTOR (Sarthi et al., 2024) recursively summarizes passages into a hierarchical tree structure for retrieval.

These methods primarily focus on improving how individual passages are represented or organized. In contrast, our work focuses on the implicit reasoning connections across documents, generating new retrieval units that make cross-document

inferences explicitly retrievable.

2.2 Multi-hop Reasoning in RAG

Standard RAG retrieves passages independently, struggling with questions that require reasoning across multiple documents (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022). Two main directions have emerged to address this limitation.

The first incorporates graph structures into RAG. GraphRAG (Edge et al., 2025) constructs an entity graph from the corpus, applies community detection, and generates summaries for each community to support global reasoning. FastGraphRAG[1] is another graph-based approach that prioritizes inference efficiency. HippoRAG (Gutiérrez et al., 2024) draws inspiration from the human hippocampal memory system, using a knowledge graph as a long-term memory index to connect related passages, with Personalized PageRank for graph-based retrieval. HippoRAG2 (Gutiérrez et al., 2025) extends this line with additional fact-level embeddings while retaining Personalized PageRank traversal at query time. MS-RAG (You et al., 2025) maintains separate vector stores for entities, relations, and chunks, using vector search for entity extraction, but still requires graph traversal at inference time. Although these methods improve multi-hop performance, they require additional inference-time processing, such as entity extraction, graph traversal, or reranking, beyond standard vector search. Recent agentic memory systems (Chhikara et al., 2025; Kang et al., 2025; Xu et al., 2025; Hu et al., 2026) focus on memory construction for agentic or dialogue settings, while our work studies index-time reasoning for fixed-corpus multi-hop QA.

The second direction uses iterative retrieval. IRCoT (Trivedi et al., 2023) interleaves chain-of-thought reasoning with retrieval, using intermediate reasoning steps to formulate new queries across multiple rounds. This improves recall of relevant evidence but introduces repeated retrieval and generation cycles that increase inference latency. Self-Correcting RAG (Xu et al., 2026) pairs a knapsack formulation of context selection with an NLI-guided tree search that can issue further retrieval calls while exploring reasoning trajectories.

In contrast, IndexRAG shifts cross-document reasoning to indexing time, making implicit connections retrievable through standard vector search without additional inference-time overhead.

[1] https://github.com/circlemind-ai/fast-graphrag

3 Method

3.1 Overview

IndexRAG follows a two-phase pipeline: offline indexing and online inference. During offline indexing, we first extract atomic knowledge units (AKUs) and entities from each document (**Stage 1**), then generate bridging facts that capture cross-document reasoning by linking documents through shared entities (**Stage 2**). Both AKUs and bridging facts are encoded and stored in a unified vector store, making cross-document reasoning directly accessible through standard vector search. During online inference, we perform single-pass retrieval with balanced context selection and prompt the LLM for answer generation. The overall pipeline is illustrated in Figure 2 and detailed below.

3.2 Offline Indexing

3.2.1 Stage 1: AKU and Entity Extraction

Given a corpus of n documents $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, we prompt an LLM to extract a set of atomic facts, structured as question-answer pairs, and associated entities from each document d_i. We retain only the answers, merging them into a single text unit per document as the minimal retrievable unit. We refer to this unit as an *atomic knowledge unit* (AKU), denoted a_i. The LLM simultaneously extracts a set of entities $E_i = \{e_1^{(i)}, e_2^{(i)}, \ldots\}$. Each AKU a_i is then encoded by a dense embedding model f and stored in a flat vector store $\mathcal{V}$. Empirically, the choice of extraction method in Stage 1 has a limited effect on overall pipeline performance (Section 5.3.1). Nevertheless, the AKU representation aggregates atomic facts rather than preserving raw passages with unrelated context, providing finer-grained retrieval units similar to propositions (Chen et al., 2024).

3.2.2 Stage 2: Bridging Fact Generation

The key observation motivating this stage is that documents sharing common entities often contain complementary information. While each document alone may only provide partial evidence, their combination can yield reasoning conclusions not explicitly stated in any single source. By making these implicit connections explicit and retrievable, Stage 2 enables cross-document reasoning without any additional inference-time processing.

