BINDRAG: BINDING-AWARE RETRIEVAL FOR MULTI-HOP QUESTION ANSWERING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) often struggle with knowledge-intensive questions that require retrieving and combining evidence across multiple reasoning steps, often from different documents. Retrieval-augmented generation (RAG) mitigates this limitation by providing external evidence, but conventional retrieval is poorly suited to multi-hop questions where evidence needed at a later step depends on an intermediate answer discovered earlier. Decomposition and iterative retrieval methods address this by generating follow-up queries step by step, but they typically leave the dependency on intermediate answers implicit or commit to a single intermediate answer, allowing early errors to propagate to later retrieval. We present **BindRAG**, a multi-hop retrieval framework that explicitly binds intermediate answers to downstream retrieval queries, making sequential dependencies directly executable. BindRAG further maintains a query-aware memory that preserves alternative bindings and hop-structured evidence while retrieval proceeds. The highest-ranked binding advances the primary retrieval path, while alternative candidates and their supporting evidence are retained for final resolution. This allows uncertain intermediate decisions to be reconsidered without repeatedly changing the retrieval trajectory. Across HotpotQA, MuSiQue, and 2WikiMultiHopQA, BindRAG improves answer accuracy by an average of 10.3 percentage points over the strongest baseline.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong capabilities in knowledge-intensive question answering, yet their reliance on parametric knowledge can lead to unsupported or outdated responses (Achiam et al., 2023; Anthropic, 2024; Zhou et al., 2025a). Retrieval-augmented generation (RAG) mitigates this limitation by retrieving relevant external evidence to inform generation (Zhang et al., 2025; Gao et al., 2023; Lewis et al., 2020; Izacard et al., 2023). However, conventional RAG is poorly matched to multi-hop questions, where evidence is distributed across documents and retrieving later evidence may depend on an intermediate entity discovered at an earlier step (Han et al., 2024; Xiong et al., 2020; Shao et al., 2023). Recent work has introduced more structured approaches to multi-hop retrieval. Graph-based RAG organizes textual knowledge into entity or passage graphs, hierarchical representations, or graph-based relevance propagation, enabling retrieval to exploit relationships across distributed pieces of evidence rather than relying solely on direct query–passage similarity (Edge et al., 2024; Sarthi et al., 2024; Wang et al., 2024; Guo et al., 2025; Gutiérrez et al., 2024; Wang et al., 2026). Other approaches interleave retrieval with reasoning or explicitly model dependencies among subproblems, allowing retrieval to progressively incorporate intermediate reasoning states or follow structured dependencies across multiple steps (Trivedi et al., 2023; Chen et al., 2026; Fu et al., 2023; Yang et al., 2024; Jiang et al., 2023; Su et al., 2024).

Despite this progress, two challenges remain in handling dependencies across retrieval steps. First, multi-hop question answering often requires identifying a specific intermediate entity before formulating and executing a subsequent retrieval query. In such cases, the downstream query can only be fully instantiated after the required entity is discovered from an earlier step. Figure 1 illustrates this challenge with an empirical case where BindRAG succeeds while all evaluated baselines fail (see Appendix C for a detailed analysis). For the question "Who became a star as a comic book character created by Gerry Conway and Bob Oksner?", the first step identifies *Vixen* as the character created

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

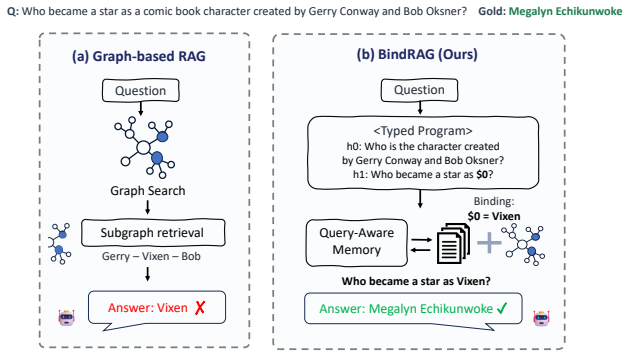

Figure 1: **Motivation for binding-aware retrieval.** In this example, graph-based RAG recovers the correct intermediate entity (*Vixen*) but returns it prematurely as the final answer. BindRAG instead explicitly binds the intermediate entity to the unresolved slot (`$0`), instantiates the downstream query (*Who became a star as Vixen?*), and continues retrieval to obtain the requested answer (*Megalyn Echikunwoke*).

by Gerry Conway and Bob Oksner. Notably, several baselines also recover *Vixen*, but incorrectly return it as the final answer. However, *Vixen* is only an intermediate entity, not the requested answer. BindRAG instead assigns it to the unresolved slot $0, instantiating the downstream query "Who became a star as $0?" as "Who became a star as Vixen?", which enables retrieval of the correct final answer, *Megalyn Echikunwoke*. We refer to this operation as *binding*: assigning an intermediate answer to an unresolved slot in a downstream query. Graph-based retrieval can traverse structural connections among entities or passages, but traversal alone does not explicitly resolve an intermediate answer and substitute it into a dependent query. Thus, while modeling dependencies among subproblems determines the order of retrieval, effective multi-hop retrieval also requires resolving and propagating the specific entity needed to instantiate each dependent query. Second, intermediate bindings are inherently uncertain, and an early mistake can propagate through subsequent retrieval steps. Retrieved evidence may support multiple plausible intermediate entities or values, making a single binding difficult to determine with certainty. Although a concrete binding is needed to continue retrieval, committing to one execution path should therefore not discard alternative candidates or their supporting evidence. BindRAG therefore advances retrieval with the top-ranked binding while preserving uncertainty and hop-level evidence for joint resolution at the final stage.

To address these challenges, we introduce **BindRAG**, a multi-hop retrieval framework built around two key components: *binding-aware execution* and *query-aware memory*. Binding-aware execution explicitly resolves intermediate answers and propagates them to dependent queries, making sequential retrieval dependencies directly executable. Query-aware memory preserves alternative bindings and hop-level evidence throughout retrieval, allowing uncertainty in intermediate decisions to remain available for grounded final resolution.

Our main contributions are:

- We formulate **binding-aware execution** for multi-hop retrieval, which makes latent dependencies between subproblems executable by representing intermediate answers as explicit bindings for downstream retrieval.
- We develop **uncertainty-aware execution** with query-aware memory, enabling BindRAG to preserve ambiguous intermediate states and retrieved evidence for final resolution without interrupting the primary retrieval trajectory.
- Extensive experiments on HotpotQA, MuSiQue, and 2WikiMultiHopQA demonstrate consistent improvements over strong dense and graph-based RAG baselines. Ablation and reasoning-structure analyses further show that the gains are primarily associated with binding propagation and query-aware memory, particularly for questions requiring sequential retrieval dependencies.

