

Universiteit Leiden

Bridge Reasoning Guided Diffusion: A Framework for Multi-Hop Retrieval-Augmented Generation

Name: Shaowan Liang
Student ID: s4303873

Date: 10/06/2026

Specialisation: Data Science

1st supervisor: Tessa Verhoef
2nd supervisor: Michiel van der Meer

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
2333 CA Leiden
The Netherlands

Abstract

Multi-hop question answering requires combining evidence connected through bridge entities that are not explicitly mentioned in the question. Graph-based retrieval exploits explicit relational structure to traverse from query entities to such indirectly connected evidence. At the same time, modern language models can often infer the bridge entities needed for subsequent reasoning. The remaining challenge is how to propagate these inferred entities to the retriever. Existing iterative methods typically encode them indirectly through rewritten queries or dense representations. This indirect encoding does not preserve bridge entities as explicit graph targets, limiting their influence on subsequent graph propagation. We refer to this limitation as the bridge propagation bottleneck.

We propose Bridge Reasoning Guided Diffusion (BRGD), an iterative GraphRAG framework that directly injects predicted bridge entities into the personalization vector of Personalized PageRank. This converts intermediate reasoning into explicit graph propagation signals rather than another query representation. BRGD further applies degree-adaptive weighting to mitigate signal dilution at high-degree bridge nodes and fuses retrieval rankings across iterations. In effect, each round makes recall from the graph memory both retrospective and predictive: it revisits the evidence retrieved so far and predicts the bridge entities needed for the next hop.

Experiments on MuSiQue and 2WikiMultiHopQA show that BRGD consistently improves both retrieval and question answering over non-iterative and iterative baselines, with larger gains on questions requiring deeper reasoning chains. These improvements generalize across sparse NER graphs, dense OpenIE graphs, and different diffusion operators, suggesting that the gains arise from bridge propagation rather than a specific graph construction or retrieval algorithm. Recovery and echo-loop analyses further indicate that BRGD enables more stable iterative retrieval.

Acknowledgements

This thesis grew out of my interest in how reasoning can actively shape the graph knowledge of RAG systems. That question motivated the idea developed in this work. It lies at the intersection of large language models and information retrieval. Exploring this question and gradually developing it into a clearer research problem has been a rewarding process, which taught me as much about doing research as about the topic itself.

I would like to thank my supervisors, Tessa Verhoef and Michiel van der Meer, for their guidance and feedback throughout this thesis. Their support gave me a sense of reassurance and steadiness as I explored a new research direction. I am especially grateful for their patience in helping me turn initially scattered ideas into a clearer understanding of the research problem, bringing theoretical depth and intellectual vitality to this work.

I am also very grateful to my family and friends for their constant support throughout my studies. I am equally grateful to my cat, whose companionship brought comfort and warmth to many busy days.

Looking back, I realize that I placed many constraints on the directions I allowed myself to pursue. I was often guided by what seemed technically feasible and by a narrow sense of what academic research should look like. These constraints made my exploration less open than it might otherwise have been. In my future work, I hope to give myself more time to step back and reflect, allowing curiosity and courage to guide more of what I do.

Contents

Chapter 1 Introduction **6**

Chapter 2 Background and Related Work **9**
2.1 Retrieval-Augmented Generation and the Retrieval Bottleneck 9
2.1.1 Dense Retrievers: Training and Scale 9
2.1.2 Knowledge Graphs: Structure, Application, and Trade-offs 10
2.1.3 Personalized PageRank: Memory Diffusion over Graphs 10
2.2 Query Rewriting and Rank Fusion 11
2.3 Multi-Hop QA Benchmarks: Differences in Difficulty 11

Chapter 3 Methodology **13**
3.1 Overview 13
3.2 Preliminary: Personalized PageRank 14
3.3 Bridge-Prediction Guided Retrieval 15
3.4 Graph Construction and Single-Pass RAG Retrieval 17
3.5 Algorithm and Complexity 19

Chapter 4 Experiments **21**
4.1 Overview 21
4.2 Datasets 21
4.3 Evaluation Protocol and Baselines 22
4.4 Implementation 23
4.4.1 Offline Indexing 23
4.4.2 Models, Prompts, and Inference 26
4.4.3 Hyperparameters and Reproducibility 26

Chapter 5 Results **28**
5.1 Main Results 28
5.1.1 Overall Performance 28
5.1.2 Hop-breakdown Performance 29
5.1.3 Graph-Agnosticity 29
5.2 Ablation Study 30
5.3 Mechanistic Comparison of BRGD and ITER-RETGEN on NER Backbone 31
5.4 Efficiency 33
5.5 Cross-query Bridge Memory 35
5.6 Case Study 36

Chapter 6 Discussion and Conclusion **39**
6.1 Discussion 39
6.1.1 Bridge Propagation as Soft Retrieval Guidance 39
6.1.2 Bridge Propagation Beyond Bridge Discovery 39
6.1.3 Long-chain Retrieval Challenge in Bridge Propagation 40
6.2 Conclusion 40

Chapter A Prompts **42**
A.1 Passage named-entity extraction (NER graph) 42

A.2 OpenIE triple extraction (OpenIE graph) 42
A.3 Bridge prediction 43
A.4 Reader (question answering) 44

Chapter 1 Introduction

Retrieval-augmented generation (RAG) has become a standard paradigm for knowledge-intensive tasks, integrating LLM outputs with external evidence to improve factual accuracy. For single-hop question answering, relevant passages often share lexical overlap with the query, allowing single-pass dense retrieval (Karpukhin et al., 2020) to achieve strong performance. Multi-hop QA, however, requires composing information across multiple documents connected by implicit *bridge entities*–intermediate concepts that do not appear in the original question but are necessary for reaching the answer. For example, answering "When was the city where the composer of *Scanderbeg* was born abolished?" requires identifying the composer (Antonio Vivaldi), his birth city (Venice), and then the event that abolished Venice. Because the question and the final evidence are connected only through intermediate entities, a single retrieval pass is often insufficient to recover the complete reasoning chain.

Graph-based methods such as HippoRAG 2 (Gutiérrez et al., 2025), GraphRAG (Edge et al., 2024), and RAPTOR (Sarthi et al., 2024) construct an offline entity-relation graph over the corpus and retrieve through graph traversal or graph diffusion[1]. This supports multi-hop retrieval because passages can be reached through intermediate entities even when they share no direct overlap with the query. However, because the graph is extracted from surface text by automatic pipelines, it captures only relations explicitly expressed in the corpus. Semantically related entity pairs whose connection requires inference, paraphrase resolution, or cross-document aggregation often remain disconnected. The resulting graph is therefore both semantically sparse and static. More importantly, once graph diffusion begins, information discovered during reasoning cannot directly modify the retrieval process.

Reasoning-based methods such as IRCoT (Trivedi et al., 2023), ITER-RETGEN (Shao et al., 2023), and Self-Ask (Press et al., 2023) instead interleave retrieval with reasoning, conditioning each retrieval step on the partial reasoning trace via question decomposition or query rewriting. This dynamic adaptation allows the retrieval target to evolve as the reasoning unfolds, often recovering evidence that the original query alone cannot surface. Crucially, these methods still reason over free text: the reasoning trace is text, and it reaches the retriever only as a rewritten query or a dense embedding. This text-based mediation is fragile in two ways. First, because the reasoning history accumulates as free text, errors compound in iterations. As the running context lengthens, the models respond less reliably to evidence in the middle of the input (Liu et al., 2024); and after a wrong answer enters the trace, later turns stay conditioned on it and extend the early error rather than correcting it, so the loop seldom recovers (Laban et al., 2025). A misretrieval in round t thus reshapes the round-$(t+1)$ query and pulls the retriever away from the gold evidence. Second, the reasoning signal is communicated only through rewritten text or dense embeddings. Intermediate entities discovered during reasoning must be re-encoded into a retrieval query, making it difficult for retrieval to directly act on them. When the reasoning is compressed into a fixed, lower-dimensional dense vector, a single bridge entity becomes one signal among many and can be diluted, so retrieval may miss it even when the reasoning has already named it.

[1]Here *diffusion* refers to graph diffusion–the propagation of relevance over the graph structure (in our method, Personalized PageRank)–not diffusion-based generative models.

The two paradigms address complementary aspects of multi-hop retrieval. Graph retrieval provides structured connectivity but limited adaptivity, while iterative reasoning provides adaptivity but limited control over graph retrieval. In many cases, an LLM can already infer useful bridge entities that point toward the missing evidence. However, existing systems provide limited mechanisms for translating these inferred entities into retrieval behavior. As a result, information discovered during reasoning often fails to meaningfully influence subsequent retrieval. We refer to this limitation as the *bridge propagation bottleneck*.

Recent work has begun to combine reasoning and graph retrieval more tightly. Reasoning-on-graph methods such as GraphReader (Li et al., 2024), ToG (Sun et al., 2024), and HopRAG (Liu et al., 2025) treat the LLM as a graph navigator that selects which nodes to expand during retrieval. Related multi-agent retrieval frameworks (Nahid and Rafiei, 2025) further distribute the retrieval process across specialized agents responsible for decomposition, filtering, and evidence aggregation. These approaches improve adaptivity, but typically require many sequential LLM calls and incur substantial inference cost.

A closely related line of work, BDTR (Guo et al., 2025), identifies a related *bridge promotion bottleneck*: bridge documents are retrieved but ranked too low to support subsequent reasoning. BDTR addresses this at the ranking stage, through dual-query generation and an additional LLM-based reranking pass, yet its reasoning signal still reaches the graph only as dense scores and rewritten queries. We instead intervene during graph diffusion, injecting the inferred bridge entity directly as a diffusion seed, without introducing an additional reranking stage. As BDTR is evaluated under a different retrieval setup, we contrast the two conceptually rather than empirically.

To address this bottleneck, we propose **Bridge Reasoning Guided Diffusion (BRGD)**, an iterative GraphRAG framework. At each iteration, an LLM examines the currently retrieved evidence and predicts a small set of bridge entities that may connect the current evidence to unexplored regions of the graph that could hold the missing evidence. These bridge entities are resolved to graph nodes and injected directly into the personalization vector of Personalized PageRank (PPR) for the next retrieval round. In this way, reasoning does not merely rewrite the query; it guides graph diffusion toward new regions of the graph through a set of bridge entities.

Unlike prior iterative methods such as IRCoT, ITER-RETGEN, and BDTR, which communicate reasoning signals through rewritten queries and dense embeddings, BRGD converts reasoning outputs into explicit retrieval targets. The retrieval process therefore remains anchored to the original query while dynamically propagating newly discovered bridge entities through graph diffusion. Because bridge entities provide structured retrieval signals rather than an accumulating free-text trace, reasoning can influence subsequent retrieval without relying solely on increasingly long reasoning histories. Our recovery and echo-loop analyses later examine this behavior (Section 5.3). The method requires only a graph with resolvable entity nodes and is therefore applicable to both sparse NER graphs and semantically richer OpenIE graphs.

We evaluate BRGD on MuSiQue (Trivedi et al., 2022) and 2WikiMultiHopQA (Ho et al., 2020) using both a sparsity-augmented NER graph and the OpenIE-based HippoRAG graph. Without any model fine-tuning, BRGD consistently outperforms both non-iterative and iterative GraphRAG baselines. The gains are particularly pronounced on deeper reasoning chains. Beyond overall performance improvements, our analysis reveals that the

gains arise primarily from retrieval rather than answer generation. BRGD recovers failed retrieval states more effectively, explores a broader set of retrieval trajectories, and reduces repetitive retrieval behavior across iterations.

Our contributions are threefold:

1. **We identify a bridge propagation bottleneck in iterative GraphRAG systems.** Although existing methods can often discover useful bridge entities through reasoning, these entities are typically communicated through rewritten queries and dense embeddings. We show that bridge entities discovered during reasoning often fail to effectively influence subsequent retrieval on graphs. The key challenge is therefore not bridge discovery itself but bridge propagation.
2. **We propose Bridge Reasoning Guided Diffusion (BRGD), a reasoning-guided graph diffusion framework for multi-hop retrieval.** BRGD converts reasoning outputs into explicit graph diffusion seeds. Instead of generating answers during iterative retrieval, it predicts a small set of bridge entities that guide graph diffusion toward unexplored evidence. This target can often be derived from partial evidence even when the answer is still unsupported. A degree-adaptive weighting keeps high-degree bridge entities[2] from being diluted during diffusion. Experiments on MuSiQue and 2WikiMultiHopQA show that BRGD consistently improves retrieval and question-answering performance on both sparse NER graphs and semantically richer OpenIE graphs.
3. **We provide a systematic analysis of reasoning-guided graph retrieval.** Through analyses across graph constructions, reasoning depths, retrieval failures, and bridge behaviors, we show how explicitly propagating bridge entities changes the interaction between reasoning and graph retrieval, leading to more effective iterative retrieval across diverse graph backbones.

[2]The *degree* of a node is the number of other entities it connects to, so high-degree bridge entities are hubs linked to many other entities in the graph.