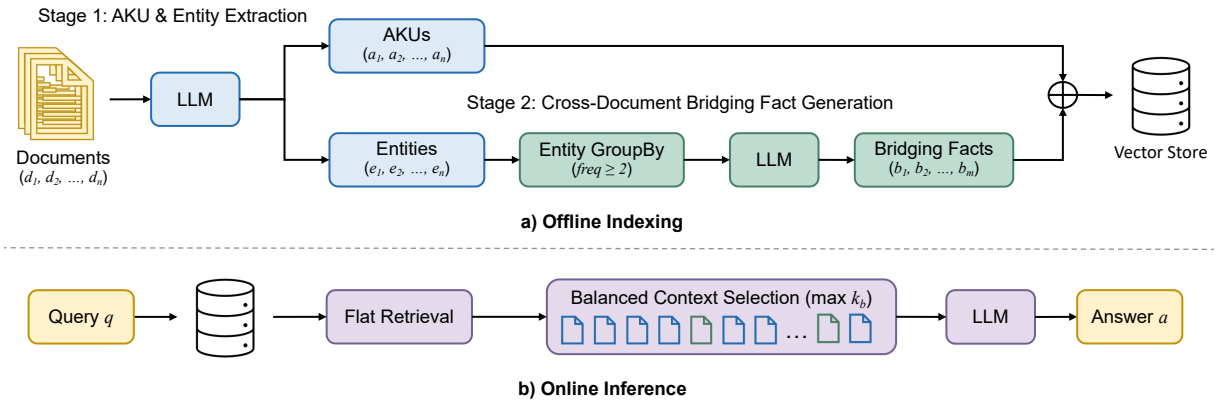


Figure 2: Overview of IndexRAG. (a) Offline Indexing: In Stage 1, the LLM extracts AKUs and associated entities from each document. In Stage 2, documents sharing the same entity are identified and used to generate bridging facts. The resulting AKUs and bridging facts are encoded and stored in a unified vector store. (b) Online Inference: the user query q is encoded and used to retrieve relevant context. After applying balanced context selection with k_b, the selected context (a mix of AKUs and bridging facts) is fed to the LLM to generate the answer a.

Bridge Entity Identification. We aggregate all entities from Stage 1 and identify *bridge entities* that appear across multiple documents:

$$\mathcal{E}_{\text{bridge}} = \left\{ e \in \bigcup_{i=1}^{n} E_i \;\middle|\; 2 \leq \text{df}(e) \leq \tau \right\} \quad (1)$$

where $\text{df}(e) = |\{d_i \in \mathcal{D} : e \in E_i\}|$ is the document frequency of entity e, and τ is an upper-bound threshold. The lower bound ensures that bridge entities connect at least two documents, while the upper bound τ excludes overly generic entities (e.g., common nouns or high-frequency terms) that would produce trivial bridging facts.

Bridging Fact Generation. For each bridge entity $e \in \mathcal{E}_{\text{bridge}}$, let $\mathcal{D}_e = \{d_i \in \mathcal{D} : e \in E_i\}$ be the set of documents mentioning e. We collect the subset of facts in each AKU a_i that mention entity e, denoted $a_i[e]$, identified via string matching against the entity list from Stage 1. We then prompt the LLM to generate bridging facts:

$$\mathcal{B}_e = \text{LLM}\left(e,\ \{a_i[e] : d_i \in \mathcal{D}_e\}\right) \quad (2)$$

where $\mathcal{B}_e$ is the resulting set of bridging facts. We limit the number of source documents to 5 and facts per document to 8 to manage input length.

Unlike cross-document summaries, which compress existing content into shorter passages, bridging facts are constructed to directly answer implicit cross-document questions. Figure 1(b) illustrates this. Given Doc A stating "Aylwin is directed by Henry Edwards" and Doc B stating "Henry Edwards was born in Weston-super-Mare", IndexRAG generates the bridging fact "The director of the film Aylwin was born in Weston-super-Mare". Unlike either source document, this fact directly encodes the two-hop reasoning chain and is semantically aligned with queries like "Where was the director of Aylwin born?", making it retrievable via standard vector search even though neither source document alone is sufficient to answer the query.

All bridging facts $\mathcal{B} = \bigcup_{e \in \mathcal{E}_{\text{bridge}}} \mathcal{B}_e$ are individually encoded by f and stored alongside the AKUs in $\mathcal{V}$. When a new document d_{new} is added, only Stage 1 for d_{new} and Stage 2 for affected bridge entities need to be re-executed. This includes both existing bridge entities that appear in d_{new} ($E_{\text{new}} \cap \mathcal{E}_{\text{bridge}}$) and newly formed bridge entities whose document frequency reaches 2 after the addition. The rest of the index remains unchanged.

3.3 Online Inference

Given a query q, we encode it with the same embedding model f and retrieve the top-k entries from $\mathcal{V}$ by cosine similarity, where $v_j \in \mathcal{V}$ is either an AKU or a bridging fact.

Balanced Context Selection. The retrieved top-k set typically contains a mix of AKUs and bridging facts. However, because bridging facts are considerably shorter than AKUs on average (166 vs. 634 characters in our experiments), they tend to dominate the top-k, crowding out the longer AKUs.

To control this, let $R = \{r_1, r_2, \dots\}$ be all entries in $\mathcal{V}$ sorted by descending similarity to q, where each r_j is either an AKU ($r_j \in \mathcal{A}$) or a bridging fact ($r_j \in \mathcal{B}$). We greedily build the context C by iterating over R in order. Each entry is

Dataset	Questions	Passages
HotpotQA	1,000	9,827
2WikiMultiHopQA	1,000	6,262
MuSiQue	1,000	9,723

Table 2: Dataset statistics for the 1,000 question subsets used in our experiments.

included if it is an AKU or if the number of bridging facts already in C is below k_b, until $|C| = k$. The procedure is summarized in Appendix B.

The selected context C is concatenated in similarity order and provided to the LLM for answer generation (see Figure 2b), without any additional inference-time overhead.

4 Experimental Setup

4.1 Datasets

We evaluate on three widely-used multi-hop QA datasets: **HotpotQA** (Yang et al., 2018), **2WikiMultiHopQA** (Ho et al., 2020), and **MuSiQue** (Trivedi et al., 2022).