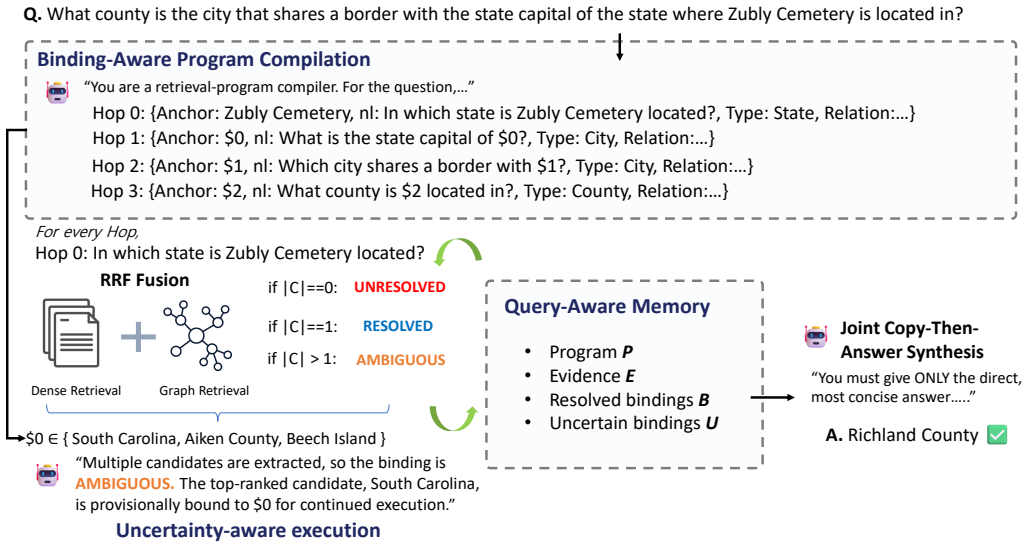

Figure 2: Framework of BindRAG. BindRAG compiles a multi-hop question into a binding-aware program and explicitly tracks ambiguous intermediate bindings during retrieval and execution.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) grounds language models in external evidence retrieved from a corpus, typically through sparse or dense passage retrieval (Lewis et al., 2020; Cuconasu et al., 2024; Sawarkar et al., 2024). Conventional RAG, however, retrieves evidence largely with respect to the original query, which is problematic for multi-hop questions where later retrieval depends on an intermediate entity discovered earlier. Decomposition and iterative methods address this by generating follow-up questions (Press et al., 2023; Khot et al., 2023) or interleaving retrieval with reasoning (Trivedi et al., 2023; Shao et al., 2023; Jiang et al., 2023; Su et al., 2024), and plan-based agents reference earlier outputs through placeholder variables (Xu et al., 2023; Kim et al., 2024).

2.2 Graph-Based Retrieval-Augmented Generation

Another line of work extends RAG with structured representations of entities and passages (Xiang et al., 2026; Peng et al., 2025), including hierarchical summarization (RAPTOR; Sarthi et al., 2024), entity-centric community summaries (GraphRAG; Edge et al., 2024), subgraph retrieval (G-Retriever; He et al., 2024), dynamic graph construction (DALK; Li et al., 2024), graph-guided passage traversal (KGP; Wang et al., 2024), relevance propagation (HippoRAG/HippoRAG2; Gutiérrez et al., 2024; 2025), hybrid graph–vector indexing (LightRAG; Guo et al., 2025), and adaptive reasoning structures without pre-built graphs (LogicRAG; Chen et al., 2026).

Graph traversal connects distributed evidence but does not itself resolve an intermediate answer and substitute it into a dependent query. Iterative methods regenerate the next query from free-form reasoning, leaving this dependency implicit, while plan-based agents make it explicit but commit to a single value per step. BindRAG deterministically substitutes typed bindings into downstream retrieval queries, and preserves alternative bindings with hop-structured evidence for resolution at the final synthesis step.

3 METHOD

3.1 PROBLEM FORMULATION

We consider open-domain multi-hop question answering. Given a question q and a corpus $\mathcal{C}$ of passages, the goal is to return a short answer y. The central difficulty is not merely that answering q requires multiple facts, but that these facts are often *not jointly addressable*: a passage needed at a later reasoning step may be indexed under an entity that is never mentioned in the original question. As illustrated in Figure 2, BindRAG addresses this challenge through two collaborative components: *Binding-aware Execution* and *Query-Aware Memory*. Binding-aware Execution converts latent dependencies among subproblems into executable retrieval operations by propagating intermediate answers into downstream queries, while Query-Aware Memory preserves intermediate binding states and hop-structured evidence throughout execution for final resolution. We first introduce the overall framework, followed by the binding-based execution procedure, query-aware memory, and evidence-grounded final resolution.

Why decomposition alone is insufficient. A natural approach is to decompose q into a sequence of sub-questions and retrieve evidence for each independently. However, decomposition alone does not resolve the addressability problem. Consider the question *"What year did the publisher of Labyrinth end?"*. After identifying the publisher, the next query must search for that publisher itself. A sub-question such as *"When did the publisher end?"* remains poorly addressable because the relevant passage is indexed under *Acornsoft*, a string absent from the original question.

We therefore make intermediate answers explicit and substitute them into downstream queries. This operation, which we call *binding*, turns a latent reasoning dependency into a literal retrieval dependency, as illustrated in Figure 2.

3.2 BINDING-AWARE PROGRAM COMPILATION

BindRAG maintains a question-local memory (QUERY-AWARE MEMORY)

$$M_q = (P, E, B, U), \tag{1}$$

which exists only during the execution of question q. Here, P is the typed reasoning program, E is the accumulated hop-structured retrieval evidence, B contains committed intermediate bindings, and U contains uncertain candidate bindings. The memory is not a persistent knowledge store; rather, it records the evolving execution state needed to instantiate subsequent retrieval queries and to expose the complete reasoning trace to final answer generation.

A typed program is an ordered sequence

$$P = (o_1, \ldots, o_H), \qquad o_h = (\kappa_h, a_h, r_h, \tau_h, \mathrm{nl}_h), \tag{2}$$

where κ_h is the operation kind, a_h is its anchor, r_h is the requested relation, τ_h is the expected answer type, and nl_h is a natural-language rendering of the operation. Anchors may contain slots: $a_h = \$i$ denotes the output of operation o_i. Once a binding b_i is obtained, every downstream occurrence of $\$i$ is replaced by b_i before retrieval.

BindRAG executes this program sequentially. At each step it (1) resolves available bindings, (2) retrieves evidence for the resulting query, (3) extracts a ranked set of candidate bindings, and (4) updates M_q with the newly retrieved evidence and binding state. After all retrieval steps, a single synthesis call jointly reads the accumulated evidence and commits to the final answer.

Explicit dependency resolution. A single LLM call compiles the original question into a typed dependency program P. Rather than generating or rewriting the next query after each retrieval step, BindRAG predicts all operations jointly in one pass. Each operation specifies an anchor, a natural-language relation, an expected answer type, and a sub-question, while dependencies between operations are represented through explicit slots (e.g., \$0). Consequently, program construction requires a single LLM call per question, and downstream queries can be instantiated directly from intermediate bindings during execution.