Chapter 2 Background and Related Work

2.1 Retrieval-Augmented Generation and the Retrieval Bottleneck

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) couples the LLM's parametric memory with a non-parametric external corpus accessed by a retriever: it retrieves relevant passages from the corpus and conditions the LLM's generation on them. The motivation is that purely parametric models cannot easily expand or revise their stored knowledge and may hallucinate, whereas retrieval gives access to knowledge that is updatable and inspectable.

When evidence is distributed across multiple passages, two bottlenecks arise. The first is the retrieval bottleneck: the evidence needed to answer is dispersed across several passages, and retrieval must gather it. The second is the multi-document reasoning bottleneck: once the evidence is in hand, the model must still comprehend, compare, and aggregate it across documents to synthesize an answer. This work focuses on the retrieval bottleneck—gathering evidence that spans passages when the query cannot reach it directly. Addressing the retrieval bottleneck requires mechanisms for representing, accessing, and expanding external knowledge. Furthermore, retrieval usually succeeds when a structured semantic connection can be established between the query and external knowledge.

The retrieval corpus is ultimately stored as raw text, over which the classical approach retrieves by lexical overlap (e.g., BM25; Robertson and Zaragoza, 2009). To move beyond surface-term matching, external knowledge is commonly re-represented in one of two ways, which are the focus of this work: dense representations in a vector space, and knowledge graphs built from entity relations. In addition to both, query augmentation uses LLM reasoning to expand the retrieval horizon, drawing retrieval toward distant passages beyond the surface reach of the query. Their effectiveness can be evaluated through multi-hop QA benchmarks, which require retrieval across multiple pieces of evidence. This chapter provides the necessary background for them, establishing the terminology and mathematical foundations for the method in Chapter 3 and the experimental comparisons in Chapter 4.

2.1.1 Dense Retrievers: Training and Scale

Dense retrieval maps queries and documents into a shared embedding space and ranks documents according to embedding similarity. DPR (Karpukhin et al., 2020) established the standard dual-encoder framework trained with contrastive learning and in-batch negatives. Subsequent work has explored both unsupervised contrastive pre-training (Contriever; Izacard et al., 2022) and large-scale encoders with billions of parameters (NV-Embed; Lee et al., 2025).

Alongside this scaling trend, lightweight sentence encoders prioritize efficiency. A representative example is all-MiniLM-L6-v2, a distilled MiniLM model (Wang et al., 2020) built on the Sentence-BERT framework (Reimers and Gurevych, 2019), containing approximately 22M parameters. Owing to its favorable efficiency–performance trade-off, it is widely adopted in practical retrieval systems.

This work focuses on methodological differences rather than retriever scaling. Therefore,

all experiments use all-MiniLM-L6-v2 as a shared encoder to isolate the effects of the proposed retrieval mechanism.

2.1.2 Knowledge Graphs: Structure, Application, and Trade-offs

In contrast to dense retrievers, which compress each passage into a single vector, knowledge graphs (KGs) model the structural information of a corpus through nodes and their relations. The nodes typically correspond to entities; edges encode the relations between them. This representation links different passages through shared entities. The corpus as a whole forms a global semantic network.

Knowledge graphs offer several advantages for retrieval tasks (Hogan et al., 2021). Because shared entities and connecting edges form a traversable network across passages, retrieval can reach distant nodes that share no surface overlap with the query. The edges encode entity relations explicitly, so both the retrieval results and the paths leading to them remain traceable and analyzable. This structure is also directly compatible with graph algorithms: graph diffusion (e.g., Personalized PageRank, introduced in the next section), graph traversal, and pattern matching all apply, and it supports symbolic and statistical reasoning over relations.

These advantages come with a price. The nodes and edges of a knowledge graph are produced by offline extraction pipelines, and the quality and coverage of extraction directly determine the graph's usability. Maintenance and update are also expensive. When new passages enter the corpus or existing facts change, updating the graph typically requires re-running extraction, indexing, and embedding. Efficient incremental updates remain challenging, and maintenance costs typically scale with corpus size.

2.1.3 Personalized PageRank: Memory Diffusion over Graphs

Personalized PageRank is a variant of the classical PageRank algorithm (Page et al., 1999). The original PageRank models the web's link structure as a directed graph. It measures each page's importance by the stationary distribution of a random walk on this graph. All pages are treated equally. Jeh and Widom (2003) generalized this formulation by allowing arbitrary personalization vectors. The generalized form is now called Personalized PageRank (PPR).

PPR has since been widely applied to entity relevance ranking, graph representation learning, recommendation systems, and community detection. As a canonical graph diffusion algorithm over knowledge graphs, PPR propagates efficiently from a set of seed nodes. This differs from graph traversal, which reaches distant nodes by walking the graph hop by hop. The resulting distribution reflects the global connectivity around these seeds.

HippoRAG (Gutiérrez et al., 2024) introduces PPR in RAG to simulate diffusion-based association in human knowledge retrieval. In HippoRAG, the knowledge graph serves as the long-term memory store, analogous to the hippocampal index in human memory.

However, human memory retrieval is constructive (Schacter and Addis, 2007) and strategic (Moscovitch, 1992; Moscovitch et al., 2016). It does not passively replay the stored long-term memory. Instead, it actively reassembles memory fragments under top-down prefrontal control. Inspired by this view, our method treats LLM reasoning as a form of

top-down control in RAG memory. We replace HippoRAG's static partial cue set with a dynamic set that reasoning can reshape. At each round, bridge entities identified by LLM reasoning are added to the current query entities. In this analogy, graph diffusion performs pattern completion: from a partial set of cue entities, it recovers the wider set of associated evidence. The next round's completion then runs from an updated cue set–the original query entities together with the bridge entities reasoning has added (see Chapter 3).

2.2 Query Rewriting and Rank Fusion

While graph-based retrieval improves structural connectivity, query reformulation improves retrieval by modifying the query itself. To bridge the semantic gap between a query and relevant passages, query reformulation rewrites or expands the query. Early work expands the query with generated content (GAR; Mao et al., 2021), while more recent work uses an LLM to generate a pseudo-document for retrieval–a hypothetical document in HyDE (Gao et al., 2023) or an expansion passage in Query2doc (Wang et al., 2023).

In order to explore and exploit various rewritten queries, RAG-Fusion (Rackauckas, 2024) generates K parallel queries from the original query in a single LLM call and performs retrieval with each query independently. The retrieved rankings are subsequently combined using Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), which preserves strong recall: a document found relevant under any single perspective is retained in the fused ranking. These K queries are mutually independent: each is conditioned only on the original query, and none can see any retrieval result or the other queries' outputs. The method therefore lacks a global, cross-round reasoning view. Rackauckas (2024) further notes that when the generated queries drift too far from the original, some answers go off topic.

This work also combines query rewriting with RRF, but adds such a global view. Rewriting happens inside a multi-round reasoning–retrieval loop with reasoning traces. Each round's rewritten query is conditioned on the previous round's reasoning trace. This ensures exploring new queries while preserving and exploiting the recalled documents of earlier rounds.

2.3 Multi-Hop QA Benchmarks: Differences in Difficulty

Table 1: Comparison of the multi-hop QA benchmarks.

Benchmark	Reasoning Depth	Difficulty
MuSiQue	Up to 4	High
2WikiMultiHopQA	Up to 2	Medium
HotpotQA	Up to 2	Low

The three benchmarks differ primarily in reasoning depth, which determines the difficulty of retrieval, as summarized in Table 1. By *reasoning depth* we mean retrievals that must be performed in order, each using the answer of the previous one. This is distinct from the number of supporting documents: a question may rest on several documents, but if those documents are retrieved independently (in parallel), they do not form a longer dependent chain.

MuSiQue (Trivedi et al., 2022) is composed, bottom-up, of up to four dependent single-hop questions (the composition forms a directed acyclic graph, for which the paper gives six shapes), so answering it can require up to four dependent retrievals in sequence. An anti-shortcut filter removes questions that can be answered without using the intermediate answers. MuSiQue is the highest in terms of difficulty.

2WikiMultiHopQA (Ho et al., 2020) contains four question types—compositional (45.2%), comparison (30.1%), bridge-comparison (20.8%), and inference (3.9%)—about 70% of which involve a bridge entity. Its inference and compositional questions are built from two triples (two dependent steps) and rest on two supporting documents; the comparison rests on two; bridge-comparison first locates a bridge entity and then compares, and rests on four (Ho et al., 2020). Although it can have up to four supporting documents, its longest chain of dependent retrievals is two: the four-document bridge-comparison compares two parallel two-step chains. Moreover, its bridge entity is usually a Wikipedia entity described by one of the supporting documents, and can be recovered directly from them. 2Wiki is therefore easier for deep-hop retrieval than MuSiQue.

HotpotQA (Yang et al., 2018) provides two supporting documents per question and is often treated as a multi-hop benchmark. However, prior work shows that a substantial fraction of its questions do not actually require multi-hop reasoning (Min et al., 2019; Chen and Durrett, 2019), making it the easiest of the three.

Consequently, our main experiments focus on MuSiQue and 2WikiMultiHopQA, which place substantially greater demands on bridge-prediction conditioned retrieval. HotpotQA is included primarily as a reference point.

Chapter 3 Methodology

3.1 Overview

We study multi-hop question answering over a corpus. Given a question q and a corpus $\mathcal{D}$ of passages d, the retriever aims to recover as many supporting passages as possible, from which the reader model produces the answer. The supporting passages are linked by bridge entities that are absent from q. A single similarity match, therefore, hardly retrieves all of them.

Our retrieval substrate is a graph-based retriever. Personalized PageRank (PPR; Section 3.2) operates over a knowledge graph G built from the corpus, following the graph-based RAG line (Gutiérrez et al., 2025). Offline, an LLM extracts the entity pairs and their relations to construct G. At query time, q selects a set of seed nodes, and PPR diffuses from these seeds to rank passages. We instantiate G under two constructions of differing semantic density: our lightweight NER construction (Section 3.4), and the OpenIE triples of HippoRAG 2. Our method applies to both. As noted in Chapter 2, this diffusion is efficient but static: the seeds are fixed once per query, and reasoning cannot redirect the diffusion.

In this paper, we make the diffusion reasoning-guided. Rather than modifying the graph itself, BRGD intervenes in the personalization vector that governs graph diffusion. BRGD proceeds in iterative rounds. In each round t, the LLM reads the passages retrieved this round, predicts the bridge entities B_t that connect the next hop, and rewrites the query. The predicted bridges are added to the personalization vector as new seeds, so the next diffusion re-allocates probability mass toward the bridge neighborhood. By predicting a *set* of candidate bridge entities rather than a single deterministic next step, the diffusion propagates from multiple candidate bridges at once and avoids committing prematurely to one reasoning path.

Here $\mathbf{s}_t$ is the personalization vector of PPR (Section 3.2)—a non-negative distribution over graph nodes that sets where the diffusion restarts. It combines two channels: a fixed query channel $\mathbf{s}^q$ that anchors diffusion to the question, and a bridge channel $\mathbf{s}_t^b$ that carries the bridge entities discovered so far. Specifically, each round is

$$D_t = \mathrm{PPR}(G, \mathbf{s}_t), \tag{1}$$

$$(B_t, q_t) = \mathrm{LLM}(c_t), \tag{2}$$

$$\mathbf{s}_{t+1} \propto \mathbf{s}^q + \mathbf{s}_t^b, \tag{3}$$

where $\mathbf{s}_0 = \mathbf{s}^q$. The trace c_t accumulates information through round t, including the round's ranked passages D_t. The three lines denote diffusion, bridge prediction with query rewriting, and dual-channel re-seeding, respectively.

After T rounds, the per-round rankings $\{D_t\}$ are fused by Reciprocal Rank Fusion and passed to a reader LLM for answer generation. Fusing across rounds helps to explore different query variants and limits the impact of a single round's error. Figure 1 shows the full pipeline. The remainder of this chapter is organized as follows. Section 3.2 reviews PPR and the three limitations that motivate our design. Section 3.3 details bridge-prediction-guided retrieval. Section 3.4 describes the two graph constructions, and presents the process of single-pass retrieval. Section 3.5 gives the full algorithm and complexity analysis.

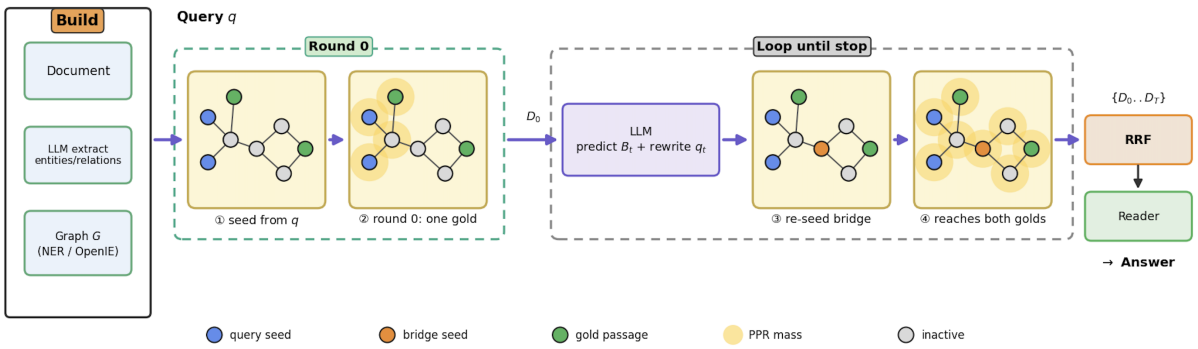


Figure 1: **Overview of BRGD.** An LLM builds a knowledge graph G offline. Round-0 Personalized PageRank from the query seeds (blue) reaches only the gold supporting passages in the query's cluster. Each loop round then has the LLM predict a bridge entity (orange) and rewrite the query; injecting the bridge as a new seed re-shapes the diffusion to reach both gold passages (green). Per-round rankings are fused by RRF for the reader; halos mark PPR mass.