HotpotQA contains 113k questions requiring reasoning over multiple Wikipedia passages. 2WikiMultiHopQA provides multi-hop questions across different types (comparison, bridge comparison, compositional, and inference) with structured reasoning paths. MuSiQue consists of 25k 2-4 hop questions composed from single-hop sub-questions, and is generally considered the most challenging among the three. Following prior work (Press et al., 2023; Trivedi et al., 2023; Gutiérrez et al., 2024; Gutiérrez et al., 2025), we sample 1,000 questions from each validation set and collect all associated passages (supporting and distractor) as the retrieval corpus. Dataset statistics are summarized in Table 2.

4.2 Implementation Details

We use GPT-4o-mini [2] as the LLM for all stages and all methods, including baselines, to ensure a fair comparison. All text is encoded using OpenAI's text-embedding-3-small [3], and we use FAISS (Douze et al., 2026) for vector indexing. Prompt templates for all stages and answer generation are provided in Appendix H.

[2] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[3] https://developers.openai.com/api/docs/models/text-embedding-3-small

	HotpotQA	2Wiki	MuSiQue
Bridge entities	8,878	4,817	6,471
Bridging facts	12,608	6,821	8,010
Non-empty rate	85%	88%	79%

Table 3: Bridging fact generation statistics.

Offline Indexing. In Stage 1, we pass each full document to the LLM for AKU and entity extraction. In Stage 2, we set the entity frequency threshold $\tau = 10$, with a maximum of 5 source documents and 8 facts per document for each bridge entity. The offline pipeline requires one LLM call per document for Stage 1 and one LLM call per bridge entity for Stage 2, with the total Stage 2 cost proportional to $|\mathcal{E}_{\text{bridge}}|$. Entities with no meaningful cross-document connection are discarded, accounting for the non-empty rates in Table 3. For baseline methods that require chunking (Naive RAG, RAPTOR, etc.), we split documents into passages of approximately 100 words following Chen et al. (2024), with an 80-character overlap in our implementation.

Online Inference. All methods retrieve 20 candidates and select the top $k = 10$ for answer generation. For IndexRAG, balanced context selection with $k_b = 3$ is additionally applied to control the proportion of bridging facts in the selected context. Unless otherwise specified, this value is used throughout rather than tuned per dataset.

Since all evaluated datasets expect short-form answers (typically a few words), we set the generation temperature to 0 and max tokens to 50. For IRCoT experiments, we adopt the interleaving procedure of Trivedi et al. (2023), with 3 reasoning steps and top-20 retrieval per step chosen for our setup. When combined with IndexRAG, bridging facts are included in the retrieval corpus and balanced context selection is applied at the final answer generation step, with $k_b = 3$.

4.3 Baselines

We compare IndexRAG against baselines spanning five categories: (1) **Sparse retrieval:** BM25 (Robertson and Walker, 1994), (2) **Dense retrieval:** Naive RAG with fixed-size chunking using the same embedding model, (3) **Iterative approach:** IRCoT (Trivedi et al., 2023), (4) **Graph-based:** FastGraphRAG [4] and HippoRAG2 (Gutiérrez et al., 2025), and (5) **Hierarchical:** RAPTOR

[4] https://github.com/circlemind-ai/fast-graphrag

	HotpotQA			2WikiMultiHopQA			MuSiQue			Average		
	EM	Acc	F1	EM	Acc	F1	EM	Acc	F1	EM	Acc	F1
BM25	47.2	51.2	60.3	31.6	32.5	35.9	9.8	10.7	19.2	29.5	31.5	38.5
Naive RAG	50.2	54.0	63.6	42.2	44.5	47.7	19.0	20.4	29.9	37.1	39.6	47.1
FastGraphRAG	50.0	53.8	63.5	**49.5**	**54.8**	**57.4**	17.5	18.8	27.2	39.0	42.5	49.4
RAPTOR	50.2	54.1	63.6	42.3	44.2	47.8	19.3	20.3	29.7	37.3	39.5	47.0
IndexRAG	**53.9**	**59.3**	**68.9**	44.8	50.3	51.7	**22.4**	**24.4**	**34.4**	**40.4**	**44.7**	**51.7**
HippoRAG2	**56.5**	**60.7**	**70.5**	50.2	55.7	57.2	23.8	25.5	34.7	43.5	47.3	54.1
IRCoT	49.8	54.9	62.5	39.7	41.3	43.1	11.1	12.3	19.3	33.5	36.2	41.6
IRCoT + IndexRAG	54.5	59.7	68.7	**53.4**	**61.2**	**61.2**	**24.6**	**26.5**	**35.0**	**44.2**	**49.1**	**55.0**

Table 4: Multi-hop QA performance (%) on three multi-hop QA benchmarks. Best single-call results per dataset are in **bold**. Best multi-call results are in **bold gray**. Methods requiring multiple LLM calls at inference time are greyed out.

(Sarthi et al., 2024). We adopt FastGraphRAG as a representative graph-based baseline due to its improved efficiency over GraphRAG (Edge et al., 2025) while maintaining competitive performance. HippoRAG2 embeds the query to retrieve candidate triples, filters them with an LLM, and then runs Personalized PageRank over the graph, resulting in additional online processing beyond flat vector search. Note that IRCoT was originally evaluated with BM25, so we re-run it with the dense retriever used throughout our experiments to keep the comparison controlled. To demonstrate the generality of Stage 2, we also report IRCoT combined with bridging facts.