For example, consider the question *"What county was Erik Hort born in?"*. The decomposer produces the following two-step program:

$$\begin{aligned} o_1:\ &\textsc{Hop}(\texttt{anchor} = \text{Erik Hort},\ \texttt{relation} = \text{birthplace},\ \texttt{type} = \text{City}) \\ &\quad \rightarrow \text{"Where was Erik Hort born?"}, \\ o_2:\ &\textsc{Hop}(\texttt{anchor} = \$0,\ \texttt{relation} = \text{county},\ \texttt{type} = \text{County}) \\ &\quad \rightarrow \text{"What county is \$0 a part of?"}. \end{aligned} \tag{3}$$

Once o_1 returns a binding, e.g., *Montebello, New York*, the slot \$0 in o_2 is replaced deterministically, yielding *"What county is Montebello, New York a part of?"*. Binding substitution is therefore mechanical rather than delegated to an additional reasoning call. Unresolved slots are never sent directly to the retriever; when no usable binding is extracted, BindRAG invokes the recovery procedure in Sec. 3.3. The expected answer type serves as soft guidance for extraction rather than a hard constraint, avoiding rejection of evidence-supported candidates whose surface form differs from the predicted type. Relations are represented as natural-language phrases rather than fixed schema identifiers and are retained with intermediate bindings in the query-aware memory.

BindRAG executes retrieval sequentially because the query at step h may depend on bindings produced at earlier steps. Let

$$\tilde{q}_h = \mathrm{Resolve}(\mathrm{nl}_h, B) \tag{4}$$

denote the query obtained after substituting all currently resolved slots.

For each $\tilde{q}_h$, we retrieve $k = 5$ passages. The resulting evidence is read once to produce a ranked list of at most three candidate bindings

$$C_h = (c_{h,1}, c_{h,2}, c_{h,3}), \tag{5}$$

where available, and the highest-ranked candidate is used as the committed binding, $b_h = c_{h,1}$.

The crucial invariant is that the top-ranked candidate is never replaced by a later post-hoc correction mechanism. Instead, candidate multiplicity controls only how broadly the next retrieval step searches and what uncertainty is exposed to the final reader.

Dense and graph retrieval. To capture complementary semantic and structural information, BindRAG combines dense and graph-based retrieval. For a resolved query $\tilde{q}_h$, the dense retriever obtains a ranking $D(\tilde{q}_h)$ from a passage embedding index, capturing semantic relevance between the query and passages. In parallel, the graph retriever captures structural and relational relevance through personalized PageRank over an OpenIE graph constructed from the corpus, following the knowledge graph construction procedure of HippoRAG2 (Gutiérrez et al., 2025), and produces a ranking $G(\tilde{q}_h)$. We combine the two rankings using reciprocal rank fusion:

$$s(p) = \sum_{L \in \{D,G\}} \frac{1}{\lambda + \mathrm{rank}_L(p)}, \qquad E_h = \text{top-}k(s), \tag{6}$$

where $\lambda = 60$ and $k = 5$.

The graph retriever serves as an auxiliary source of structural evidence rather than the central mechanism of BindRAG. Removing it has inconsistent effects across datasets (Table 2); the primary contribution of BindRAG is instead to make sequential retrieval explicitly dependent on intermediate bindings.

3.3 Uncertainty-aware execution

Candidate multiplicity provides a simple, call-free uncertainty signal. After extracting C_h, we distinguish three execution states:

$$\ell_h = \begin{cases} \textsc{Unresolved}, & |C_h| = 0, \\ \textsc{Resolved}, & |C_h| = 1, \\ \textsc{Ambiguous}, & |C_h| \geq 2. \end{cases} \tag{7}$$

The label itself requires no additional LLM call. Importantly, these states do not trigger post-hoc replacement of b_h. Our experiments show that using uncertainty to re-rank or replace intermediate

bindings does not improve answer accuracy. We therefore use uncertainty only to control evidence acquisition and memory exposure.

When $|C_h| = 1$, the candidate is committed to B and substituted directly into all downstream slots. When multiple plausible candidates are extracted, the top-ranked candidate remains the committed binding, but BindRAG additionally retrieves evidence conditioned on the alternative candidates. To control context growth, the union of these branch-specific results is reduced to five passages, prioritizing passages associated with the committed anchor. The complete candidate set is stored in U and later exposed to the final synthesis stage as an uncertain intermediate binding. Thus ambiguity changes the *retrieval scope* and the information exposed to the final reader, but never retrospectively changes the committed top-1 binding.

When no candidate is extracted, directly continuing would leave a downstream slot unresolved. BindRAG instead rewrites the current sub-question into a retrieval query that preserves the anchor entity and requested relation while adding only alternative expressions of the relation that may appear in relevant passages. The rewritten query is accepted only if the original anchor is preserved; we then retrieve again with the same $k = 5$ budget and repeat extraction, for at most two rewrite attempts. If recovery still fails, a forced ranked extraction supplies a binding so that subsequent slots remain executable. This recovery path is intentionally narrow: it repairs retrieval-query mismatch, but it cannot repair an incorrect binding produced by an earlier step.

3.4 Query-Aware Memory and Final Synthesis

After sequential execution, BindRAG retains the original passages rather than compressing them into intermediate summaries. Let

$$E = \mathrm{Dedup}\left(\bigcup_{h=1}^{H} E_h\right) \tag{8}$$

denote the accumulated evidence. After each hop, E_h is incorporated into the evidence component of M_q; after execution, E denotes the deduplicated union of all hop-level evidence. We perform order-preserving exact deduplication but no relevance-based pruning.

Hop-structured evidence. Rather than presenting E as a flat list, passages are grouped by the retrieval step in which they first appeared. The final reader therefore receives an execution trace of the form

Step 1: "Where was Erik Hort born?"
$\Rightarrow$ Montebello, New York
[passages first retrieved at Step 1]

Step 2: "What county is Montebello, New York a part of?"
[passages first retrieved at Step 2] (9)

The reader additionally receives the committed bindings B and, for ambiguous steps, the candidate sets stored in U. Hop grouping changes only evidence organization: no passage is removed by this operation. The final stage receives the original question, the typed program, the accumulated hop-structured evidence, the resolved bindings in B, and uncertain bindings in U. A single LLM call then produces the answer. A direct answer-generation prompt leaves substantial performance on the table even when the required evidence has already been retrieved. We therefore separate *grounding* from *commitment* inside the same generation call.

Copy-then-answer. Before producing its final answer, the reader first copies up to three plausible answer spans verbatim from the accumulated passages and only then commits to a final answer (full prompt in Appendix E.3). Only the `ANSWER` field is returned by the system; the candidate list is an internal grounding scaffold consumed by no subsequent module, so this mechanism adds no additional LLM call. The final answer is not constrained to match one of the copied spans exactly, which preserves cases where the evidence supports the answer but the question requires a different surface form.

Why defer alternatives to the final step? Re-selecting intermediate bindings can alter all downstream queries and propagate an incorrect correction, whereas final-stage selection cannot perturb the retrieval trajectory. We therefore preserve alternative candidates as evidence for final resolution rather than repeatedly revising intermediate bindings; additional analysis is provided in Appendix D.