3.2 Preliminary: Personalized PageRank

Personalized PageRank (PPR) (Jeh and Widom, 2003) assigns each node of a graph a relevance score with respect to a set of seed nodes. It is the stationary distribution $\mathbf{p}$ of a random walk that, at each step, teleports to a *personalization vector* $\mathbf{s}$ with probability α and otherwise transitions along an outgoing edge:

$$\mathbf{p} = (1-\alpha)\, M^{\top}\mathbf{p} + \alpha\, \mathbf{s}, \tag{4}$$

where M is the column-stochastic transition matrix of the graph. The personalization vector $\mathbf{s} \in \mathbb{R}^{|V|}$ is a non-negative, ℓ_1-normalized distribution over the node set: each entry s_i is the probability of restarting to node i. Its support defines the seed set, and the corresponding values allocate the restart mass among the seeds. Because the teleport term repeatedly re-injects mass at the seeds, the stationary distribution $\mathbf{p}$ concentrates on the seeds and their multi-hop neighborhoods; $\mathbf{s}$ therefore governs where the diffusion is initiated and localized. In our setting, $\mathbf{s}$ places mass on nodes identified from the query, and then on the bridge entities predicted by the LLM (Section 3.3). The resulting distribution $\mathbf{p}$ therefore ranks the passages according to their graph proximity to these seeds.

PPR is well suited to multi-hop retrieval for two reasons. First, a single diffusion propagates mass from the seeds across the entire connected component, covering multi-hop evidence in one pass. Second, the diffusion is obtained by sparse power iteration: since each node links to only a few others, the transition matrix M is sparse, with mostly zero entries, so each iteration is a cheap sparse matrix-vector product, repeated until convergence. It issues no language-model call, making it efficient and training-free.

PPR, however, exhibits three limitations that our method targets. First, the diffusion is *static*: once $\mathbf{s}$ is fixed for a query, the walk remains anchored to the same seeds throughout, and reasoning cannot redirect it. Second, it is *purely structural*: the walk is agnostic to node semantics, so the retrieval quality is bounded by seed quality. Third, its stationary mass is influenced by *node degree*: the diffusion may over- or under-allocate probability

mass according to a seed's connectivity, so informative bridge seeds are not always assigned proportional influence. These limitations motivate the three components of our method in Section 3.3: we re-seed the diffusion with reasoning-predicted bridge entities to address the first two, and apply a degree-adaptive re-weighting to the bridge seeds to counter the third. We isolate the contribution of this re-weighting by ablation in Section 5.2.

3.3 Bridge-Prediction Guided Retrieval

We have shown that, with the knowledge graph fixed, the retrieval quality under PPR depends on its seed selection: if a query's relevant entities are absent from the round-0 seed set, the diffusion cannot reach the evidence behind them. This is common in multi-hop questions. Dense similarity to the query is a poor proxy for a bridge's importance to the reasoning chain. A bridge that does not resemble the question is scored too low to be seeded. Even a seeded bridge can receive a weight far below the distractor entities that match the question surface, so its signal is drowned out. A purely dense-to-graph seeding thus underserves the very entities that connect the hops. We therefore make the seed set a function of reasoning and run retrievals across rounds. Round 0 begins with the single-pass retrieval of Section 3.4: the query selects a seed vector $\mathbf{s}^q$, and PPR returns the passage ranking D_0. In each round t, the LLM analyzes the retrieved evidence, predicts the bridge entities, and rewrites the query. The predicted entities enter the personalization vector as new seeds, and the new diffusion yields D_t. We detail one round below.

Bridge-prediction guided reasoning. Each round t, the LLM conditions on the accumulated trace c_t–the question, the past reasoning traces and discovered bridges, and the passages retrieved (D_t)–and returns a set of bridge entities B_t and a rewritten query q_t through a fixed prompt (Figure 2; full text in Appendix A): $(B_t, q_t) = \mathrm{LLM}(c_t)$. Bridge discovery is both retrospective and predictive: it explains the gap between the currently retrieved evidence and query, while anticipating the entities required to reach the next hop. By design the prompt targets single-step recall rather than the final answer: each round asks only for the entities that advance retrieval by one hop–a target the LLM can supply from partial evidence–rather than an answer the retrieved evidence may not yet support.

Unlike dense retrieval, which compresses reasoning into the query text, we have the LLM list these entities explicitly. Each entity is resolved to a graph node by lexical or embedding matching, and entities that cannot be resolved from the graph are discarded. In practice this discarding is rare: on MuSiQue, 99.5% of the bridge entities predicted across all rounds resolve to a graph node. This high rate likely reflects that the graph entities and the predicted bridges are produced by the same LLM, which names entities consistently. This conversion from textual reasoning output to discrete graph nodes constitutes the key interface between reasoning and graph diffusion. At every round, the LLM explicitly decides a region in which the next diffusion expands on the graph. The decision is formed as discrete seed nodes rather than text that would dilute or bias the signal.

Dual-channel seeding with degree-adaptive weights. To allow reasoning-derived bridge entities to directly influence graph retrieval while preserving the semantic meaning of the original query, bridge seeds and query seeds are injected into the personalization vector in parallel, forming a dual-channel seed: $\mathbf{s}_t \propto \mathbf{s}^q + \mathbf{s}^b_t$, where $\mathbf{s}^b_t = \sum_{v \in B_{1:t}} w(v)\, \mathbf{e}_v$, with $B_{1:t} = \bigcup_{i \le t} B_i$ the bridges discovered through round t. The query channel $\mathbf{s}^q$ keeps

Bridge-prediction prompt (schematic)

Input (c_t)

- Question q
- All earlier reasoning traces $r_{1:t-1}$ and discovered bridges $B_{1:t-1}$
- Passages retrieved this round D_t

Instruction. Analyze the semantic gap between the retrieved evidence and the question; name the bridge entities that would connect the current evidence to the answer; rewrite the query; and decide whether to stop.

Output (JSON)

```
{ "analysis":            "...",
  "discovered_entities": [...],          # B_t
  "rewritten_query":     "...",          # q_t
  "should_stop":         true | false }
```

Figure 2: Schematic of the bridge-prediction prompt; the full text is given in Appendix A.

diffusion anchored to the original question, while the bridge channel $\mathbf{s}_t^b$ steers the retrieval toward regions suggested by the predicted bridge entities.

Bridge entities frequently correspond to highly connected nodes in the graph. As shown in Table 4 and Table 5, both graph constructions exhibit a strongly long-tailed degree distribution, while many of the largest hubs correspond to generic but structurally important connectors such as countries, regions, and dates. These nodes often link otherwise distant regions of the graph, and therefore naturally emerge as bridge candidates. Consequently, an effective bridge mechanism should preserve their influence during diffusion.

However, Personalized PageRank diffusion treats high-degree nodes unfavorably. Recall that the walk transitions through $M^\top = \mathbf{D}^{-1}\mathbf{A}$, the row-stochastic matrix of the PPR update above, where each node distributes its mass equally among its neighbors. Consequently, the larger a node's neighborhood, the less mass each neighbor receives. As a result, the signal originating from a high-degree bridge may be substantially diluted, even when the bridge itself is highly relevant, as identified by the reasoning LLM. In particular, the probability of reaching a specific relevant neighbor–often the one containing the gold supporting passage–can become too weak to meaningfully affect retrieval.

To compensate for this effect, we let the weight of a bridge seed grow logarithmically with its degree:

$$w(v) = w_0 \cdot (1 + \log(\deg(v) + 1)),$$

where w_0 is the base weight. This gives higher-degree bridge seeds a larger share of the fixed bridge-channel mass relative to lower-degree ones, counteracting the degree-based dilution above. Because the personalization vector is normalized, this redistributes mass across seeds by degree rather than amplifying the bridge channel as a whole. Using a logarithm rather than the raw degree keeps the compensation moderate and prevents a small number of hubs from dominating the retrieval process. A single diffusion on $\mathbf{s}_t$ then

shapes the probability mass toward the bridge and produces the ranking of documents D_t.

Cross-round persistence and stopping. To avoid context dilution that a growing reasoning history brings, BRGD does not carry a full textual history across rounds. Instead, the prompt keeps only a compact state–the reasoning trace and the discrete bridge entities discovered so far–so each round's bridge reasoning conditions on what has been found rather than on an expanding transcript.

To handle error accumulation, the LLM reasoning reads the *global fused* top-N retrieval result, not the latest round alone. The per-round rankings are not directly comparable, because each round runs PPR from a different seed vector and yields a different stationary distribution, so we combine them with Reciprocal Rank Fusion, letting each round contribute $1/(k + \text{rank}(d))$ to a document's score, where the constant k dampens low-ranked documents. Reasoning over this fused, global evidence keeps the loop anchored to the original question and lets a bridge found early continue to inform later rounds.

The loop runs for at most T rounds; after each round, the LLM judges from the retrieved passages whether they already contain a complete answer and if so stops. The fused ranking is then passed to the reader.

3.4 Graph Construction and Single-Pass RAG Retrieval

BRGD is designed to operate on graph-based retrieval systems that index the corpus as a graph and retrieve over it. The bridge-prediction mechanism of Section 3.3 imposes a single requirement on this graph: its nodes must be entities that the LLM can name from the retrieved passages and resolve back to seeds. Both NER and OpenIE produce such entity nodes. This property is crucial because bridge prediction operates entirely at the entity level. We therefore construct a light, sparse NER graph, comparing it with the dense OpenIE graph used by HippoRAG. We use the two to test whether the gain from the bridges is independent of the graph. This paragraph first contrasts the two extraction schemes and then describes the construction and single-pass retrieval.

Knowledge Extraction: OpenIE vs. NER. Both schemes remain faithful to the text and infer no new relations between entities, which is precisely what lets the bridge entities predicted in Section 3.3 resolve back to graph nodes. OpenIE extracts relational triples (subject, relation, object): the relation is an open-vocabulary phrase that labels how two entities connect, and the arguments are not restricted to named entities but may be descriptive phrases with modifiers; such extraction is typically performed by an LLM. NER, in contrast, returns only named entities, with no relations and no descriptive phrases, and is light enough to run with either a tagger or an LLM. Because NER yields no relations, we link entities that co-occur within a sentence by an unlabeled edge that merely records their joint mention. Figure 3 contrasts the two for the same passage. The OpenIE graph is semantically denser, containing richer nodes and explicitly labeled relational edges. The NER graph contains only named-entity nodes and unlabeled co-occurrence edges, and therefore captures less semantic information than the OpenIE graph.

Graph construction. The construction turns a flat corpus into a connected graph that models two kinds of association: among entities within a document, and across documents through shared entities. Diffusion can then traverse entities to reach passages several

Knowledge extraction: NER vs. OpenIE (same passage)

Passage. *Radio City is India's first private FM radio station, started on 3 July 2001 ... and launched PlanetRadiocity.com, which offers music news, videos, and songs.*

NER (named entities; edges from same-sentence co-occurrence, *unlabeled*)

```
nodes:  Radio City, India, 3 July 2001, PlanetRadiocity.com, ...
edges:  Radio City  -  India
        Radio City  -  3 July 2001
        Radio City  -  PlanetRadiocity.com
```

OpenIE (subject, relation, object; edges *labeled*)

```
Radio City           -[ is ]->          private FM radio station
Radio City           -[ located in ]->  India
Radio City           -[ started on ]->  3 July 2001
PlanetRadiocity.com  -[ offers ]->      news
```

Figure 3: Knowledge extraction on the same passage. NER produces only named-entity nodes linked by *unlabeled* co-occurrence edges within the same sentence. OpenIE produces *labeled* relational triples whose arguments may be descriptive phrases (e.g., "private FM radio station"), adding both such nodes and labeled edges. The OpenIE graph is therefore denser; the NER graph is sparser.

hops away. Graph-based ranking methods have long represented textual units as graph nodes and semantic associations as weighted edges, then ranked them through PageRank-style propagation. This is exemplified by TextRank (Mihalcea and Tarau, 2004). HippoRAG (Gutiérrez et al., 2024) is a representative GraphRAG system built on Personalized PageRank. Following this paradigm, the NER graph takes entities and passages as nodes and uses three edge types, each capturing one kind of association. (i)Entity–entity co-occurrence edges capture within-document association: two entities co-occurring in a sentence are linked, weighted by their co-occurrence count in the corpus. (ii) Entity–passage edges link each entity to the passages that mention it, so that two passages sharing an entity lie two hops apart; this makes their cross-document association explicit and lets the mass on an entity flow to passages. (iii) Synonymy edges link entities whose embedding similarity exceeds a threshold, merging surface variants of the same entity and raising the connectivity of the graph so that diffusion can reach entities that co-occurrence alone would leave disconnected.