4.4 Evaluation Metrics

We evaluate using three metrics: Exact Match (EM), Accuracy (Acc), and F1 score, all computed after answer normalization following Rajpurkar et al. (2016). In addition, we report average retrieval latency and the number of LLM calls at inference time to assess efficiency. Formal definitions are provided in Appendix A.

5 Results

5.1 Quantitative Results

5.1.1 Multi-hop QA Performance

Table 4 presents the main results across the three multi-hop QA benchmarks. We report F1 score as the primary metric as it captures partial credit for partially correct answers.

Single-call methods. Among methods requiring only a single LLM call at inference time, IndexRAG achieves the highest average scores across all three metrics. IndexRAG obtains an average F1 of 51.7, improving over Naive RAG (+4.6), BM25 (+13.2), FastGraphRAG (+2.3), and RAPTOR (+4.7). On HotpotQA and MuSiQue, IndexRAG ranks first across all metrics, with particularly strong gains on MuSiQue (34.4 vs. 29.9 for Naive RAG), where all baselines achieve their lowest scores across the three datasets. On 2WikiMultiHopQA, FastGraphRAG achieves the best single-call performance (57.4). Comparison and bridge comparison questions, which account for 44% of this dataset, require parallel entity lookups that align naturally with explicit graph structures. IndexRAG's bridging facts are generated along sequential reasoning paths and provide limited benefit for these question types, as further analyzed in Section 5.3.3. Despite this, IndexRAG achieves the best average F1 among all single-call methods.

Method	EM	Time(s)	Calls
Naive RAG	19.0	0.29	1.0
FastGraphRAG	17.5	2.55	1.0
RAPTOR	19.3	0.47	1.0
IndexRAG	**22.4**	0.30	1.0
HippoRAG2	23.8	3.13	2.0
IRCoT + IndexRAG	24.6	1.08	3.2

Table 5: Quality-efficiency comparison (%) on MuSiQue. Gray rows indicate multi-call methods.

Multi-call methods. When combined with IRCoT, IndexRAG achieves the best average performance among all evaluated methods. IRCoT + IndexRAG reaches an average F1 of 55.0, slightly exceeding HippoRAG2 (54.1) and well above standalone IRCoT (41.6). Adding bridging facts to IRCoT improves performance by 13.4 points over IRCoT alone on average, with the largest gain on 2WikiMultiHopQA (+18.1), suggesting that index-time bridging facts complement query-time iterative retrieval in this setting. HippoRAG2 achieves the highest single-dataset score on HotpotQA (70.5), but requires LLM-based triple filtering and Personalized PageRank traversal at query time, resulting in significantly higher retrieval la-

Query: Where was the director of the film Aylwin born? **Gold Answer:** Weston-super-Mare	
Naive RAG	
Retrieved	[1] Aylwin is a 1920 British silent drama film directed by Henry Edwards... [2] Jim Wynorski (born August 14, 1950) is an American screenwriter... [3] Frank Launder (28 January 1906 – 23 February 1997) was a British writer, film director...
Generated	✗ Henry Edwards
IndexRAG	
Retrieved	[1] Aylwin is a 1920 British silent drama film directed by Henry Edwards... *(AKU)* [2] Henry Edwards directed both Aylwin and In the Soup, showcasing his career... *(bridging fact)* [3] Henry Edwards, born on 18 September 1882 in Weston-super-Mare... *(bridging fact)*
Generated	✓ Weston-super-Mare

Table 6: Case study from 2WikiMultiHopQA, analyzed in Section 5.2.1. Only the top-3 results are shown, with the decisive fact highlighted in green.

tency, as detailed in Section 5.1.2.

5.1.2 Quality-Efficiency Comparison

Table 5 presents the quality-efficiency trade-off on MuSiQue. Among single-call methods, IndexRAG achieves the best EM (22.4) with a retrieval latency of 0.30 seconds, nearly identical to Naive RAG while improving EM by 3.4 points. FastGraphRAG, despite requiring only a single LLM call, requires 2.55 seconds per query, 8.5× slower than IndexRAG while achieving lower EM (17.5). Among multi-call methods, HippoRAG2 achieves a slightly higher EM of 23.8 but at the cost of two LLM calls and a retrieval latency of 3.13 seconds, over 10× that of IndexRAG. IRCoT + IndexRAG achieves the best EM of 24.6 with a retrieval latency of 1.08 seconds, still 2.9× faster than HippoRAG2 at the retrieval stage, although it averages 3.2 LLM calls per query and we do not measure end-to-end latency including generation.

These results show that IndexRAG achieves strong performance with minimal online overhead by shifting part of the computation to indexing. To make this trade-off explicit, Appendix D reports indexing cost and storage overhead. Compared with HippoRAG2, IndexRAG produces a 17–20× smaller index across the three datasets and reduces offline indexing cost by 30–50%. Within IndexRAG, Stage 2 (bridging fact generation) contributes only 14–18% of the total offline cost.

5.2 Qualitative Results

5.2.1 Case Study

To illustrate how bridging facts improve multi-hop retrieval, we examine a concrete example from 2WikiMultiHopQA (Table 6). The query "Where was the director of the film Aylwin born?" requires two-hop reasoning: (1) identifying the director of the film Aylwin as Henry Edwards, and (2) determining his birthplace. The relevant information is spread across two documents, requiring cross-document reasoning to connect the film to the director's biographical details.