4 EXPERIMENTS

4.1 DATASETS AND IMPLEMENTATION DETAILS

We evaluate our method on three multi-hop question answering benchmarks: MuSiQue (Trivedi et al., 2022), HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (Ho et al., 2020). Following HippoRAG (Gutiérrez et al., 2024), we randomly sample 1,000 questions from the validation split of each dataset. Detailed dataset statistics and reasoning-type distributions are provided in Appendix A.

To ensure a fair comparison, all methods use the same retrieval corpus constructed from the sampled evaluation questions. We evaluate all the baseline methods using the DIGIMON evaluation framework (Zhou et al., 2025b), ensuring consistent experimental settings across different approaches. We employ `BAAI/bge-m3` as the dense retriever for both our method and all retrieval-based baselines. Unless otherwise specified, the top-k retrieval size is set to 5 for all methods. We use `Qwen2.5-32B-Instruct` as the backbone language model for question decomposition, reasoning, and answer generation across all methods to eliminate the influence of different LLMs. Following prior work, we use `gpt-4o-mini` (Achiam et al., 2023) solely as the LLM evaluator for automatic answer assessment. All experiments are conducted with a decoding temperature of 0.0.

4.2 BASELINES

We compare BindRAG against a diverse set of baselines spanning three categories.

(i) LLM-only inference. We first evaluate the backbone LLM without external retrieval, providing a reference for the contribution of retrieved evidence to multi-hop question answering.

(ii) Vanilla retrieval-augmented generation. We evaluate standard dense RAG under multiple retrieval budgets, where the top-k passages are retrieved directly with respect to the original question and provided to the reader without explicit multi-hop execution. This category serves as a strong non-graph retrieval baseline and isolates the benefit of structured multi-hop retrieval beyond simply increasing the amount of retrieved context.

(iii) Graph-based retrieval-augmented generation. We compare against representative graph-based RAG systems that introduce structured representations or graph-guided retrieval over external knowledge, including KGP (Wang et al., 2024), RAPTOR (Sarthi et al., 2024), G-Retriever (He et al., 2024), Microsoft GraphRAG (Edge et al., 2024), LightRAG (Guo et al., 2025), DALK (Li et al., 2024), HippoRAG2 (Gutiérrez et al., 2025), and LogicRAG (Chen et al., 2026). These methods cover complementary approaches to structured retrieval, including hierarchical abstraction, graph traversal, entity–passage association, and graph-based relevance propagation.

For methods that provide both single-step and multi-step retrieval variants, we use the multi-step configuration when available. Unless otherwise specified, all baselines are evaluated under the same datasets, corpus, backbone language model, and evaluation protocol as BindRAG to enable controlled comparison.

4.3 METRICS

We evaluate all methods using two complementary metrics. **String Accuracy (Str-Acc)** computes whether the gold answer is contained in the generated answer after normalizing both strings to lowercase, removing articles (*a*, *an*, and *the*), removing punctuation, and collapsing consecutive whitespace. Since this metric cannot account for semantically equivalent answers with different surface forms, we additionally report **LLM Accuracy (LLM-Acc)**, which measures whether the generated answer is semantically equivalent to the gold answer using `gpt-4o-mini` as an automatic evaluator.

Table 1: Main results on three multi-hop QA benchmarks. Str-Acc denotes string matching accuracy, and LLM-Acc denotes LLM-based answer accuracy. All results are evaluated on 1,000 questions using the same evaluation protocol. Best results are shown in **bold**, and second-best results are underlined.

Type	Method	HotpotQA		MuSiQue		2WikiMQA	
		Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.
LLM-only inference	Llama-3 (8B)	21.6	24.1	3.9	5.1	23.8	23.3
	Qwen (32B)	25.6	28.2	6.1	8.3	28.1	27.2
	GPT-3.5-turbo	34.1	37.5	8.0	12.6	29.4	29.2
	GPT-4o-mini	32.5	35.5	7.4	11.3	28.8	27.9
Vanilla RAG	Vanilla RAG (Top-1)	45.3	48.5	14.6	17.7	28.5	28.3
	Vanilla RAG (Top-3)	59.8	64.2	22.0	25.5	51.2	49.0
	Vanilla RAG (Top-5)	63.4	68.2	25.3	28.9	53.1	50.3
Graph-based RAG	G-Retriever (GR)	22.7	25.5	3.3	4.3	25.4	24.7
	DALK	34.2	37.6	10.8	13.6	35.3	33.9
	LightRAG	54.5	59.7	23.2	28.1	45.4	44.3
	KGP	54.2	58.2	21.6	25.6	47.1	46.0
	GraphRAG	61.1	66.5	28.1	32.7	49.5	48.2
	RAPTOR	62.7	66.9	24.4	28.1	43.9	42.4
	HippoRAG2	63.8	67.6	28.7	29.0	67.4	59.6
	LogicRAG	60.9	69.5	29.3	36.8	74.6	70.5
Ours	**BindRAG**	**70.3**	**74.7**	**48.3**	**52.6**	**80.1**	**76.0**

Table 2: Ablation study of BindRAG across three multi-hop QA benchmarks. Values report string-matching accuracy, with absolute changes from the full model shown in parentheses.

Ablation	MuSiQue	HotpotQA	2WikiMQA
BindRAG	**48.3**	**70.3**	**80.1**
− Binding Propagation	39.9 (−8.4)	68.7 (−1.6)	72.2 (−7.9)
− Query-Aware Memory (QAM)	42.0 (−6.3)	70.6 (+0.3)	65.1 (−15.0)
− Ambiguity-Aware Candidate Handling	47.3 (−1.0)	67.7 (−2.6)	78.2 (−1.9)
− KG-Augmented Retrieval	47.7 (−0.6)	69.8 (−0.5)	77.1 (−3.0)

4.4 Main Results

Table 1 shows that BindRAG achieves the highest String Accuracy on all three benchmarks, outperforming the strongest graph-based baseline by 6.5, 19.0, and 5.5 points on HotpotQA, MuSiQue, and 2WikiMQA, respectively. The largest gain occurs on MuSiQue, where later retrieval frequently depends on entities discovered at earlier steps, supporting the benefit of explicit binding propagation. The same overall trend holds under LLM-based evaluation, indicating that the gains are not specific to lexical matching.

4.5 Ablation Study

Table 2 shows that binding propagation and query-aware memory are the primary sources of improvement. Removing binding propagation reduces accuracy by 8.4 points on MuSiQue and 7.9 points on 2WikiMQA, compared with only 1.6 points on HotpotQA, supporting its role in questions where later retrieval depends on entities discovered at earlier steps. Removing query-aware memory causes an even larger 15.0-point drop on 2WikiMQA and a 6.3-point drop on MuSiQue, while leaving HotpotQA nearly unchanged. Together, these results indicate that explicit binding and persistent intermediate evidence are most beneficial when retrieval must follow sequential dependencies and integrate information across steps.