Diffusion seed selection. Round 0 selects the seed entities from the query. For each entity pair $p = (e_1, e_2)$ that co-occurs in a sentence s, we form a single embedding by concatenating both entities with the sentence and encoding the result,

$$e_p = \mathrm{Enc}(e_1 \| e_2 \| s),$$

where $\mathrm{Enc}(\cdot)$ denotes the sentence encoder. The query is encoded as

$$e_q = \mathrm{Enc}(q),$$

and the relevance score is computed as

$$\mathrm{rel}(p, q) = \cos(e_q, e_p).$$

Algorithm 1 Bridge Reasoning Guided Diffusion (BRGD)

Require: query q, graph G, maximum rounds T

$\mathbf{s} \leftarrow \mathbf{s}^q = \textsc{SeedSelect}(q)$; $B \leftarrow \emptyset$; $c \leftarrow q$ ▷ seeds, bridges, reasoning trace

for $t = 0$ **to** T **do**

  $D_t \leftarrow \mathrm{PPR}(G, \mathbf{s})$ ▷ single diffusion

  $(r_t, B_t, q_t, \mathit{stop}) \leftarrow \mathrm{LLM}(c, D_t)$ ▷ reason over trace and retrieved passages

  $c \leftarrow c \| r_t$ ▷ accumulate reasoning trace (global view)

  if *stop* **or** $t = T$ **then break**

  end if

  $B \leftarrow B \cup B_t$ ▷ cross-round persistence

  $\mathbf{s} \leftarrow \mathbf{s}^q + \sum_{v \in B} w(v)\, \mathbf{e}_v$ ▷ dual-channel, degree-adaptive

end for

$R \leftarrow \mathrm{RRF}(\{D_0, \ldots, D_t\})$ ▷ fuse the per-round rankings

answer $\leftarrow \textsc{Read}\big(q,\ \text{top-5}(R)\big)$ ▷ reader on the top-5 fused passages

return *answer*

The top-k pairs by relevance–selected greedily with Maximal Marginal Relevance (MMR; Carbonell and Goldstein, 1998) to limit redundancy among seeds–provide the seed entities; together with a small-weight passage seed and the entities matched directly from the query, they form the seed distribution $\mathbf{s}^q$.

Single-pass diffusion. We then run one PPR (Section 3.2) on the graph with $\mathbf{s}^q$ as the personalization vector, where the three edge types play distinct roles. The mass on the seeds first propagates among entities over the co-occurrence and synonymy edges, and then aggregates onto passage nodes over the entity–passage edges; the stationary scores of the passage nodes give the round-0 ranking D_0. This single pass is also the per-round operator of Section 3.3, with later rounds extending the seed from $\mathbf{s}^q$ to $\mathbf{s}^q + \mathbf{s}^b_t$.

3.5 Algorithm and Complexity

Algorithm 1 summarizes the full procedure. It takes the query q, the graph G, and a round budget T. The personalization vector $\mathbf{s}$ is initialized to the query seeds $\mathbf{s}^q$ (Section 3.4), and the reasoning trace c to the question. At round t, a single PPR returns the passage ranking D_t (Section 3.4); conditioned on the trace c and the retrieved passages D_t, the LLM emits a reasoning step r_t, the bridge entities B_t, a rewritten query q_t, and a stopping signal *stop* (Section 3.3). The reasoning step is appended to c, so the trace accumulates across rounds and gives each round a global view of the reasoning so far. The discovered bridges accumulate in B and re-seed $\mathbf{s}$, each node v entering through its indicator vector $\mathbf{e}_v$ with the degree-adaptive weight $w(v)$ (Section 3.3). The per-round rankings $\{D_0, \ldots, D_t\}$ are finally fused by Reciprocal Rank Fusion into R, from whose top-5 passages a reader produces the answer.

Complexity. The cost of BRGD is dominated by LLM calls rather than graph diffusion. Each retrieval round performs a Personalized PageRank diffusion on a sparse graph, which consists primarily of sparse matrix–vector operations and incurs negligible computational overhead compared with LLM inference. Consequently, the overall runtime scales approximately linearly with the number of reasoning rounds and is largely determined

by the number of LLM calls. Within the loop, BRGD runs at most one reasoning call per round; at the end, a single reader call produces the answer from the top-5 passages of the fused ranking of multiple rounds. Having an early stop judged by the LLM (Section 3.3), the loop can break before reaching the maximum number of rounds. As a result, it empirically requires around three LLM calls per query on average (Chapter 5). The reasoning is performed at a node level–one call predicts a small set of bridge entities for the round. The loop carries only a compact state across rounds, the reasoning trace and the discovered bridges, rather than re-accumulating the retrieved passages.

Chapter 4 Experiments

4.1 Overview

This chapter evaluates BRGD on multi-hop question answering, measuring both its retrieval and its answer quality. To isolate the contribution of the retrieval mechanism itself, all methods share the same encoder, the same answer-generation LLM, and the same corpus, and differ in how they retrieve. The experiments are organized around the following research questions, each addressed in one section of Chapter 5:

- **RQ1 (Main results).** Does BRGD outperform strong non-iterative and iterative baselines on multi-hop retrieval and question answering? (Section 5.1)
- **RQ2 (Graph-agnosticism).** Does the gain hold on both a sparse NER graph and a dense OpenIE graph? (Section 5.1)
- **RQ3 (Component contributions).** How much does each key design component–bridge seeding, degree-adaptive weighting, and rank fusion–contribute? (Section 5.2)
- **RQ4 (Mechanism).** Why does BRGD outperform text-based iterative retrieval? (Section 5.3)
- **RQ5 (Efficiency).** How does BRGD's per-query cost, measured by retrieval rounds and LLM calls, compare to iterative retrieval? (Section 5.4)
- **RQ6 (Cross-query reuse).** Are the discovered bridges reusable across queries, or specific to each query? (Section 5.5)

The remainder of this chapter is organized as follows. Section 4.2 describes the datasets and the retrieval corpus. Section 4.3 presents the evaluation metrics, baselines, and protocol. Section 4.4 details the implementation and hyperparameter settings.

4.2 Datasets

Dataset	#Questions	Corpus	Avg sup/q	2-sup	3-sup	4-sup
MuSiQue	1,000	11,656	2.65	518	316	166
2WikiMultiHopQA	1,000	6,119	2.47	765	–	235

Table 2: Statistics of the evaluation data. The corpus is the deduplicated union of the candidate passages of all 1,000 questions; retrieval is performed within each dataset's own global corpus. The last three columns give the number of questions grouped by the number of supporting passages; for MuSiQue, this number equals the reasoning depth.

We evaluate on MuSiQue and 2WikiMultiHopQA, whose reasoning structure and difficulty were described in Section 2.3. Both the datasets and the retrieval corpora are taken from the evaluation setup released by HippoRAG (Gutiérrez et al., 2024). For each benchmark we use 1,000 questions; Table 2 reports the statistics.

Each dataset has its own global corpus, the deduplicated union of the candidate passages of all its 1,000 questions. Retrieval is performed within this global corpus, not within each question's candidate set: a method must locate the gold passages among all passages in the

dataset rather than among a dozen pre-filtered candidates. For MuSiQue, each question lists about 20 candidates, yet the retrieval ranges over the full corpus of 11,656 passages.

The supporting passages of each question form its gold set for retrieval recall (Section 4.3); Table 2 also reports their distribution. MuSiQue is dominated by deep chains, with around half of the 1,000 questions requiring more than two hops; in 2WikiMultiHopQA, the 235 four-passage questions are all of the bridge-comparison type (Section 2.3). We use this distribution for the hop-wise analysis in Chapter 5.

4.3 Evaluation Protocol and Baselines

Evaluation metrics. We evaluate performance along two dimensions: question answering and retrieval. For question answering, we use Exact Match (EM) and F1. Before scoring, both the predicted answer and the gold answer are normalized by lowercase, with articles and punctuation removed, and collapsing multiple whitespaces. EM measures whether the normalized prediction matches the gold answer exactly. F1 is the harmonic mean of token-level precision and recall between the normalized prediction and the gold answer.

EM and F1 reflect not only answer generation quality but also retrieval quality, since answer generation is conditioned on the retrieved passages. As a result, these metrics evaluate the RAG system from an end-to-end perspective.

For retrieval, we use R@5. For a question with a gold supporting set G and a top-k retrieved list top-k,

$$\mathrm{R@}k = \frac{|G \cap \text{top-}k|}{|G|},$$

the fraction of gold supporting passages retrieved within the top k, macro-averaged over all questions. R@5 reflects the overall coverage of the gold supporting passages. Beyond these aggregate scores, we also report R@k grouped by the number of supporting passages (i.e., the hop count; see Section 4.2), to analyze retrieval performance at different reasoning depths (Chapter 5).

Baselines. We group the compared methods into three categories, all run and evaluated under a unified experimental setup.

Non-iterative (single-round). (i) **Dense**: single-round dense retrieval without any graph, ranking passages by the cosine similarity between the MiniLM embeddings of the query and each passage; (ii) **NER RAG**: a single PPR diffusion on our NER graph (Section 3.4); (iii) **HippoRAG (OpenIE)**: a single PPR diffusion on the OpenIE graph, corresponding to the retrieval backbone of HippoRAG 2 (Gutiérrez et al., 2025).

Iterative. (iv) **IRCoT** (Trivedi et al., 2023): a classic iterative baseline that interleaves chain-of-thought reasoning with retrieval; (v) **ITER-RETGEN** (Shao et al., 2023): a retrieve-and-generate method that rewrites the query with the reasoning generated in each round and retrieves accordingly. For a fair comparison, both are built on the same retrieval backbone as BRGD, so as to isolate the effect of the iterative strategy itself.

Our method. (vi) **BRGD**: run on both the sparse NER graph and the dense OpenIE graph, corresponding to the graph-generalization question in RQ2.

Controlled setup. To isolate the contribution of the retrieval mechanism itself, all methods share the same experimental configuration and differ in their retrieval strategy. Specifically, all methods use the same corpus, the same encoder (all-MiniLM-L6-v2; Wang et al., 2020; Reimers and Gurevych, 2019), and the same answer-generation LLM. For the methods that involve reasoning (IRCoT, ITER-RETGEN, and BRGD), we further fix the same reasoning LLM and the same maximum number of rounds T. After retrieval, every method conditions the same answer-generation LLM on its top-5 retrieved passages to produce the final answer.

All LLM calls use temperature= 0. Under this configuration, retrieval is highly stable across runs: the graph diffusion is deterministic with fixed random seed. So recall is essentially reproducible. Answer-quality metrics (EM and F1) show small run-to-run variation, because the answer-generation LLM is not perfectly deterministic even at temperature= 0–commercial serving backends introduce floating-point and batching nondeterminism–and EM in particular is a strict binary metric that flips on minor wording changes. We therefore report all main experiments from a single run, and for methods with close EM/F1 we discuss the magnitude of the differences in the analysis.

4.4 Implementation

4.4.1 Offline Indexing

System overview. This section describes the implementation of BRGD. The pipeline has two phases. The *offline indexing* phase turns the corpus into a knowledge graph together with its sentence-level index and the associated embeddings. The *online retrieval* phase runs the reasoning-guided diffusion loop of Chapter 3 on top of this index and produces the answer with a reader. The offline extraction, graph construction, and embedding computation, as well as the online seed selection, graph diffusion, and retrieval, are all performed locally; only the steps that require an LLM–entity extraction, bridge reasoning, and the reader–are executed remotely through the DashScope API. All experiments were run on a single workstation with an Apple M2 chip, 16 GB of RAM, and no discrete GPU; graph and retrieval operations use the CPU only.

Graph construction. Graph construction organizes the flat corpus into a connected graph, so that probability mass can propagate between passages through shared entities, and so that the graph exposes entity nodes to which predicted bridges can resolve. Both graphs use entities as nodes, but differ in how they extract and connect them. The *NER graph* (our work) is built by a four-stage pipeline–sentence segmentation, entity extraction, entity matching, and graph construction:

1. *Sentence segmentation*: each passage is split into sentences with a regular expression that handles abbreviations such as Mr./Dr.
2. *Entity extraction*: an LLM extracts named entities from each passage.
3. *Entity matching*: each extracted entity is matched back, by substring, to the sentences that mention it, yielding an entity-to-sentence and an entity-to-passage inverted mapping; entities co-occurring in the same sentence are paired, and each pair records its source sentence.

4. *Graph construction*: with entities and passages as nodes, three edge types are built: entity–entity co-occurrence edges (sentence-level co-occurrence is used as a weak signal of semantic association, weighted by the corpus co-occurrence count), entity–passage edges (weight 1.0), and synonymy edges (a nearest-neighbor search over the entity embeddings, connecting entities whose cosine similarity exceeds a threshold, weighted by that similarity).