Method	EM	Acc	F1
Chunking	50.2	54.0	63.6
Summary	51.7	55.4	66.0
QA extraction	**53.2**	**57.4**	**67.2**

Table 7: Stage 1 extraction method comparison (%) on HotpotQA without bridging facts.

For both methods, we present the top-3 retrieved results. Naive RAG retrieves the correct film passage as its top-ranked result, which identifies Henry Edwards as the director of Aylwin. However, the remaining two passages contain unrelated directors. The passage containing Edwards' birthplace exists in the vector store but falls outside the top-k, causing the LLM to output the director's name rather than his birthplace. IndexRAG retrieves the same film passage as its top-ranked result. However, the remaining two slots are filled by bridging facts generated during offline indexing: one connecting Edwards to his directing career and another linking him to his birthplace in Weston-super-Mare. With this cross-document context directly available, the LLM correctly answers the query.

This example demonstrates that bridging facts do not merely summarize existing content but create new retrievable units that connect information across document boundaries, making evidence that fell outside the top-k directly accessible through standard vector search.

Configuration	EM	Acc	F1
Naive RAG	19.0	20.4	29.9
+ Stage 2	**22.3**	**24.4**	**34.4**
QA Extraction	18.1	20.1	30.1
+ Stage 2	**22.4**	**24.4**	**34.4**

Table 8: Stage 2 bridging facts added on top of two different Stage 1 representations (%) on MuSiQue.

Dataset	Datastore	Recall@10	EM
HotpotQA	QA extraction	66.4	53.2
	+ Bridging	57.5	**53.9**
2Wiki	QA extraction	50.5	41.7
	+ Bridging	46.7	**44.8**
MuSiQue	QA extraction	62.0	18.1
	+ Bridging	52.8	**22.4**

Table 9: Recall@10 vs EM (%) with and without bridging facts.

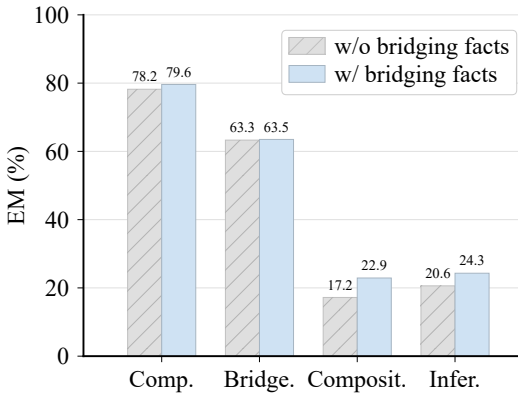


Figure 3: Fine-grained EM (%) on 2WikiMultiHopQA by question type. Results are shown with and without bridging facts.

5.3 Ablation Study

5.3.1 Effect of Each Stage

Stage 1: Extraction Method. Table 7 compares three Stage 1 strategies on HotpotQA without bridging facts. Naive chunking yields the lowest performance with an F1 of 63.6. Summarization provides a modest improvement (+2.4) over naive chunking by compressing each document before indexing. QA extraction achieves the best results across all metrics, improving F1 by 3.6 over chunking and 1.2 over summarization. Structuring document content as question-answer pairs appears to yield denser, more query-aligned units. Each AKU encodes one piece of answerable information rather than raw document text. QA extraction is therefore our default Stage 1 strategy, used throughout unless otherwise specified.

Stage 2: Compatibility with the Base Representation. Table 8 tests whether the gain from Stage 2 depends on the Stage 1 extraction method. On MuSiQue, bridging facts added to Naive RAG and to QA extraction reach nearly the same final score (22.3 and 22.4 EM, respectively), so the improvement appears independent of which Stage 1 representation is in the index. This is practically important, since Stage 2 can be applied on top of an existing RAG system without modifying the underlying indexing pipeline.

5.3.2 Recall vs. End-to-End Performance

Table 9 examines the relationship between passage recall and QA performance when bridging facts are added to the vector store. Across all three datasets, adding bridging facts consistently reduces Recall@10 yet improves EM, with the largest EM gain on MuSiQue (+4.3), followed by 2WikiMultiHopQA (+3.1) and HotpotQA (+0.7), suggesting bridging facts are most beneficial for harder datasets requiring more cross-document reasoning.

The recall drops because bridging facts compete with original passages for retrieval slots. Since bridging facts are newly generated by the LLM, they do not correspond to any original passage and therefore contribute no gain to passage-level recall. Despite this, the consistent EM improvement across all three datasets suggests that the cross-document reasoning encoded in bridging facts is more valuable for answering multi-hop questions than the original passages they replace. Appendix F quantifies this. When bridging facts are present in the index but withheld from the prompt, F1 stays within 1 point of Naive RAG. Adding them to the prompt raises F1 by 4.5 to 5.3 points.

5.3.3 Performance Across Question Types

We break results down by question type to see which kinds of reasoning benefit. Of our three datasets, 2WikiMultiHopQA has the most fine-grained type labels, so Figure 3 reports its four types (comparison, bridge comparison, compositional, and inference). The largest gains from bridging facts appear on compositional (+5.7) and inference (+3.7) questions, both of which require combining evidence from separate documents to derive an answer. This aligns with the design of bridging facts, which explicitly encode cross-document connections.