Ambiguity handling and graph augmentation provide smaller, more dataset-dependent gains. Removing ambiguity handling reduces accuracy by 1.0–2.6 points across datasets, suggesting that preserving alternative bindings is useful when intermediate decisions are uncertain. Removing KG-augmented retrieval has only minor effects on MuSiQue and HotpotQA (-0.6 and -0.5 points) and a larger effect on 2WikiMQA (-3.0 points). Thus, BindRAG's gains are driven primarily by

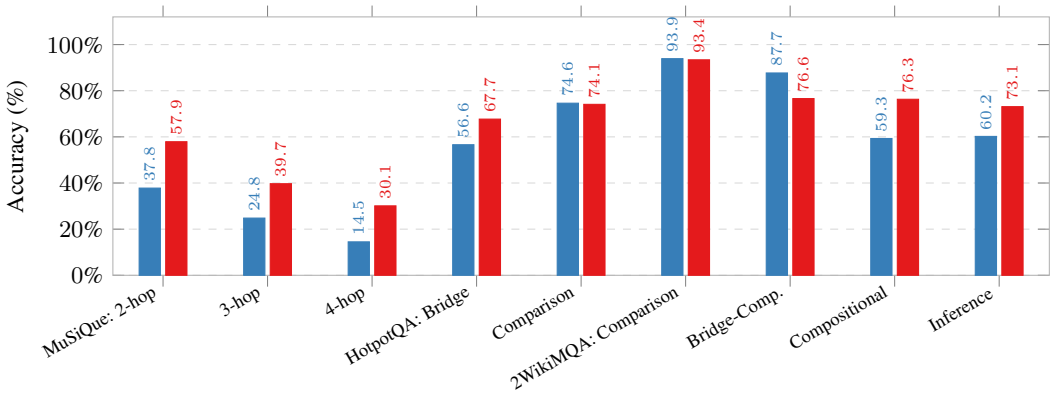

Figure 3: **Accuracy breakdown by reasoning structure.** BindRAG (red) and LogicRAG (blue) are compared by gold hop count on MuSiQue and by question type on HotpotQA and 2WikiMultiHopQA. BindRAG shows the largest gains on questions requiring sequential dependency reasoning, with the advantage widening as hop count increases on MuSiQue and on Compositional/Inference questions on 2WikiMultiHopQA.

dependency-aware execution and query-aware state rather than graph retrieval alone. BindRAG's gains largely persist across GPT-4o-mini and Llama-3.1-8B-Instruct, although performance varies by dataset and metric; full results are reported in Table B.1.

4.6 Analysis

Where does BindRAG help? Figure 3 shows that BindRAG's gains concentrate on questions requiring sequential dependency resolution. On MuSiQue, BindRAG substantially outperforms LogicRAG across 2–4 hops, with the advantage persisting as reasoning chains become longer. On HotpotQA, the gain is large for Bridge questions (+11.1 points) but negligible for Comparison questions, consistent with binding being most useful when one fact identifies the entity needed for subsequent retrieval. A similar pattern appears on 2WikiMQA: BindRAG substantially improves Compositional and Inference questions while leaving simple Comparison questions nearly unchanged. Bridge-Comparison is an exception, where LogicRAG performs better, indicating that explicit sequential execution is not uniformly preferable across all reasoning structures.

What drives the improvement? Together, the ablations and structural analysis show that BindRAG's gains come primarily from resolving sequential retrieval dependencies rather than from graph augmentation. Binding propagation makes intermediate answers actionable as downstream query terms, while query-aware memory preserves evidence and uncertainty for final resolution.

5 Conclusion

We introduced BindRAG, a multi-hop retrieval framework that represents intermediate answers as explicit bindings and uses them to instantiate downstream retrieval queries. By combining binding propagation with query-aware memory, BindRAG makes retrieval dependencies executable while preserving intermediate evidence and uncertainty for final resolution.

In our main experiments, BindRAG consistently outperforms strong RAG baselines across three multi-hop QA benchmarks, with the largest gains on questions requiring sequential retrieval dependencies. Ablations confirm that binding propagation and query-aware memory drive these improvements. These results highlight a key distinction: decomposition specifies *what* to retrieve, whereas binding determines *how* later evidence becomes retrievable. Future work will focus on improving robustness to incorrect intermediate bindings.

AI USE STATEMENT

Generative AI tools were used to assist with research ideation and execution, drafting portions of the manuscript, and improving the clarity and presentation of the writing. AI-assisted suggestions related to the research methodology and experimental analysis were independently evaluated by the authors, and all AI-assisted manuscript content was reviewed and revised by the authors. The authors take full responsibility for the methodology, implementation, experiments, results, claims, and final content of this work.

REFERENCES

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

Anthropic. The claude 3 model family: Opus, sonnet, haiku. 2024. URL `https://api.semanticscholar.org/CorpusID:268232499`.

Shengyuan Chen, Chuang Zhou, Zheng Yuan, Qinggang Zhang, Zeyang Cui, Hao Chen, Yilin Xiao, Jiannong Cao, and Xiao Huang. You don't need pre-built graphs for rag: Retrieval augmented generation with adaptive reasoning structures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30270–30278, 2026.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 719–729, 2024.

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.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pp. 10421–10430. PMLR, 2023.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. In *EMNLP (Findings)*, pp. 10746–10761, 2025.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 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. *arXiv preprint arXiv:2502.14802*, 2025.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh Halappanavar, Ryan A Rossi, Subhabrata Mukherjee, Xianfeng Tang, et al. Retrieval-augmented generation with graphs (graphrag). *arXiv preprint arXiv:2501.00309*, 2024.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907, 2024.

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*, pp. 6609–6625, 2020.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251): 1–43, 2023.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks. In *International Conference on Learning Representations*, 2023.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *International Conference on Machine Learning*, 2024.

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

Dawei Li, Shu Yang, Zhen Tan, Jae Young Baik, Sukwon Yun, Joseph Lee, Aaron Chacko, Bojian Hou, Duy Duong-Tran, Ying Ding, et al. Dalk: Dynamic co-augmentation of llms and kg to answer alzheimer's disease questions with scientific literature. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 2187–2205, 2024.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems*, 44(2):1–52, 2025.

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*, pp. 5687–5711, 2023.

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

Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 155–161, 2024. doi: 10.1109/MIPR62202.2024.00031.

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*, pp. 9248–9274, 2023.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, 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*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL `https://aclanthology.org/2022.tacl-1.31/`.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023.

Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. Archrag: Attributed community-based hierarchical retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 15868–15876, 2026.

Yu Wang, Nedim Lipka, Ryan A Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19206–19214, 2024.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 66145–66178, 2026.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv:2009.12756*, 2020.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 730–740, 2024.

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*, pp. 2369–2380, 2018.

Qinggang Zhang, Zhishang Xiang, Yilin Xiao, Le Wang, Junhui Li, Xinrun Wang, and Jinsong Su. Faithfulrag: Fact-level conflict modeling for context-faithful retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 21863–21882, 2025.

Huachi Zhou, Wenqi Pei, Qinggang Zhang, Daochen Zha, Hao Chen, and Xiao Huang. Self-monitoring large language models for click-through rate prediction. 2025a. URL `https://openreview.net/forum?id=jNCwczhHLP`.

Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, et al. In-depth analysis of graph-based rag in a unified framework. *arXiv preprint arXiv:2503.04338*, 2025b.

Table B.1: Backbone robustness of BindRAG and LogicRAG across three multi-hop QA benchmarks. Str-Acc denotes string-matching accuracy, and LLM-Acc denotes LLM-based answer accuracy.