The *OpenIE graph* (the comparison backbone) follows HippoRAG 2, and differs from the NER graph in extraction and edge construction: it does not segment sentences, but extracts (subject, relation, object) triples at the passage level with an LLM on top of NER, using the triple arguments as entity nodes and the relations as labeled entity–entity edges; its entity–passage and synonymy edges are built as in the NER graph. The two graphs therefore differ in two respects: the OpenIE graph adds descriptive-phrase nodes from the triple arguments, so it has more nodes, and its entity–entity edges are labeled relations rather than the NER graph's unlabeled co-occurrence.

Graph statistics. Both graphs are built over the same global corpus; their statistics are reported in Table 3. The two graphs are of comparable size: the OpenIE graph has more nodes and the NER graph slightly more edges. In terms of connectivity, the NER graph is more connected, with a higher average degree and more hubs. It has fewer but cleaner named-entity nodes, each recurring across more passages, and it links every entity pair that co-occurs in a sentence. Each entity therefore accumulates many co-occurrence edges.

However, the degree distributions of two graph constructions are heavily long-tailed. The median node degree is only 4–7, but the maximum reaches several thousand (4179 for the NER graph on MuSiQue). The number of hubs also falls off sharply with scale (Table 4): each graph has up to nearly two thousand nodes of degree ≥ 100, but only a handful of degree ≥ 500.

In the NER graph, most hubs are generic connectors–countries, nationalities, and dates–with little discriminative value (Table 5), which is consistent with the fact that NER restricts graph nodes to named entities. By contrast, the OpenIE graph includes a wider variety of concepts beyond standard named entities, such as descriptive role phrases (e.g., "film director"), resulting in richer semantic coverage but also increased noise (e.g., "waste water discharges"). This long-tailed hub structure motivates the degree-adaptive weighting of Section 3.3. Some semantically important bridge nodes may have extremely high degrees, causing their signals to be diluted during diffusion. Degree-adaptive weighting compensates for this effect by preserving bridge influence at high-degree hubs. We isolate its contribution by ablation in Section 5.2.

Table 3: Statistics of the two graphs, built over the same global corpus. Avg/Med/P99/Max are node-degree statistics.

Graph	Dataset	Nodes	Edges	Avg	Med	P99	Max
NER	MuSiQue	74,505	559,257	15.0	6	174	4179
OpenIE	MuSiQue	107,671	546,133	10.1	4	139	2482
NER	2Wiki	45,603	302,730	13.3	7	97	3944
OpenIE	2Wiki	55,159	267,922	9.7	4	83	1223

Table 4: Number of entity hubs by degree threshold. Hub counts fall off sharply with scale: each graph has hundreds to nearly two thousand entity nodes of degree ≥ 100, but only a handful of degree ≥ 500.

Graph	Dataset	≥ 100	≥ 250	≥ 500
NER	MuSiQue	1931	232	47
OpenIE	MuSiQue	1839	130	9
NER	2Wiki	421	89	9
OpenIE	2Wiki	315	10	2

Table 5: Representative entity hubs at four degree scales (the first three match the thresholds of Table 4; the last is a random sample of low-degree entities). Hubs at every scale are generic connectors–countries, nationalities, dates, and, in OpenIE, role phrases–whereas typical low-degree entities are specific named entities, which are what a query targets.

Graph	Dataset	≥ 500	250–500	100–250	< 100 (random)
NER	MuSiQue	United States (4179), Spain (736), 1994 (508)	2000 (485), Argentina (295), 18 Feb 2013 (250)	17 Aug 2007 (249), 14 Apr 1994 (147), 19 Nov 2016 (100)	Hasan Minhaj (7), Olympic football tournaments (3), HOSA (2)
NER	2Wiki	American (3944), United States (689), India (565)	2010 (449), 2001 (313), 1969 (250)	2000 (249), 10 Jul 1987 (121), 13 Aug 2014 (100)	James Cagney (38), Augustus of Saxe-Weissenfels (8), Museum of Modern Glass Art (4)
OpenIE	MuSiQue	United States (2482), India (634), New York City (503)	England (453), 14 Oct 2010 (283), 26 Oct 2010 (250)	11 Jan 2013 (249), 7 Mar 2007 (145), 14 Sep 1952 (100)	Hank Aaron's record (3), waste water discharges (3), Alabama militia (2)
OpenIE	2Wiki	American (1223), film director (500)	screenwriter (415), actor (322), Italian (250)	producer (233), 11 Mar 1977 (118), 8 Feb 2013 (100)	Richard the Lionheart's army (4), Ram Baksh Singh's fort (3), 801 (4)

Sentence-level index for seed selection. In addition to the graph, the NER pipeline maintains a sentence-level index that anchors the query to the sentences that actually mention the relevant entities. The index stores the embedding of each sentence together with the entity-to-sentence inverted mapping above. During seed selection (Section 3.4), the relevance of each candidate entity pair combines the similarity of its own embedding and that of its source sentence to the query: $\mathrm{rel}(p, q) = \alpha_p \cos(e_p, e_q) + \alpha_s \cos(e_{s(p)}, e_q)$, where e_p is the entity-pair embedding, $e_{s(p)}$ the embedding of its source sentence, e_q the query embedding, and $\alpha_p = \alpha_s = 0.5$; MMR then selects a diverse set of seed pairs from the high-scoring candidates. Combining sentence similarity with pair similarity uses the sentence context to disambiguate cases where matching on the entities alone would be ambiguous.

4.4.2 Models, Prompts, and Inference

Models and interface. The whole pipeline uses a single local encoder and a single remote LLM. The encoder is all-MiniLM-L6-v2 (about 22M parameters), runs locally through Sentence-Transformers, and produces all embeddings of passages, sentences, entities, and entity pairs.

All LLM calls use qwen-plus, Alibaba's mid-tier commercial model (positioned between qwen-turbo and qwen-max), accessed through the paid DashScope API. Each LLM call is billed; the LLM calls dominate the pipeline cost (Section 3.5). The LLM call performs four generation tasks–passage named-entity extraction, query key-entity extraction, bridge reasoning, and the reader; the OpenIE triple extraction uses the same model. All calls set temperature to 0 and disable the thinking mode. qwen-plus is a large proprietary model whose parameter count is not publicly disclosed. We call the floating `qwen-plus` alias rather than pinning a snapshot in code. Throughout our experiment window (April to June 2026), this alias resolved to the `qwen-plus-2025-12-01` snapshot, so that snapshot is the model behind all reported runs.

Prompts. The four LLM tasks above each use one fixed one-shot prompt, which together with bridge prediction gives five: passage named-entity extraction (for the NER graph), query key-entity extraction, OpenIE triple extraction (for the OpenIE graph), bridge prediction (sketched in Figure 2), and the reader. All prompts are kept fixed across datasets; their full text is given in Appendix A.

Answer extraction. The reader outputs in the format "Thought: ... Answer: ..."; we take the text after "Answer:" as the predicted answer, with a regular-expression fallback when the output does not follow this format. This step only extracts the answer from the generation and does not change the retrieval.

4.4.3 Hyperparameters and Reproducibility

Hyperparameters. Table 6 lists all hyperparameters. These values are either taken from prior work or fixed on a small development set; none are tuned on the test set.

Reproducibility. The NER, OpenIE, and embedding results are cached and reused across all methods, so that the comparison reflects only the difference in retrieval mechanism. The implementation is in Python and relies mainly on igraph (graph construction and PPR), PyTorch and Sentence-Transformers (embeddings and nearest-neighbor search), the OpenAI Python client (access to qwen-plus through DashScope), and NumPy. All LLM calls use temperature $= 0$ for stable outputs. To support reproduction, we report the model versions, API access dates, and the versions of the main software libraries. As noted in the system overview, graph construction, indexing, embedding computation, and retrieval are all performed locally, and only LLM inference is executed through the API. Our prompts (Appendix A) were developed for qwen-plus. Because the diffusion acts on the entities the model names, applying BRGD with a different reasoning LLM may require adapting these prompts to that model's output format and naming conventions.

Table 6: Hyperparameter settings.

Component	Hyperparameter	Value
Seed selection	MMR λ	0.7
	MMR candidate pool	30
	selected seed pairs	5
	entity pair / sentence weight	0.5 / 0.5
	passage seed weight	0.05
Graph	synonymy threshold	0.8
Diffusion	PPR restart α	0.5
Bridge seeding	base weight w_0	0.5
Iteration	max rounds T	3
Rank fusion	RRF constant k	60
	history / current weight	0.3 / 1.0
Reader	top-k	5

Table 7: Main results on MuSiQue and 2WikiMultiHopQA (1,000 questions each), all using the all-MiniLM-L6-v2 encoder and a top-5 reader. "+ BRGD" is our method, shown on both the semantically sparse NER graph and the semantically rich OpenIE graph. Best per column in bold.

	MuSiQue		2WikiMultiHopQA	
Method	EM	R@5	EM	R@5
Closed-book LLM	.092	–	.141	–
Dense Retriever (no graph)				
+ IRCoT	.232	.454	.333	.597
+ ITER-RETGEN	.324	.592	.505	.746
NER graph (semantically sparse)				
NER (baseline)	.260	.467	.379	.659
+ IRCoT	.262	.483	.499	.677
+ ITER-RETGEN	.352	.616	.555	.837
+ BRGD (ours)	.396	.638	.649	.896
OpenIE graph / HippoRAG 2 (semantically rich)				
HippoRAG (baseline)	.280	.521	.479	.769
+ ITER-RETGEN	.374	.656	.608	.889
+ BRGD (ours)	**.425**	**.679**	**.709**	**.962**

Chapter 5 Results

5.1 Main Results

5.1.1 Overall Performance

Table 7 reports the main results on MuSiQue and 2WikiMultiHopQA. Overall, BRGD achieves the best EM under both graph constructions. On MuSiQue, the EM gains over the strongest iterative baseline are modest (0.352 to 0.396 on the NER graph, 0.374 to 0.425 on the OpenIE graph), and BRGD's larger, more consistent advantage on this benchmark lies in retrieval recall. On 2WikiMultiHopQA the EM gains are substantial (0.555 to 0.649 and 0.608 to 0.709).

These EM gains are accompanied by consistent improvements in retrieval recall: BRGD attains the highest R@5 in most settings, on both datasets. Retrieval gains are generally accompanied by QA gains, although the two can decouple: on MuSiQue, the recall improvements are larger and more consistent than the EM improvements.

BRGD also outperforms existing iterative retrieval methods. Compared with IRCoT and ITER-RETGEN, BRGD achieves higher recall consistently, and higher EM in most settings, under the same experimental conditions.

All methods perform better on 2WikiMultiHopQA than on MuSiQue. For example, BRGD achieves 0.709 EM on 2WikiMultiHopQA but 0.425 EM on MuSiQue. This result is consistent with the benchmark analysis in Section 2.3 and confirms that MuSiQue is the more challenging retrieval benchmark.

Finally, all retrieval-based methods substantially outperform the closed-book LLM baseline. This result highlights the importance of external evidence for multi-hop question answering.

5.1.2 Hop-breakdown Performance

Tables 8 and 9 report performance grouped by the number of supporting passages. It is important to note that the number of supporting passages does not always correspond to reasoning depth (Section 2.3). For MuSiQue, 2-hop, 3-hop, and 4-hop questions form increasingly longer dependent retrieval chains. In contrast, all 4-supporting questions in 2WikiMultiHopQA belong to the bridge-comparison category. Their structure consists of two parallel two-step chains rather than a single deeper chain. Moreover, all such questions are closed-set: the answer is either a yes/no decision or one of the entities explicitly mentioned in the question. Consequently, the EM scores of the 4-hop questions are higher than those of the 2-hop.

On MuSiQue, both EM and R@5 decrease as the retrieval chain becomes longer. Under the OpenIE graph, BRGD's EM drops from 0.535 on 2-hop questions to 0.205 on 4-hop questions, while R@5 decreases from 0.845 to 0.380. Similar trends are observed for all baselines. These results confirm that longer dependent retrieval chains substantially increase retrieval difficulty.

In terms of retrieval recall, BRGD remains the strongest method at every depth, achieving the highest R@5 on the MuSiQue 4-hop subset under both graph constructions. The advantage over ITER-RETGEN is largest at 4-hop: under the OpenIE graph, BRGD achieves 0.380 R@5 versus 0.328 for ITER-RETGEN, and under the NER graph 0.389 versus 0.324.

The 2WikiMultiHopQA results exhibit a counter-intuitive pattern. Owing to the closed-set answer format, the 4-supporting subset achieves substantially higher EM than the MuSiQue 4-hop subset. However, retrieval remains challenging. Even under the OpenIE graph, the HippoRAG baseline reaches only 0.609 R@5. BRGD improves this score to 0.924, substantially outperforming ITER-RETGEN (0.720). Therefore, although the EM scores of this subset are influenced by answer format, the recall results still demonstrate a large retrieval advantage for bridge-guided diffusion.