Comparison questions also benefit from bridging facts (+1.4), as they often require retrieving infor-

mation of two entities from different documents. Bridge comparison questions show minimal improvement (+0.2). These questions require two reasoning chains in parallel, each resolving a piece of information (e.g., birthdate) for a different entity before the two are compared. For instance, "Which film has the director born later, X or Y?" requires resolving the director and birthdate for both X and Y separately. Since our bridging facts are generated along sequential reasoning paths, they tend to cover only one of the two parallel paths, providing limited benefit for this question type.

These results suggest that bridging facts are most effective for question types that require synthesizing evidence across documents, such as compositional and inference questions. The relatively smaller gains on comparison and bridge comparison questions, which together account for 44% of the dataset, partially explain why IndexRAG does not lead the single-call methods on 2WikiMultiHopQA.

6 Conclusion

We present **IndexRAG**, a novel approach that shifts cross-document reasoning from online inference to offline indexing. During indexing, IndexRAG generates bridging facts that capture entity-level connections across documents, stored alongside AKUs in a unified vector store. At inference time, a single retrieval pass and a single LLM call suffice to produce the final answer. Extensive experiments on three multi-hop QA benchmarks show that IndexRAG achieves the best average F1 among single-call methods, outperforming the strongest baseline by 2.3 F1 points. The cross-document work happens once at index time, so the online cost stays that of flat vector search. Retrieval is 6.6–10.4$\times$ faster than HippoRAG2, on an index 17–20$\times$ smaller and cheaper to build. IndexRAG also composes with iterative retrieval. Adding bridging facts to IRCoT raises its average F1 to 55.0, the best result among all evaluated methods. An ablation shows that bridging facts help mainly as direct evidence read by the generator rather than as hints that reorder retrieval.

Future work could explore bridging fact generation conditioned on question type to handle diverse reasoning patterns, and higher-order bridging structures that compose several connections for longer reasoning chains. Adapting k_b per query and predicting which bridge entities are worth expanding could further improve quality and reduce indexing cost. More broadly, other reasoning that does not depend on the query could move to index time.

Limitations

IndexRAG targets multi-hop questions whose evidence can be connected through shared entities, and its benefits are concentrated there. Gains are smaller on comparison-style questions (Ho et al., 2020), which call for parallel lookup rather than a hidden intermediate entity (Section 5.3.3), and they shrink with hop depth, becoming negligible at four hops (Appendix G). Outside the multi-hop regime the benefit is small: on single-hop Natural Questions (Kwiatkowski et al., 2019) bridging facts add 0.1 F1 over an index of AKUs alone (Appendix E).

IndexRAG also relies on LLM-based preprocessing during offline indexing, including bridge entity extraction and bridging fact generation. Missed or noisy entities and hallucinated bridging facts may introduce irrelevant retrieval units and hurt performance. Dedicated NER models, source attribution, or verification mechanisms could improve robustness. Our evaluation is limited to English multi-hop QA benchmarks under a pre-indexed corpus setting with clean text queries. On these same benchmarks, multi-hop retrieval has been found to amplify rather than absorb upstream transcription errors (Bao, 2026), so robustness to noisy input is an open direction. Generalization to other languages, domains, and dynamic corpora also remains unexplored.

References

Zhenghua Bao. 2026. Better retrieval, worse robustness: How multi-hop rag amplifies upstream asr errors. *Preprint*, arXiv:2608.22872.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. 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, Miami, Florida, USA. Association for Computational Linguistics.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. *Mem0: Building Production-Ready AI Agents with Scalable Long-Term Memory*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pages 2993–3000. IOS Press.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2026. The faiss library. *IEEE Transactions on Big Data*, 12(2):346–361.

Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. 2024. Moshi: a speech-text foundation model for real-time dialogue. *Preprint*, arXiv:2410.00037.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From local to global: A graph rag approach to query-focused summarization. *Preprint*, arXiv:2404.16130.

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

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 21497–21515. PMLR.

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, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Chuanrui Hu, Xingze Gao, Zuyi Zhou, Dannong Xu, Yi Bai, Xintong Li, Hui Zhang, Tong Li, Chong

Zhang, Lidong Bing, and Yafeng Deng. 2026. EverMemOS: A self-organizing memory operating system for structured long-horizon reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 45836–45853, San Diego, California, United States. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. 2025. Memory OS of AI agent. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25961–25970, Suzhou, China. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah 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, Singapore. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, page 232–241, Berlin, Heidelberg. Springer-Verlag.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

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, Toronto, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Shijia Xu, Zhou Wu, Xiaolong Jia, Yu Wang, Kai Liu, and April Xiaowen Dong. 2026. Self-correcting RAG: Enhancing faithfulness via MMKP context selection and NLI-guided MCTS. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 20953–20976, San Diego, California, United States. Association for Computational Linguistics.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 17577–17604. Curran Associates, Inc.

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, Brussels, Belgium. Association for Computational Linguistics.

Xiaozhou You, Yahui Luo, and Lihong Gu. 2025. MS-RAG: Simple and effective multi-semantic retrieval-augmented generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22609–22625, Suzhou, China. Association for Computational Linguistics.

Dong Zhang, Shimin Li, Xin Zhang, Jun Zhan, Pengyu Wang, Yaqian Zhou, and Xipeng Qiu. 2023. SpeechGPT: Empowering large language models with intrinsic cross-modal conversational abilities. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15757–15773, Singapore. Association for Computational Linguistics.