Backbone	Method	HotpotQA		MuSiQue		2WikiMQA	
		Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.	Str-Acc.	LLM-Acc.
Qwen2.5-32B-Instruct	LogicRAG	60.9	69.5	29.3	36.8	74.6	70.5
	BindRAG	**70.3**	**74.7**	**48.3**	**52.6**	**80.1**	**76.0**
GPT-4o-mini	LogicRAG	62.3	69.2	32.5	39.3	69.1	66.0
	BindRAG	**69.7**	**72.3**	**43.6**	**47.4**	**69.7**	**65.5**
Llama-3.1-8B-Instruct	LogicRAG	53.9	61.8	23.9	27.2	55.0	52.5
	BindRAG	**59.2**	**62.5**	**28.2**	**32.4**	**50.4**	**46.9**

A Datasets

Table A.1 summarizes the evaluation datasets and corpus statistics. The sampled HotpotQA set consists of 81.1% bridge and 18.9% comparison questions. For MuSiQue, 86.9% follow chain reasoning and 13.1% require branch-join reasoning. For 2WikiMultiHopQA, the distribution is 41.3% compositional, 24.4% comparison, 23.5% bridge-comparison, and 10.8% inference.

Table A.1: Statistics of the evaluation datasets. We randomly sample 1,000 questions from each benchmark for evaluation.

Dataset	#Eval	Corpus Size	Reasoning Type	#Hops
HotpotQA	1,000	9,811	Bridge / Comparison	2
MuSiQue	1,000	11,656	Chain / Branch-Join	2–4
2WikiMQA	1,000	6,119	Compositional / Comparison / Bridge-Comparison / Inference	2

B Analysis

B.1 Robustness across language-model backbones.

Table B.1 evaluates whether the behavior of BindRAG depends on the Qwen2.5-32B backbone used in the main experiments. BindRAG remains effective when the backbone is replaced with GPT-4o-mini or Llama-3.1-8B-Instruct, although absolute performance decreases with weaker backbones. With GPT-4o-mini, BindRAG reaches 69.7 on HotpotQA, 43.6 on MuSiQue, and 69.7 on 2WikiMQA. On MuSiQue, this remains 11.1 points above LogicRAG using the same backbone (32.5). With the substantially smaller Llama-3.1-8B-Instruct model, BindRAG obtains 28.2 on MuSiQue compared with 23.9 for LogicRAG. However, the gains are not uniform across all settings; on 2WikiMultiHopQA with Llama-3.1-8B-Instruct, BindRAG underperforms LogicRAG, suggesting that the effectiveness of binding-aware execution can depend on both dataset characteristics and backbone capacity.

These results indicate that BindRAG is not specific to a single language model. At the same time, the reduction from Qwen2.5-32B to Llama-3.1-8B shows that explicit retrieval structure does not eliminate the need for a capable reader: binding extraction and final evidence synthesis still depend on the underlying model.

B.2 Sensitivity to retrieval depth.

Figure B.1 examines whether BindRAG's gains can be explained simply by retrieving more passages. Across all three datasets, increasing the retrieval depth from $k = 1$ to $k = 3$–5 improves performance, while further increasing the evidence budget yields only marginal changes. The effect is most pronounced on MuSiQue: Str-Acc increases from 40.8 at $k = 1$ to 47.6 at $k = 3$ and 48.3

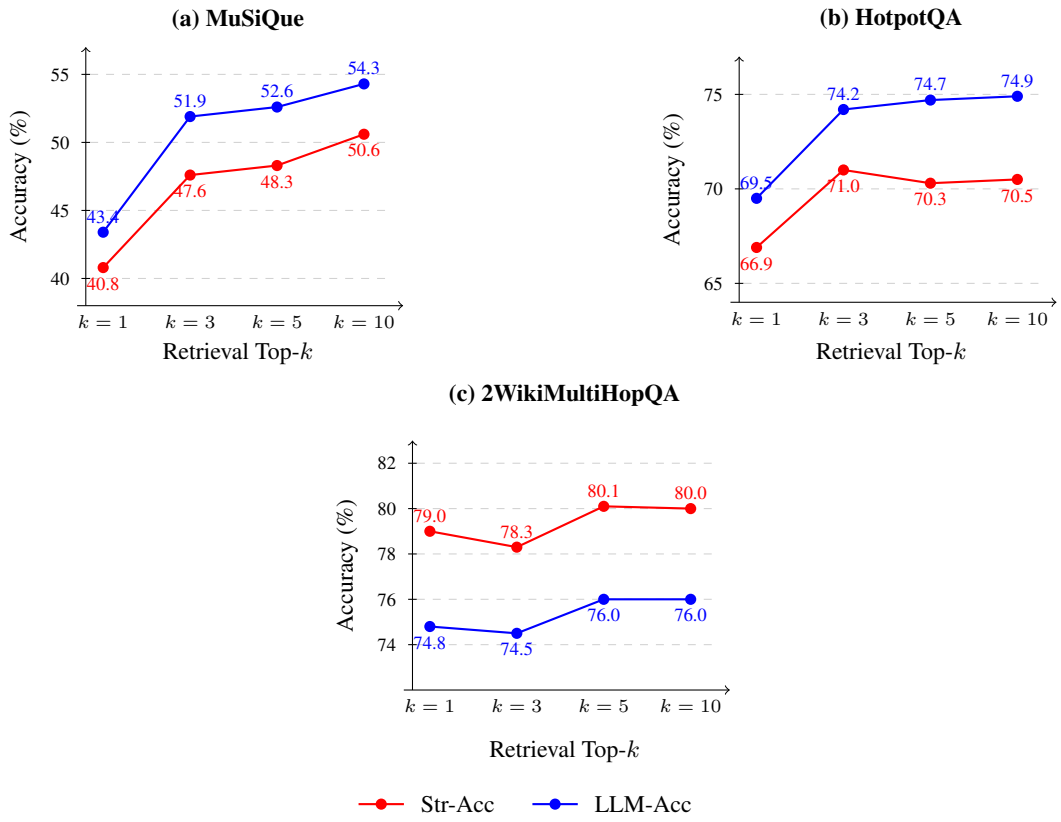

Figure B.1: **Sensitivity to retrieval depth across datasets.** We vary the retrieval depth $k \in \{1, 3, 5, 10\}$ while keeping all other components of BindRAG fixed. Performance generally improves as retrieval depth increases from $k = 1$ to $k = 3$–5 and remains relatively stable thereafter, indicating that BindRAG is robust to the choice of retrieval depth.

at the default $k = 5$, while LLM-Acc rises from 43.4 to 51.9 and 52.6, respectively. Increasing the budget further to $k = 10$ gives only a modest additional gain in Str-Acc (50.6) and a similarly small change in LLM-Acc (54.3).