5.1.3 Graph-Agnosticity

Table 7 shows that BRGD consistently improves performance under both graph constructions. On MuSiQue, BRGD improves EM from 0.260 to 0.396 on the NER graph and from 0.280 to 0.425 on the OpenIE graph. On 2WikiMultiHopQA, the corresponding improvements are from 0.379 to 0.649 and from 0.479 to 0.709. Similar trends are observed for retrieval recall. These results indicate that the benefit of BRGD is not tied to a particular graph representation.

The OpenIE graph achieves stronger overall performance than the NER graph, for both the baselines and BRGD. We attribute this to its richer node set and explicitly labeled relations, although we do not isolate this factor. The advantage is not uniform across depths; on the MuSiQue 4-hop subset, in particular, the two graphs perform comparably. This indicates that retrieval over the fact-based NER or OpenIE graph remains a

bottleneck on this subset.

However, the performance gap between the two graph constructions decreases after applying BRGD. The OpenIE graph remains consistently stronger, but its advantage over the NER graph is smaller than in the corresponding single-pass retrieval setting. This suggests that, by iteratively introducing reasoning-derived bridge entities, the method is able to recover part of the connectivity and semantic information that is not explicitly encoded in the graph itself.

Table 8: Exact Match by reasoning depth (number of supporting passages). MuSiQue has 2/3/4-hop questions; 2WikiMultiHopQA has 2/4-hop. Best per column in bold.

	MuSiQue			2Wiki	
Method	2-hop	3-hop	4-hop	2-hop	4-hop
NER graph (semantically sparse)					
NER (baseline)	.315	.231	.145	.373	.400
+ IRCoT	.365	.193	.084	.495	.511
+ ITER-RETGEN	.442	.329	.114	.549	.574
+ BRGD (ours)	.521	.316	.169	.625	.728
OpenIE graph / HippoRAG 2 (semantically rich)					
HippoRAG (baseline)	.359	.241	.108	.456	.553
+ ITER-RETGEN	.492	.291	.163	.580	.698
+ BRGD (ours)	**.535**	**.361**	**.205**	**.659**	**.872**

Table 9: R@5 by reasoning depth. Best per column in bold. The advantage of the bridge method is largest at 4-hop, where the iterative baselines drop sharply (e.g., NER+IRCoT falls to .350 on 2Wiki) while the bridge method retains high recall.

	MuSiQue			2Wiki	
Method	2-hop	3-hop	4-hop	2-hop	4-hop
NER graph (semantically sparse)					
NER (baseline)	.594	.378	.239	.718	.466
+ IRCoT	.626	.381	.229	.777	.350
+ ITER-RETGEN	.764	.509	.324	.888	.659
+ BRGD (ours)	.790	.520	**.389**	.928	.794
OpenIE graph / HippoRAG 2 (semantically rich)					
HippoRAG (baseline)	.661	.434	.238	.819	.609
+ ITER-RETGEN	.841	.519	.328	.940	.720
+ BRGD (ours)	**.845**	**.559**	.380	**.974**	**.924**

5.2 Ablation Study

Component ablations. Table 10 presents the ablation results on MuSiQue. Removing any component leads to a performance drop, indicating that all three contribute to the final system.

The largest drop comes from removing bridge seeds. EM decreases from 0.396 to 0.358, while R@5 drops from 0.638 to 0.516. Since query rewriting and RRF are still retained, this result shows that bridge-guided diffusion is the primary source of improvement.

RRF fusion also provides consistent gains. Without RRF, EM decreases to 0.370 and R@5 decreases to 0.582. This suggests that different rounds retrieve complementary evidence, and combining rankings across rounds improves overall coverage.

Degree-adaptive weighting has a consistent effect. Replacing adaptive weights with uniform bridge weights reduces EM from 0.396 to 0.373 and R@5 from 0.638 to 0.584. This indicates that rebalancing seed mass by degree, rather than weighting all bridges equally, yields a more effective diffusion.

Table 10: Component ablations on 1000-sample MuSiQue. Each row below the full method removes one component; all use the NER graph and qwen-plus API.

Variant	EM	F1	R@5
NER + BRGD (full method)	.396	.492	.638
− bridge seeds (query rewrite + RRF only)	.358	.445	.516
− RRF fusion (last round ranking only)	.370	.473	.582
− degree-adaptive weights (flat bridge weight)	.373	.470	.584

Diffusion operator. The gain is robust to the choice of diffusion operator. Katz diffusion (Katz, 1953) is an alternative propagation operator: it scores how strongly two nodes are connected by summing over all walks between them, weighting a walk of length ℓ by α^ℓ, so short walks contribute the most and longer ones fade geometrically; with the graph adjacency matrix A, this Katz score is $\sum_{\ell \geq 1} \alpha^\ell A^\ell = (I - \alpha A)^{-1} - I$. Replacing the Personalized PageRank diffusion with Katz diffusion ($\alpha = 0.2$) leaves retrieval and answer quality essentially unchanged (ΔEM < 0.001, ΔR@5 $= 0.003$ on MuSiQue), confirming that the improvement comes from the injected bridge entities, not from the specific propagation operator.

5.3 Mechanistic Comparison of BRGD and ITER-RETGEN on NER Backbone

Win–Loss Analysis. Table 12 compares BRGD and ITER-RETGEN at the question level. On both datasets, BRGD wins substantially more questions than it loses. On MuSiQue, BRGD answers 113 questions correctly that ITER-RETGEN fails, whereas the reverse occurs on only 67 questions. On 2WikiMultiHopQA, the corresponding numbers are 148 and 54. The performance gap therefore reflects a broad improvement across many questions rather than a small number of exceptional cases.

Failure Attribution. To identify the source of this gap, we analyze failure cases. We focus on 2WikiMultiHopQA because its error attribution is more explainable. Even the strongest methods remain retrieval-limited on MuSiQue, making it difficult to separate retrieval errors from reader errors. In contrast, a larger fraction of 2WikiMultiHopQA questions achieve complete retrieval, enabling a cleaner decomposition of failure causes.

	NER+ITER-RETGEN	NER+BRGD
Bridge named in reasoning (coverage)	83.5%	63.1%
Bridge seeded on the graph	26.9%	**64.7%**
Mean seed weight (share)	2.8%	**17.3%**

Table 11: Gold-bridge propagation on MuSiQue (1,000 queries, 1,640 gold bridges). ITER-RETGEN names the gold bridge more often but seeds it on the graph far less, and at a fraction of the weight. The gold bridge resolves to a graph node in 100% of cases, so the gap is one of propagation, not graph coverage.

For the 148 questions where BRGD succeeds and ITER-RETGEN fails, and the 54 questions where ITER-RETGEN succeeds and BRGD fails, we classify the losing method's error into retrieval failure and QA failure. A failure is considered a retrieval failure when not all gold supporting passages are retrieved (R@5<1), and a QA failure when all gold supporting passages are retrieved (R@5$=1$) but the final answer remains incorrect.

Table 13 shows that the majority of failures originate from retrieval rather than answer generation. Among the 148 cases where BRGD wins, 130 (87.8%) of ITER-RETGEN's errors are retrieval failures, compared with only 18 (12.2%) QA failures. Similarly, among the 54 cases where ITER-RETGEN wins, 41 (75.9%) of BRGD's errors are retrieval failures and 13 (24.1%) are QA failures.

The retrieval failures of ITER-RETGEN are particularly revealing. Of the 130 retrieval failures, only 2 fail to produce any bridge entity. The remaining 128 cases already contain a bridge entity but still fail to retrieve the gold supporting passages. This suggests that the main limitation is not bridge discovery itself, but the extent to which the discovered bridge can influence subsequent retrieval.

Bridge Propagation. The failure analysis above indicates that ITER-RETGEN often discovers the bridge yet fails to retrieve on it. We quantify this directly using the *gold* bridge—the annotated intermediate answer of each question—across all 1,000 MuSiQue queries (1,640 gold bridges). For each gold bridge we record whether a method (i) names it anywhere in its reasoning (*coverage*), (ii) places non-zero personalization weight on its graph node (*seeded*), and (iii) how large that weight is as a share of the seed vector (Table 11). ITER-RETGEN actually names the gold bridge *more* often than BRGD (83.5% vs. 63.1%), yet seeds it on the graph far less often (26.9% vs. 64.7%) and at about one-sixth of the weight. The baseline is not failing to *discover* the bridge—if anything it discovers it more—but failing to *propagate* it into the graph walk.[3]

What Bridges Does the LLM Discover? To better understand the role of bridge prediction, we analyze the composition of the bridge entities generated by BRGD. Figure 4 shows that most bridges are neither query entities nor final answers. Instead, they are contextual entities that appear around the reasoning chain. The distribution differs across datasets. On 2WikiMultiHopQA, most bridges lie directly on the gold reasoning

[3]Coverage is a text match against the reasoning trace, whereas seeding is measured on the graph, so the two are not nested: a bridge can be seeded through the base query channel without being named verbatim, which is why BRGD's seeded rate slightly exceeds its coverage.

chain, reflecting its short and explicit multi-hop structure. On MuSiQue, a substantially larger fraction of bridges are contextual entities outside the annotated reasoning chain, including both evidence-related entities and distractors. This observation suggests that bridge prediction is not merely extracting intermediate answers. Rather, it identifies entities that may help redirect retrieval toward useful regions of the graph.

Recovery from Failed Retrieval. Table 14 further examines questions whose round-0 retrieval completely fails (R@5= 0). In this setting, both methods operate on the same graph, use the same diffusion algorithm, and start from the same failed retrieval result. The remaining difference therefore lies entirely in the iterative retrieval process. BRGD consistently exhibits stronger recovery. On MuSiQue, it recovers 69% of failed queries, compared with 55% for ITER-RETGEN. On 2WikiMultiHopQA, the corresponding numbers are 74% and 44%. The same trend holds for complete recovery (R@5= 1).

Escaping Echo Loops. To further characterize the iterative behavior of the two methods, we analyze the similarity between consecutive reasoning traces. Table 15 shows that ITER-RETGEN produces substantially higher overlap across iterations, measured by both bigram overlap and ROUGE-L. In many failure cases, similar reasoning traces lead to highly similar retrieval outcomes. BRGD, by contrast, generates more diverse reasoning trajectories and exhibits substantially lower overlap between rounds.

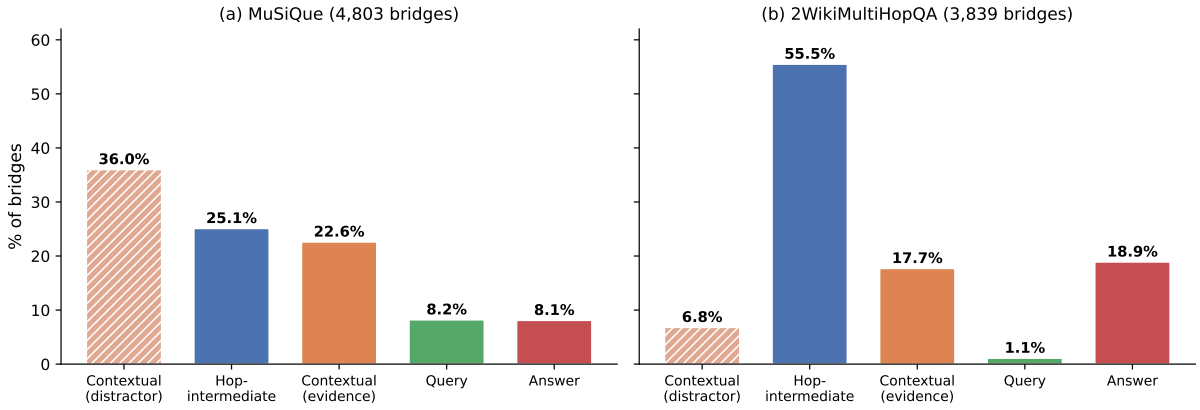


Figure 4: Composition of the bridge entities the LLM discovers. We label each bridge by matching it against the question, the intermediate decomposition answers, and the final answer; entities that match none of these are *contextual*, which we further split into evidence or distractor by whether the entity links to a gold passage in the graph. The two datasets behave differently: on **(a)** MuSiQue many bridges are off-chain distractors, whereas on **(b)** 2WikiMultiHopQA–whose reasoning chains are short and explicit–almost all bridges lie on the chain.

5.4 Efficiency

BRGD and NER+ITER-RETGEN are both iterative, so we compare their per-query cost by the number of LLM calls (Table 16). NER+ITER-RETGEN runs a fixed three rounds for every query and reads the answer from its final generation. BRGD instead stops

Table 12: Per-question outcomes of BRGD vs. ITER-RETGEN (NER backbone). For each question we record which method answers correctly. BRGD wins more often than it loses on both datasets; Table 13 analyses why each method loses.

Dataset	Both correct	BRGD win	ITER win	Both wrong	Net
MuSiQue	285	113	67	535	+46
2WikiMultiHopQA	501	148	54	297	+94

Table 13: Why each method loses on 2WikiMultiHopQA. We take the questions where one method is correct and the other is not, and classify the losing method's failure as a retrieval failure (the gold passages were not fetched) or an answer-generation failure (the gold was fetched but the answer is wrong). On both sides, losses are dominated by retrieval.