A Evaluation Metrics

Let $\hat{a}$ and a^* denote the normalized prediction and gold answer, and let $t(\cdot)$ denote the token sequence of a string and $T(\cdot)$ its set of distinct tokens. Exact Match (EM) measures whether the prediction is identical to the gold answer:

$$\text{EM} = \begin{cases} 1 & \text{if } \hat{a} = a^* \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Accuracy (Acc) measures whether the gold answer appears as a substring of the prediction:

$$\text{Acc} = \begin{cases} 1 & \text{if } a^* \subseteq \hat{a} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

F1 computes the token-level harmonic mean of precision and recall, counting each matching token once in the numerator and all tokens in the denominators:

$$P = \frac{|T(\hat{a}) \cap T(a^*)|}{|t(\hat{a})|}, \quad R = \frac{|T(\hat{a}) \cap T(a^*)|}{|t(a^*)|} \tag{5}$$

$$F_1 = \frac{2PR}{P+R} \tag{6}$$

B Balanced Context Selection

Algorithm 1 presents the full procedure for balanced context selection, which controls the proportion of bridging facts in the retrieved context.

Algorithm 1 Balanced Context Selection

$C \leftarrow \emptyset$, $n_b \leftarrow 0$
for r_j in R **do**
  if $|C| = k$ **then**
    break
  end if
  if $r_j \in \mathcal{A}$ **or** $n_b < k_b$ **then**
    $C \leftarrow C \cup \{r_j\}$
    if $r_j \in \mathcal{B}$ **then**
      $n_b \leftarrow n_b + 1$
    end if
  end if
end for
return C

C Effect of Balanced Context Selection

The parameter k_b controls the maximum number of bridging facts included in the retrieved context. Table 10 shows results for $k_b \in \{0, 1, 2, 3, 5\}$, where

k_b	HotpotQA	2Wiki	MuSiQue
0 (no bridging)	53.2	41.7	18.1
1	52.7	43.3	21.6
2	**54.1**	43.2	22.4
3	53.9	**44.8**	**22.4**
5	52.9	44.4	22.3

Table 10: Balanced retrieval ablation (EM, %).

Dataset	Latency (s)	Index (MB)	Cost ($)
HotpotQA	–	144 / 2657	4.43 / 6.37
2Wiki	0.44 / 2.89	83 / 1426	2.60 / 5.22
MuSiQue	0.30 / 3.13	114 / 2278	3.67 / 6.56

Table 11: End-to-end cost. Each cell is IndexRAG / HippoRAG2. Latency is retrieval time per query, and was not measured on HotpotQA. Cost is recorded directly from OpenAI API responses over the full retrieval corpus of each dataset using gpt-4o-mini pricing.

$k_b = 0$ corresponds to retrieval without bridging facts. Performance peaks at $k_b = 2$ on HotpotQA and MuSiQue (with MuSiQue tying at $k_b = 2$ and $k_b = 3$), and at $k_b = 3$ on 2WikiMultiHopQA, then declines as k_b increases further. This is because bridging facts are shorter than AKUs on average (166 vs. 634 characters), so including too many bridging facts displaces original context, which contains more informative content. We observe that datasets requiring more cross-document reasoning, such as MuSiQue, tolerate a higher k_b.

D End-to-End Cost Accounting

Index-time reasoning trades online cost for offline cost, so both sides need accounting. Table 11 reports retrieval latency, index size on disk, and offline indexing cost against HippoRAG2 on each dataset.

IndexRAG is cheaper to index than HippoRAG2 despite generating new bridging facts. The bridging-fact generation stage accounts for only 18% of the total IndexRAG offline cost on HotpotQA, 17% on 2WikiMultiHopQA, and 14% on MuSiQue, suggesting that index-time bridging adds little at the margin rather than hiding an expensive preprocessing step.

E Single-Hop Sanity Check

To verify that adding bridging facts does not harm retrieval quality outside the multi-hop regime IndexRAG targets, we run a sanity check on Natural Questions (Kwiatkowski et al., 2019). We sample 1,000 answerable questions from the validation set, retrieve their associated Wikipedia articles from the DPR-preprocessed BEIR/NQ corpus (Karpukhin et al., 2020; Thakur et al., 2021), and merge passages per article, resulting in 924 unique articles after deduplication. This mirrors our closed-corpus multi-hop QA protocol.

Method	EM	F1
Naive RAG	33.2	47.7
AKU-only (no bridging)	35.4	49.6
IndexRAG (AKU + bridging)	**35.9**	**49.7**

Table 12: Single-hop sanity check on Natural Questions.

Method	2Wiki		MuSiQue	
	EM	F1	EM	F1
Naive RAG	42.2	47.7	19.0	29.9
Retrieval-only	41.7	47.2	17.7	29.1
Full IndexRAG	**44.8**	**51.7**	**22.4**	**34.4**

Table 13: Mechanism ablation. Bridging facts remain in the index but are removed from the final prompt in the middle row.

Table 12 shows that IndexRAG does not degrade single-hop performance. The gain over Naive RAG comes primarily from AKU granularity rather than bridging facts: AKU-only retrieval improves EM by 2.2 points, while adding bridging facts contributes a further 0.5 EM. This is expected because bridging facts are designed for cross-document inference, whereas single-hop questions mainly benefit from more focused retrieval units.

F Mechanism Analysis: Direct Evidence or Retrieval Hint?