The same saturation pattern is observed on HotpotQA and 2WikiMultiHopQA. On HotpotQA, LLM-Acc improves from 69.5 at $k = 1$ to 74.7 at $k = 5$, with little change thereafter, whereas Str-Acc varies only slightly after $k = 3$. On 2WikiMultiHopQA, retrieval depth has an even smaller effect: LLM-Acc changes from 74.8 at $k = 1$ to 76.0 at $k = 5$, and Str-Acc remains within a narrow range of 78.3–80.1. Notably, 2WikiMultiHopQA consistently exhibits higher Str-Acc than LLM-Acc across retrieval depths, reversing the ordering observed on HotpotQA and MuSiQue. A similar dataset-specific reversal is visible in the top-k analysis of LogicRAG (Chen et al., 2026), suggesting that this behavior is not specific to BindRAG but may instead reflect characteristics of the benchmark and evaluation metrics. In particular, 47.9% of 2WikiMultiHopQA consists of comparison or bridge-comparison questions, for which substring-based evaluation may be more permissive than semantic LLM-based judgment.

Overall, these results indicate that BindRAG does not rely on an unusually large retrieval budget. Most of the benefit is already obtained by $k = 3$–5, and increasing the evidence budget beyond $k = 5$ provides only limited additional benefit. We therefore use $k = 5$ as the default operating point, balancing retrieval coverage with computational and context costs while maintaining strong performance across all three benchmarks.

C Additional Case Studies

We further examine representative examples in which BindRAG succeeds while the evaluated baselines fail. We compare the main BindRAG configuration ($k = 5$) against VanillaRAG, G-Retriever, DALK, LightRAG, KGP, GraphRAG, RAPTOR and LogicRAG under the same top-5 retrieval setting.

Case selection. To reduce evaluation artifacts, we identify examples for which BindRAG is correct and all evaluated baselines are incorrect under both Str-Acc and LLM-Acc. From this strict subset, we select one representative example per benchmark with a clear intermediate binding chain. Across the three cases, the baselines either terminate at an intermediate entity, confuse its semantic role, or fail to reliably resolve the entity required for the next hop.

C.1 HotpotQA: Terminating at an Intermediate Entity

Question. *Who became a star as a comic book character created by Gerry Conway and Bob Oksner?*

Gold / BindRAG. Megalyn Echikunwoke / Megalyn Echikunwoke

Binding trace. BindRAG decomposes the question into two dependent hops:

$$\text{Gerry Conway and Bob Oksner} \xrightarrow{\text{created character}} \boxed{\text{Vixen}}$$
$$\boxed{\text{Vixen}} \xrightarrow{\text{became a star as}} \text{Megalyn Echikunwoke.}$$

At the first hop, BindRAG extracts the ranked candidates *Vixen*, *Mari Jiwe McCabe*, and *Codename: Assassin*, and selects $\$0 =$ *Vixen* as the intermediate binding. It then substitutes this binding into the second-hop query, *Who became a star as Vixen?*, and retrieves evidence supporting *Megalyn Echikunwoke* as the final answer.

Baseline behavior. Six of the eight baselines—VanillaRAG, RAPTOR, GraphRAG, KGP, DALK, and LogicRAG—predict *Vixen*. This is the correct intermediate entity identified at the first hop, but not the answer to the original question. The remaining baselines predict *the flash* (G-Retriever) and *Hercule Poirot* (LightRAG).

This example illustrates a characteristic intermediate-entity failure: several baselines successfully identify the bridge entity *Vixen* but terminate the reasoning chain before completing the dependent hop. BindRAG instead explicitly binds the intermediate entity and conditions subsequent retrieval on it, allowing the reasoning chain to continue to the requested answer.

C.2 MuSiQue: Preserving the Role of an Intermediate Entity

Question. *What is the place of birth of who does the voice of Stan on the series that Insecurity is part of?*

Gold / BindRAG. Denver / Denver, Colorado

Binding trace. The question requires three dependent hops:

$$\textit{Insecurity} \xrightarrow{\text{part of series}} \boxed{\textit{South Park}} \xrightarrow{\text{voice actor for Stan}} \boxed{\textit{Trey Parker}} \xrightarrow{\text{place of birth}} \textit{Denver, Colorado}.$$

The corresponding typed operations expect `Creative_work`, `Person`, and `Place`, respectively. BindRAG therefore maintains both the identity and the semantic role of each intermediate binding as it progresses through the reasoning chain.

Baseline behavior. The baselines exhibit several forms of entity and type confusion. LightRAG predicts *South Park, Colorado*, while RAPTOR, GraphRAG, and KGP predict *United States*. VanillaRAG and G-Retriever return *Agana Naval Air Station, Guam* and *heidelberg*, respectively. DALK returns *Unknown*, while LogicRAG returns *No*.

This example is particularly informative because *South Park* can denote both a creative work and a geographic place. LightRAG's prediction, *South Park, Colorado*, illustrates how an intermediate work entity can be confused with the requested final answer type. BindRAG instead maintains *South Park* as a `Creative_work` binding, resolves *Trey Parker* at the next hop, and only then retrieves the requested birthplace.

C.3 2WikiMultiHopQA: Grounding an Implicit Relation

Question. *Who is the father-in-law of John Ernest, Duke of Saxe-Eisenach?*

Gold / BindRAG. William IV, Landgrave of Hesse-Kassel / William IV, Landgrave of Hesse-Kassel

Binding trace. Rather than treating *father-in-law* as a single retrieval relation, BindRAG decomposes the question into two dependent operations:

$$\text{John Ernest, Duke of Saxe-Eisenach} \xrightarrow{\text{spouse}} \boxed{\$0} \xrightarrow{\text{father}} \text{William IV, Landgrave of Hesse-Kassel.}$$

The first hop resolves the spouse entity required by the second hop. This intermediate binding is then substituted into the next sub-question, *Who is the father of* $\$0$*?*, grounding the second retrieval step on the result of the first.

Baseline behavior. Unlike the HotpotQA example, the baseline predictions are widely distributed across related historical figures. VanillaRAG and RAPTOR predict *Frederick III of the Palatinate*; GraphRAG predicts *Johann Friedrich II*; KGP predicts *John George I, Duke of Saxe-Eisenach*; LightRAG predicts *William Duke of Saxe-Weimar*; G-Retriever predicts *john william iii*; and LogicRAG predicts *Louis IV, Landgrave of Hesse-Kassel*. DALK returns *Unknown Father*.

The dispersion of predictions suggests that the baselines do not reliably ground the intermediate entity needed to resolve the relationship. BindRAG instead decomposes the relation into explicit *spouse* and *father* operations and conditions the second retrieval step on the intermediate binding.

C.4 Summary of Failure Modes

Although the three examples differ in surface form, they reveal a common structural failure mode. In HotpotQA, several baselines retrieve the correct intermediate entity but treat it as the final answer. In MuSiQue, an intermediate entity can be confused with the semantic type of the requested answer. In 2WikiMultiHopQA, the intermediate entity needed to resolve a relational chain is not reliably grounded.

BindRAG addresses these cases by explicitly representing intermediate outputs as typed bindings and conditioning subsequent retrieval on the resolved entities. These examples illustrate that the benefit of BindRAG is not merely associated with retrieving additional evidence, but with preserving and propagating intermediate reasoning states across dependent retrieval steps.