Failure cause	ITER-RETGEN fails ($n = 148$, BRGD wins)	BRGD fails ($n = 54$, ITER wins)
Retrieval failure (gold not fetched, R@5<1)		
Bridge not produced	2 (1.4%)	0 (0.0%)
Bridge produced, gold not fetched	128 (86.5%)	41 (75.9%)
Retrieval subtotal	**130 (87.8%)**	**41 (75.9%)**
Answer-generation (QA) failure (gold fetched, answer wrong)		
Reasoning error	6 (4.1%)	2 (3.7%)
Hallucination	2 (1.4%)	2 (3.7%)
Refusal	4 (2.7%)	5 (9.3%)
Format only (EM=0)	6 (4.1%)	4 (7.4%)
QA subtotal	**18 (12.2%)**	**13 (24.1%)**

Table 14: Recovery from a failed first round. We take the queries where the single-pass round-0 retrieval returns none of the gold passages, and measure how often each method recovers them in later rounds. Both methods use the same graph and PPR, so the only difference is how they form their next-round seeds. BRGD recovers from these dead starts more often than NER+ITER-RETGEN.

Method	Recovered (R@5>0)	Fully (R@5=1)	mean R@5
MuSiQue (n=185)			
NER+ITER-RETGEN	102 (55%)	15 (8%)	0.279
BRGD	**127 (69%)**	**23 (12%)**	**0.381**
2WikiMultiHopQA (n=27)			
NER+ITER-RETGEN	12 (44%)	4 (15%)	0.306
BRGD	**20 (74%)**	**11 (41%)**	**0.574**

Table 15: Cross-round repetition of the reasoning generations (mean over consecutive rounds). Jaccard_1 and Jaccard_2 are word- and bigram-overlap. On both datasets, ITER-RETGEN reuses much of its previous text each round; BRGD's reasoning changes as it explores and names new bridges.

Method	Jaccard_1	Jaccard_2	ROUGE-L
MuSiQue			
NER+ITER-RETGEN	0.50	0.33	0.50
BRGD	**0.36**	**0.14**	**0.30**
2WikiMultiHopQA			
NER+ITER-RETGEN	0.62	0.48	0.64
BRGD	**0.38**	**0.16**	**0.36**

early once the discovered bridge set stops changing: it averages 2.57 rounds on MuSiQue and 2.05 on 2WikiMultiHopQA, and halts before the third round on 27% and 63% of queries. It then issues one dedicated reader call. BRGD therefore does fewer retrieval rounds but adds a final reader call, so its total number of LLM calls is comparable to NER+ITER-RETGEN rather than higher (3.57 vs. 3.00 on MuSiQue, 3.05 vs. 3.00 on 2WikiMultiHopQA). The early-stopping saving is largest on 2WikiMultiHopQA, whose shorter reasoning chains are often resolved within one or two rounds. Beyond call count, BRGD is also more token-efficient. On MuSiQue it uses an estimated $\sim$3.2K input and $\sim$0.4K output tokens per query ($\sim$3.6K in total), versus $\sim$4.1K input and $\sim$0.6K output ($\sim$4.7K total) for NER+ITER-RETGEN. On 2WikiMultiHopQA the same pattern holds: BRGD's earlier stopping (2.05 rounds on average, versus a fixed three) means fewer generation calls and a lower overall token cost.

Table 16: Mean per-query LLM calls for the two iterative methods. NER+ITER-RETGEN runs a fixed three rounds and reads the answer from its last generation, so its calls equal its rounds. BRGD stops early once its discovered bridges stabilise (2.57 rounds on MuSiQue, 2.05 on 2WikiMultiHopQA), then issues one dedicated reader call, so its calls equal its rounds plus one.

Method	MuSiQue	2WikiMultiHopQA
NER+ITER-RETGEN	3.00	3.00
BRGD	3.57	3.05

5.5 Cross-query Bridge Memory

The previous sections show that bridge entities improve retrieval within a query. A natural question is whether they are merely query-specific guesses, or stable retrieval knowledge that can be reused: if the right bridges are known, must they be re-discovered for every query? We study two facets on MuSiQue: reusing a query's own bridges in a single pass, and borrowing bridges from earlier, similar queries.

Single-round bridge value. We first ask how much a query's own bridges are worth without iteration. We take the bridges BRGD discovers for a query, inject them as seeds,

and run one PPR (Table 17). This single-round injection raises EM from 0.261 to 0.346: bridges help even in a single round. It still falls short of BRGD's multi-round reasoning (0.396). Re-seeding alone recovers part of the gain; the rest comes from iterative reasoning.

Table 17: Value of a query's own bridges in a single retrieval pass (MuSiQue, n=1,000). Injecting the discovered bridges as seeds and running one PPR recovers part of the gain over the single-pass NER baseline; BRGD's multi-round reasoning recovers more.

Setting	EM
Single-pass NER (no bridges)	0.261
Own bridges (single round)	0.346
BRGD (multi-round)	**0.396**

Cross-query bridge transfer. We then ask whether bridges generalise across queries. For a new query we find its most similar past query; when their cosine similarity exceeds 0.85 we borrow that query's bridges and let the LLM reflect on and edit the borrowed set (Table 18). On the 18% of queries that have such a neighbour (n=182), this barely improves on the single-pass NER baseline and stays well below BRGD. Bridges are query-conditioned: they do not transfer, and an LLM cannot recover their value for a different query. We observe that LLMs could confidently delete right bridges and/or add wrong ones. This reinforces the per-query reasoning at the core of BRGD.

Table 18: Borrowing bridges from the most similar past query (cosine >0.85) with LLM refinement (refine), on the 18% of MuSiQue queries that have such a neighbour (n=182), against the single-pass NER baseline and BRGD on the same queries.

Setting (n=182)	EM
Single-pass NER (no bridges)	0.302
refine (borrowed + LLM edit)	0.330
BRGD (multi-round)	**0.407**

5.6 Case Study

Figure 5 traces one representative question through both methods. They use the same LLM. The question asks which country the director of *Get Carter* (2000) is from. The chain has two hops: the film leads to its director, Stephen Kay, whose biography records that he was born in New Zealand.

In the first round, both methods retrieve the Wikipedia passage about the film *Get Carter* (2000). The passage describes the film as "an American action thriller film". It names Stephen Kay as the director, but it does not say where he is from. NER+ITER-RETGEN answers from this passage alone. It assumes that an American film implies an American director, and replies "United States". The assumption is wrong, but the loop cannot correct it. Each round appends the previous generation to the query, yet retrieval keeps returning the same film passage and never reaches Stephen Kay's biography. Anchored to that same

passage, ITER-RETGEN generates "United States" again in rounds 2 and 3. Recall stays at 0.50.

BRGD does not answer from the film page. It observes that the director's origin is still missing. It names *Stephen Kay* as a bridge entity, and seeds the next retrieval with him. His biography is then retrieved. Recall rises to 1.00, and the answer–New Zealand–is read from it.

The two methods run under the same setup–the same LLM, graph, corpus, and top-five passages, with comparable numbers of LLM calls (Section 5.4)–and differ only in how reasoning interacts with retrieval. BRGD uses the inferred bridge entity as an explicit retrieval target, which directs retrieval toward the missing evidence. ITER-RETGEN instead relies on generated text to guide the next round, causing retrieval to remain anchored to the same evidence when the required bridge is absent. This behaviour is consistent with our aggregate results rather than an isolated case: the echo-loop analysis (Section 5.3) shows that ITER-RETGEN's reasoning repeats far more across rounds than BRGD's. Across all 1,000 queries, ITER-RETGEN names the gold bridge more often than BRGD but seeds it on the graph far less (Table 11). The gap is propagation, not discovery.

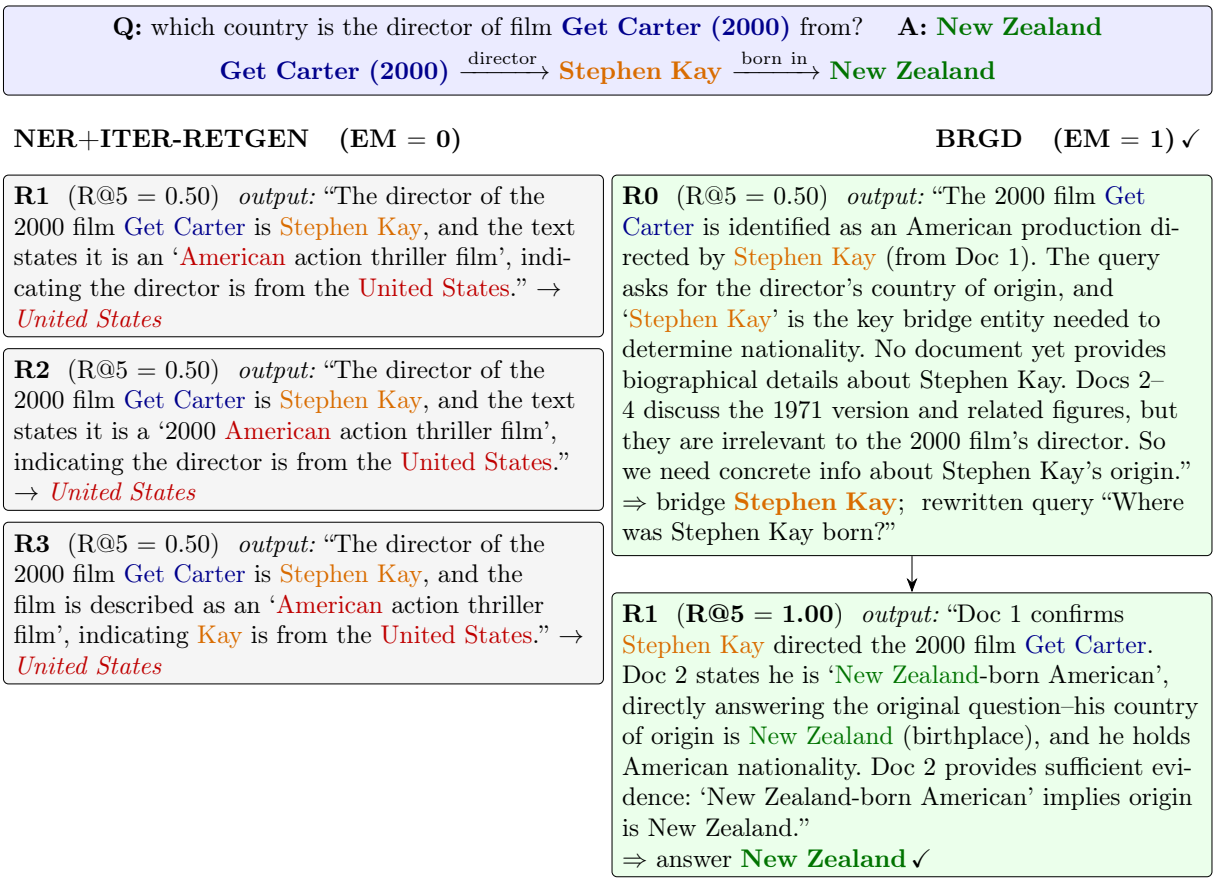


Figure 5: Information flow for a single 2WikiMultiHopQA question under the two iterative retrievers (NER graph, qwen-plus, five passages per round). Both methods retrieve the film's page; only BRGD reaches the director's biography. NER+ITER-RETGEN generates its answer from the retrieved passages and infers the director's nationality from the film's "American" label. BRGD names the director as a bridge entity, seeds the next round's retrieval with him, and reads the answer from his biography. Each box reports that round's recall (R@5) and the column headers report the final exact-match (EM) score. BRGD's per-round recall is logged; ITER-RETGEN's is reconstructed deterministically from its logged round queries.

Chapter 6 Discussion and Conclusion

6.1 Discussion

6.1.1 Bridge Propagation as Soft Retrieval Guidance

BRGD does not change the overall iterative reasoning–retrieval pipeline. Instead, its primary difference from existing iterative methods lies in how intermediate reasoning influences subsequent retrieval. Existing iterative methods typically convert the reasoning outcome into a concrete retrieval action, such as a rewritten query or the next graph expansion. As a result, each reasoning step makes a relatively strong commitment to the subsequent retrieval direction. Such early commitment makes these methods more susceptible to error accumulation across retrieval iterations. Once the retrieval process deviates from the correct trajectory, subsequent retrieval can become increasingly influenced by earlier retrieval decisions, making recovery more difficult.

Rather than directly committing intermediate reasoning to the next retrieval action, BRGD identifies a set of bridge entities from the current reasoning state and retrieved evidence, and injects them as seeds for subsequent graph propagation. Retrieval is still performed by the underlying graph propagation algorithm, while bridge entities guide, rather than determine, the retrieval process. In this way, intermediate reasoning is translated into a structured retrieval signal instead of a single retrieval action.

A possible explanation is that this design reduces the early commitment to a single retrieval direction, allowing subsequent retrieval to incorporate new reasoning signals throughout the iterative process rather than relying primarily on previous retrieval decisions. Rather than requiring each reasoning step to determine the next retrieval action, bridge entities provide soft retrieval guidance that can continue influencing subsequent graph propagation across retrieval iterations. Although this mechanism is not directly validated, it is consistent with our empirical observations.