A bridging fact could help in two ways: the generator may read it as evidence, or its presence in the index may change which AKUs are retrieved. To separate the two, we evaluate a variant that keeps bridging facts in the index, where they still compete for top-k slots, but removes them from the prompt before generation. If the benefit comes from better retrieval, this variant should still beat Naive RAG on F1. If the generator has to read the fact itself, that advantage should vanish.

Table 13 shows the retrieval-only variant staying within 1 F1 of Naive RAG and falling far below full IndexRAG. Putting the facts back into the prompt recovers 4.5 F1 on 2WikiMultiHopQA and 5.3 F1 on MuSiQue. The generator therefore has to read the fact itself, and better retrieval alone accounts for almost none of the improvement.

The same variant also puts a ceiling on what slot displacement costs. Its F1 sits 0.5 below Naive RAG on 2WikiMultiHopQA and 0.8 below on

MuSiQue subset	N	ΔEM	ΔF1
2-hop	535	+6.2	+5.4
3-hop	318	+3.1	+4.5
4-hop	147	+1.4	−0.2

Table 14: IndexRAG improvement by hop depth on MuSiQue.

MuSiQue. The variant differs from Naive RAG in two ways: bridging facts take up top-k slots, and fixed chunks are replaced by AKUs. Those gaps therefore bound displacement rather than measure it. The AKU-only index isolates it, differing from the retrieval-only variant only in whether bridging facts are present. On MuSiQue, F1 falls from 30.1 (Table 8) to 29.1, a drop of 1.0.

G Effect of Hop Depth

We further analyze IndexRAG on MuSiQue by hop depth, using the dataset's N_{hop} field to partition the 1,000 questions into 535 two-hop, 318 three-hop, and 147 four-hop subsets. Table 14 reports the improvement of IndexRAG over the no-bridging variant on each subset.

The benefit decreases with hop depth and is negligible on four-hop questions. This pattern is consistent with the design of bridging facts, which primarily encode pairwise connections between documents sharing an entity. Longer reasoning chains may require composing multiple bridging facts or generating higher-order bridging structures.

H Prompt Templates

We present all prompt templates used in our experiments. Figure 4 shows the Stage 1 AKU extraction prompt, which instructs the LLM to extract factual information from a document as atomic question-answer pairs. Figure 5 shows the summary-based Stage 1 baseline. The LLM is asked to produce a comprehensive summary of each document. For Stage 2, Figure 6 provides documents sharing a common entity and prompts the LLM to generate cross-document bridging facts in JSON format. Figure 7 is the answer generation prompt shared across all methods, requiring the LLM to output concise, exact answers without explanation. Figure 8 presents the IRCoT reasoning prompt, which elicits step-by-step chain-of-thought reasoning and guides the LLM to suggest follow-up search queries when additional information is needed.

[context]
{text}
[/context]
You are an expert information extractor. Extract ALL factual information from the content above as question-answer pairs.
 # Requirements:
- Each FAQ answers exactly ONE question (atomic)
- Questions are natural English ("Who directed X?" not "Director of X")
- Answers are complete and self-contained
- Extract every entity, date, relationship, event, and property
- ALL output in English
 # Output Schema:

```
{"all_faqs": [
   {"question": "...", "answer": "...",
    "entities": ["Entity1", "Entity2"]}
]}
```

 # Example:
Source: "Ed Wood is a 1994 American biographical film directed by Tim Burton, starring Johnny Depp."

```
{"all_faqs": [
  {"question": "Who directed Ed Wood?",
   "answer": "Tim Burton directed Ed Wood.",
   "entities": ["Ed Wood", "Tim Burton"]}
]}
```

Return ONLY valid JSON. Start with `{`, end with `}`. No markdown, no comments.

Figure 4: **Prompt template for Stage 1 AKU extraction.** The `entities` field is the entity set E_i used by Stage 2. The example is shown with one of its four question-answer pairs for brevity.

Given the following document, write a comprehensive summary that captures all key facts, entities, relationships, and details. Be thorough and do not omit important information.
Document: {text}
Summary:

Figure 5: **Prompt template for the Stage 1 summary strategy.**

Given the following information about "{entity}" from multiple source documents, generate bridging facts that connect information across these documents.
{doc_sections}
Requirements:
- Each bridging fact must combine information from 2+ documents
- Be factually accurate — only connect information that is logically related
- Each fact should be self-contained and understandable without context
- Do not generate speculative connections
- If documents share the entity name but are about unrelated topics, return empty
Return a JSON array of strings. If no meaningful connections exist, return [].

Figure 6: **Prompt template for Stage 2 bridging fact generation.**

You are a precise question answering assistant. Answer with ONLY the exact information requested, with no explanations or extra words. If the answer is a name, give only the name. If the answer is a number, give only the number. If the answer is yes/no, give only yes or no.
Context: {context}
Question: {question}

Figure 7: **Prompt template for answer generation (used by all methods).**

You are a reasoning assistant that helps answer multi-hop questions step by step.
Question: {question}
Retrieved Information: {context}
Reasoning so far: {cot_so_far}
Write ONE brief reasoning sentence that makes progress toward answering the question. If more information is needed, suggest a specific search query.
Format your response as:
Reasoning: ⟨one sentence of reasoning⟩
Search: ⟨next search query, or DONE if ready to answer⟩

Figure 8: **Prompt template for IRCoT chain-of-thought reasoning, following Trivedi et al. (2023).**