D Analysis of Alternative Binding Strategies

BindRAG deliberately treats uncertainty differently during retrieval and final answer generation. Re-selecting among intermediate candidates can change every downstream query, so an incorrect correction may propagate through the remaining retrieval chain. At the final step, by contrast, there is no downstream retrieval to perturb, allowing alternative candidates to inform the final reading decision without changing the evidence trajectory.

This design is supported by our component analysis: reducing intermediate extraction from three candidates to one has negligible effect on MuSiQue, whereas removing candidate enumeration at

the final commitment step causes a significant degradation. These results suggest that alternative candidates are more useful as final-stage evidence for uncertainty than as opportunities for repeated intermediate correction.

E PROMPTS USED IN BINDRAG

We provide the prompts for the three LLM-driven components central to BindRAG: (1) typed program compilation, (2) ranked candidate extraction, and (3) joint copy-then-answer synthesis. The same prompts are used across HotpotQA, MuSiQue, and 2WikiMultiHopQA.

E.1 TYPED PROGRAM COMPILATION

BindRAG first compiles each question into a typed dependency program. Each Lookup/Hop operation specifies a natural-language sub-question, an anchor entity, the relation to retrieve, and an expected answer type. Dependencies between hops are explicitly represented using slot tokens $\$i$, where $\$i$ denotes the output of operation i.

```
[System]
You are a retrieval-program compiler.  For the question, FIRST decide
hop_count = the number of reasoning hops (Lookup/Hop sub-queries the
question needs).  THEN produce exactly that many Lookup/Hop ops, plus
any Compare/Aggregate step.

Output ONLY a JSON object:  {"hop_count":  N, "program":  [ops...]}.

Each Lookup/Hop op must have the form:
{"kind":  "Lookup"|"Hop", "nl":  self-contained natural-language
sub-question, "anchor":  entity or "$i" (output of op i), "relation":
what to find, "answer_type":  expected answer type}.

"nl" is the FORWARD retrieval question for the hop.  When the hop
depends on an earlier result, write "$i" exactly where that entity
belongs (for example, "What is the nationality of $0?").

Make dependencies EXPLICIT. When a hop needs an earlier result,
reference it as "$i":  use "$i" as the anchor when the previous result
is the subject of the new hop, or mention "$i" explicitly in the
relation/question when it only constrains the subject.  Never leave
a dependency implicit.

Use a literal anchor only for an entity's FIRST mention.

hop_count counts ONLY Lookup/Hop ops, including $i-anchored hops.
Compare/Aggregate steps are excluded from hop_count.

Emit EXACTLY hop_count Lookup/Hop ops -- one per reasoning hop.

[User]
Question:  {few-shot question 1}
Program:  {few-shot program 1}

...

Question:  {few-shot question 10}
Program:  {few-shot program 10}

Question:  {input question}
Program:
```

We use ten fixed few-shot examples drawn from the HotpotQA training set and apply the same examples to all three benchmarks. The examples are disjoint from the evaluation questions and cover common multi-hop structures, including bridge and comparison reasoning.

As an example, one bridge-style demonstration compiles the question "The Oberoi family is part of a hotel company that has a head office in what city?" into the following program:

```
{"hop_count": 2, "program": [
  {"kind":"Hop", "nl":"What hotel company is the Oberoi family part
of?", "anchor":"Oberoi family", "relation":"part of hotel company",
"answer_type":"Organization"},
  {"kind":"Hop", "nl":"In what city is the head office of $0?",
"anchor":"$0", "relation":"head office city", "answer_type":"Place"}
]}
```

E.2 Ranked Candidate Extraction

At each reasoning hop, BindRAG extracts up to three ranked candidate bindings from the retrieved evidence in a single LLM call. The highest-ranked candidate is used for downstream slot substitution, while alternative candidates are retained as uncertain bindings for subsequent retrieval and final synthesis.

```
[System]
You list candidate answers to a sub-question using only the provided
passages. Reply with a numbered list of short answers (an entity,
name, date, or short phrase) -- one per line, no explanation, no
sentences. Give as many candidates as the question asks for, ordered
best first. Reply UNKNOWN only if the passages contain no plausible
candidate at all.

[User]
Passages:
{passage_1}
---
{passage_2}
---
...

Entities and relations already extracted from the passages
above (subject | relation | object). The passages above remain
authoritative; use these only to avoid missing a candidate:
{kg_triples}

Sub-question: {sub_question}

The answer should be {answer_type}. Prefer candidates of that kind
-- the specific value the sub-question asks about, not a different
nearby value of another kind. This is a preference, not a filter:
keep a candidate the passages clearly support even when its kind is
only roughly {answer_type}.

List the 3 most likely answers, best first, one per line, numbered.
Each line must be a short answer only (entity/name/date) with no
explanation. Use UNKNOWN only if the passages contain no plausible
answer at all.

Answers:
```

The expected answer type acts as a preference rather than a hard filter. The KG triples are restricted to those extracted from passages retrieved for the current hop.

E.3 Joint Copy-then-Answer Synthesis

After executing all reasoning hops, BindRAG performs a single joint synthesis over the accumulated execution state. The synthesis input contains the typed decomposition, resolved intermediate bindings, uncertain bindings, alternative binding traces when available, and retrieved evidence grouped by reasoning hop.

Rather than directly generating an unconstrained answer, the model first copies up to three candidate answer spans verbatim from the retrieved passages and then selects the final answer.

```
[System]
You must give ONLY the direct, most concise answer.  Do NOT explain
or add any context.  If the answer is yes/no, say 'Yes.'  or 'No.'; if
it is a name, give only the name; a date, only the date; a number,
only the number; otherwise the briefest phrase.  Always commit to
your single best answer from the passages (use prior knowledge to
disambiguate if needed) -- never say you don't know or that the
passages are insufficient.

[User]
Decomposition:  {"hop_count":  N, "program":  [ops...]}

Resolved sub-answers (intermediate hops):
- {sub_question_1} => {binding_1}
- {sub_question_2} => {binding_2}

Alternative binding trace ($i = {alt_value}) -- an equally supported
reading of one earlier step, with what it leads to:
- {sub_question} => {answer}

Uncertain intermediate bindings (choose using the passages; the first
is not necessarily correct):
- {sub_question} candidates:  {cand_1} | {cand_2} | {cand_3}

Evidence for step 1 -- "{sub_question_1}" [resolved:  {binding_1}]:
{hop1_passage_a}
---
{hop1_passage_b}

...

Evidence for FINAL step -- "{sub_question_H}" [resolved:  {binding_H}]:
{hopH_passage_a}
---
{hopH_passage_b}

Target response length and format:
Provide ONLY the final answer as a short phrase of a few words.  No
explanation, no analysis, no extra sentences.

Question:  {question}

First list up to 3 candidate answers copied VERBATIM from the
passages, most likely first.  Then give the final answer.  Prefer one
of your listed spans; use a different wording only if no span states
the answer the question asks for.

Reply in exactly this format:
CANDIDATES: <span1> | <span2> | <span3>
ANSWER: <final answer>
```

Only the content following `ANSWER:` is used as the final prediction; the candidate spans are retained as an execution trace.