Recovery and echo-loop analyses support this interpretation by showing that explicit bridge propagation makes the retrieval process easier to recover from failed states while reducing repetitive reasoning across retrieval iterations. Compared to ITER-RETGEN, BRGD substantially increases the recovery rate of failed retrieval trajectories, improving the proportion of recovered cases from 55% to 69% on MuSiQue and from 44% to 74% on 2WikiMultiHopQA. The proportion of fully recovered cases also increases from 8% to 12% on MuSiQue and from 15% to 41% on 2WikiMultiHopQA. Together, these findings suggest that the benefit of bridge propagation lies not only in improving retrieval accuracy, but also in providing a more explicit and structured interface through which intermediate reasoning can influence subsequent graph retrieval.

6.1.2 Bridge Propagation Beyond Bridge Discovery

The failure attribution analysis separates two possible bottlenecks, bridge discovery and bridge propagation. In most analyzed failure cases, the bridge entity has already been identified, so the key challenge lies in propagation rather than discovery.

This observation provides a new perspective for future GraphRAG systems. Existing work has mainly focused on improving reasoning quality or knowledge graph quality,

whereas our results suggest that how intermediate reasoning is propagated into downstream retrieval deserves equal attention. The degree-adaptive weighting used in BRGD can be viewed as a simple optimization of the graph propagation process. By adjusting the influence of the bridge entities according to the local graph topology, those entities participate more effectively in subsequent retrieval. Our ablation confirms that even this simple, purely local reweighting yields a consistent gain (Section 5.2). However, this design still relies solely on local structural information, leaving substantial room for improving bridge propagation. Future work may incorporate semantic information, relation-aware weighting, or learned propagation policies to further strengthen this reasoning-to-retrieval coupling.

6.1.3 Long-chain Retrieval Challenge in Bridge Propagation

Experiments on graph representations and propagation operators jointly suggest that the effectiveness of BRGD is not tied to a particular graph construction or diffusion algorithm. BRGD consistently improves performance on both the semantically sparse NER graph and the semantically rich OpenIE graph. Moreover, replacing Personalized PageRank with Katz diffusion results in virtually identical performance. These observations indicate that the main source of improvement comes from the BRGD framework rather than from a specific graph representation or propagation operator.

Hop-wise analysis further reveals where bridge propagation is most beneficial. On MuSiQue, both EM and R@5 decrease steadily as the retrieval chain grows from two to four hops, confirming that long dependent retrieval chains remain a major challenge for multi-hop retrieval. Nevertheless, BRGD consistently achieves the highest retrieval recall in all reasoning depths, with its largest recall advantage over ITER-RETGEN appearing in the 4-hop subset. This suggests that bridge propagation becomes increasingly valuable as reasoning chains become longer. Notably, on the MuSiQue 4-hop subset, both the semantically sparse NER graph and the semantically rich OpenIE graph encounter a similar retrieval bottleneck. Under single-pass retrieval, the two graph constructions achieve R@5 scores of only 0.239 and 0.238, respectively, while BRGD improves them to 0.389 and 0.380. Here the advantage of the richer graph nearly disappears, which suggests that once a graph already covers the needed knowledge, improving its representation does little for long-chain retrieval. The main challenge shifts instead to keeping the intermediate reasoning accurate across rounds and limiting error accumulation.

Future work can target this reasoning step directly. Stronger reasoning models may improve the intermediate steps and reduce error propagation across rounds. Adaptive retrieval strategies that dynamically adjust the reasoning depth based on question complexity may further improve computational efficiency. In addition, trajectory-aware optimization that prunes, redirects, or terminates unpromising bridge-guided retrieval trajectories may further reduce error accumulation while improving both retrieval efficiency and robustness.

6.2 Conclusion

As the reasoning capabilities of large language models continue to improve, recent research has gradually shifted from simply enhancing model capability to improving interactions

between models, external knowledge, and external tools. In this work, we propose BRGD (Bridge Reasoning Guided Diffusion), an iterative GraphRAG framework for multi-hop retrieval. Unlike existing iterative methods that directly convert intermediate reasoning into retrieval actions, BRGD identifies bridge entities from intermediate reasoning and uses them as seeds for graph propagation. This design follows a basic principle in information retrieval, that semantic similarity does not always imply retrieval relevance, since the evidence for a multi-hop question is often connected through bridge entities rather than reachable by similarity alone. By propagating bridge entities with optimized propagation weights, BRGD continuously transfers structured reasoning signals throughout the retrieval process and enables tighter reasoning–retrieval interaction.

On MuSiQue and 2WikiMultiHopQA, BRGD delivers consistent gains in both retrieval and answer accuracy, and our analyses trace where these gains come from. Ablations attribute them mainly to bridge-guided propagation rather than to query rewriting or rank fusion. The gains persist across graph constructions and diffusion operators, indicating that the benefit is not tied to a specific graph or algorithm. Recovery, echo-loop, and failure attribution analyses further show a more stable iteration with reduced error accumulation.

More broadly, this work suggests that iterative GraphRAG depends not only on the graph or the retrieval algorithm, but also on how reasoning and retrieval interact. We hope it draws attention to that question, not only what to retrieve but also how reasoning shapes retrieval, and offers a simple framework for exploring it.

Appendix A Prompts

The pipeline uses five fixed one-shot prompts, all run at temperature 0. We reproduce them verbatim below. Braces such as `{passage}` or `{question}` mark a runtime-filled slot, and `[ . . . ]` marks an in-context example passage abridged for space.

A.1 Passage named-entity extraction (NER graph)

```
[system]
Your task is to extract named entities from the given paragraph. Respond with a
    JSON list of entities.

[one-shot user]
Radio City
Radio City is India's first private FM radio station and was started on 3 July
    2001. It plays Hindi, English and regional songs. Radio City recently forayed
    into New Media in May 2008 with the launch of a music portal -
    PlanetRadiocity.com that offers music related news, videos, songs, and other
    music-related features.

[one-shot assistant]
{"named_entities":
    ["Radio City", "India", "3 July 2001", "Hindi", "English", "May 2008",
    "PlanetRadiocity.com"]
}

[user]
{passage}
```

A.2 OpenIE triple extraction (OpenIE graph)

```
[system]
Your task is to construct an RDF (Resource Description Framework) graph from the
    given passages and named entity lists. Respond with a JSON list of triples,
    with each triple representing a relationship in the RDF graph.

Pay attention to the following requirements:
- Each triple should contain at least one, but preferably two, of the named
    entities in the list for each passage.
- Clearly resolve pronouns to their specific names to maintain clarity.

[one-shot user]
Convert the paragraph into a JSON dict, it has a named entity list and a triple
    list.
Paragraph: Radio City [... same passage as above ...]
{"named_entities": ["Radio City", "India", "3 July 2001", "Hindi", "English", "May
    2008", "PlanetRadiocity.com"]}

[one-shot assistant]
{"triples": [
    ["Radio City", "located in", "India"],
    ["Radio City", "is", "private FM radio station"],
    ["Radio City", "started on", "3 July 2001"],
    ["Radio City", "plays songs in", "Hindi"],
    ["Radio City", "plays songs in", "English"],
```

```
    ["Radio City", "forayed into", "New Media"],
    ["Radio City", "launched", "PlanetRadiocity.com"],
    ["PlanetRadiocity.com", "launched in", "May 2008"],
    ["PlanetRadiocity.com", "is", "music portal"],
    ["PlanetRadiocity.com", "offers", "news"],
    ["PlanetRadiocity.com", "offers", "videos"],
    ["PlanetRadiocity.com", "offers", "songs"]
]}

[user]
Convert the paragraph into a JSON dict, it has a named entity list and a triple
    list.
Paragraph:
{passage}
{named_entity_json}
```

A.3 Bridge prediction

```
[system]
You are a retrieval reasoning assistant inspired by how human memory uses both
    observed cues and hypothetical context. Given an original query, the documents
    retrieved so far, and optionally a reasoning trace, your job is to:

1. Analyze what information has been found and what is still missing.
2. Identify CONCRETE bridge entities you SEE in retrieved documents (these will
    guide structured retrieval).
3. Hypothesize a CONTEXT SCENARIO - natural language describing the likely time
    period, location, domain, or related concepts (this is your reasoning context,
    NOT the retrieval query).
4. Rewrite the query using ONLY bridge entities and the original question - keep it
    focused and short. Do NOT include the context_scenario in the rewritten_query.
5. Decide whether to continue retrieval or stop.

Respond in JSON format:
{
    "analysis": "Brief analysis of what's found vs missing",
    "discovered_entities": ["entity1", "entity2"],
    "context_scenario": "Hypothesized context: time period, location, domain,
    related concepts",
    "rewritten_query": "Focused natural language query with bridge entities only",
    "should_stop": false
}

Rules:
- "discovered_entities": ONLY entities you can directly SEE in retrieved documents.
    1-5 entities max. Use lowercase. DO NOT guess.
- "context_scenario": Short natural language description of the hypothesized
    context. Stored as your reasoning trace and seen by future rounds, but NOT used
    directly for retrieval. Examples:
  * "Likely 16th century European religious reform, possibly Lutheran/Calvinist
    figures"
  * "North Carolina state administration history, 18-19th century capital
    designation"
  * "Modern NBA basketball event, post-2000 era"
- "rewritten_query": Focused query using ONLY bridge entities + reformulation of
    the original question. DO NOT add context_scenario, time periods, or
```

```
    hypothesized terms here. Keep it short and concrete.
- "should_stop": true ONLY if retrieved documents already contain a complete answer.

Example:
Q: "When did the city where Curry's college is located become NC's capital?"
After round 0 finding Stephen Curry and Davidson College:
{
    "analysis": "Found Curry attended Davidson College in NC. Need the city of
    Davidson and its history as state capital.",
    "discovered_entities": ["davidson college", "north carolina"],
    "context_scenario": "Likely 18-19th century North Carolina state administration
    history; possibly involving Raleigh or another NC city becoming the state
    capital.",
    "rewritten_query": "When did Davidson, North Carolina become the state
    capital?",
    "should_stop": false
}
```

A.4 Reader (question answering)

```
[system]
As an advanced reading comprehension assistant, your task is to analyze text
    passages and corresponding questions meticulously. Your response start after
    "Thought: ", where you will methodically break down the reasoning process,
    illustrating how you arrive at conclusions. Conclude with "Answer: " to present
    a concise, definitive response, devoid of additional elaborations.

[one-shot user]
Wikipedia Title: Neville A. Stanton
Neville A. Stanton is a British Professor of Human Factors and Ergonomics at the
    University of Southampton. [...]

Wikipedia Title: Southampton
The University of Southampton, which was founded in 1862 and received its Royal
    Charter as a university in 1952, has over 22,000 students. [...]
[+ three more Wikipedia passages]

Question: When was Neville A. Stanton's employer founded?

Thought:

[one-shot assistant]
The employer of Neville A. Stanton is University of Southampton. The University of
    Southampton was founded in 1862.
Answer: 1862.

[user]
Wikipedia Title: {passage}
...
Question: {query}
Thought:
```

References

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 335–336, 1998.

Jifan Chen and Greg Durrett. Understanding dataset design choices for multi-hop reasoning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 4026–4032, 2019.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 758–759, 2009.

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

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

Kai Guo, Xinnan Dai, Shenglai Zeng, Harry Shomer, Haoyu Han, Yu Wang, and Jiliang Tang. Beyond static retrieval: Opportunities and pitfalls of iterative retrieval in graphrag. *arXiv preprint arXiv:2509.25530*, 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

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

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudio d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4):1–37, 2021.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research (TMLR)*, 2022.

Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th International Conference on World Wide Web (WWW)*, pages 271–279, 2003.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 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, 2020.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1): 39–43, 1953.

Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. LLMs get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

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. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 9459–9474, 2020.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, Wenbo Su, and Bo Zheng. 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, 2024.

Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. HopRAG: Multi-hop reasoning for logic-aware retrieval-augmented generation. *arXiv preprint arXiv:2502.12442*, 2025.

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

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4089–4100, 2021.

Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 404–411, 2004.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional questions do not necessitate multi-hop reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4249–4257, 2019.

Morris Moscovitch. Memory and working-with-memory: A component process model based on modules and central systems. *Journal of Cognitive Neuroscience*, 4(3):257–267, 1992.

Morris Moscovitch, Roberto Cabeza, Gordon Winocur, and Lynn Nadel. Episodic memory and beyond: The hippocampus and neocortex in transformation. *Annual Review of Psychology*, 67:105–134, 2016.

Md Mahadi Hasan Nahid and Davood Rafiei. Prism: Agentic retrieval with llms for multi-hop question answering. *arXiv preprint arXiv:2510.14278*, 2025.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, 1999.

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

Zackary Rackauckas. RAG-Fusion: A new take on retrieval-augmented generation. *arXiv preprint arXiv:2402.03367*, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992, 2019.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

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

Daniel L. Schacter and Donna Rose Addis. The cognitive neuroscience of constructive memory: Remembering the past and imagining the future. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1481):773–786, 2007.

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

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step ques-

tions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 10014–10037, 2023.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9414–9423, 2023.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2369–2380, 2018.