BUBBLERAG: ONE QUESTION, MANY POSSIBLE PATHS TO EVIDENCE IN KNOWLEDGE GRAPHS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph-based retrieval-augmented generation supports knowledge-intensive question answering by connecting evidence across documents. Retrieving useful evidence, however, requires determining how query concepts are represented in the graph and how their supporting facts are connected. A query concept may match multiple nodes or edges, and the usefulness of each match can depend on its connections to other supporting facts. Pruning these alternatives too early may exclude useful evidence, while exploring all of them extensively can be costly. Moreover, relevant evidence may follow different structural patterns, making the resulting candidates difficult to compare. These issues create three challenges: preserving plausible concept matches, exploring diverse connections, and comparing candidates with different evidence structures. To address these challenges, we propose BubbleRAG, a training-free retrieval framework. BubbleRAG first extracts query concepts and retrieves candidate nodes or edges for each concept. It then constructs connected candidate subgraphs to assess these matches together with their connections, and ranks the subgraphs by query-concept coverage and semantic relevance. The top-ranked subgraphs provide starting points for LLM-guided exploration to retrieve additional facts needed for answering. Experiments on HotpotQA, MuSiQue, and 2WikiMultiHopQA show that BubbleRAG achieves the best average F1 and LLM-judged accuracy among the evaluated baselines with both 8B and 30B backbones, with the largest F1 gain reaching 7.99 points on MuSiQue.

1 INTRODUCTION

Large Language Models (LLMs) can answer a wide range of questions, but their reliance on parametric knowledge makes factual errors and outdated answers a concern in knowledge-intensive tasks (Magesh et al., 2024; Huang et al., 2025b). Retrieval-Augmented Generation (RAG) addresses this problem by providing external evidence for answer generation (Fan et al., 2024; Lewis et al., 2021). For multi-hop questions, the required facts often span several documents and entities, making it necessary to recover their connections as well as their individual content (Ji et al., 2022; Lin et al., 2025; Peng et al., 2025). Graph-based RAG makes these connections explicit by organizing entities and relations into a knowledge graph (KG), which can guide the retrieval of evidence across documents (Gutiérrez et al., 2024; Cheng et al., 2025; Zhu et al., 2026).

However, having a graph does not immediately tell the retriever where to find evidence for a particular question. Schema knowledge can help narrow the search, but does not necessarily resolve the ambiguity in concept mappings or evidence connections. A query concept may match several nodes or edges within the same schema, and relevant evidence may follow different relation sequences or branching structures. The retriever therefore needs to select among these alternatives to assemble useful evidence. Using the query *Find a Machine Learning expert* in Figure 1, we illustrate three related difficulties:

- **Grounding diversity.** A query concept may match multiple semantically related nodes or edges. In the upper-left panel of Figure 1, possible matches for *Machine Learning* include Machine Learning itself, Logistic Regression, and Supervised Learning. The usefulness of a match can depend on how it connects to other supporting facts, making it difficult to assess from semantic

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.

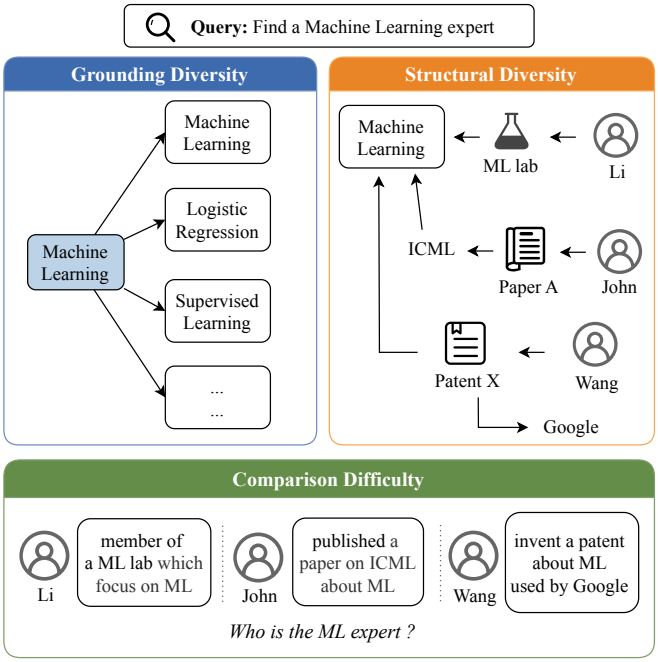


Figure 1: Retrieval challenges illustrated by *Find a Machine Learning expert*: a concept has multiple plausible graph realizations (upper left), supporting facts follow different connections (upper right), and the resulting candidates have different evidence structures to compare (bottom).

similarity alone. Pruning alternatives too early may discard useful evidence, while exploring every plausible match can be costly.

- **Structural diversity.** Even with relevant concept matches, their supporting facts need not share one connection pattern. In the upper-right panel, evidence about Li follows an affiliation through an ML lab, evidence about John follows a publication through Paper A and ICML, and evidence about Wang follows a patent connection. Restricting search to one expected relation sequence may miss other useful structures, while indiscriminate exploration collects unrelated facts. The retriever must discover which connections provide relevant evidence within its search budget.
- **Comparison difficulty.** Exploring different realizations and connections produces candidates with different kinds of support. In the bottom panel, laboratory membership, a publication, and a patent each provide potentially relevant evidence, but the number of edges or the similarity of one node does not directly determine which candidate should be prioritized for further exploration. A common criterion is therefore needed to compare candidates across these structures.

Prior Solutions. Existing methods address these difficulties through different retrieval mechanisms. Schema-aligned approaches generate query patterns or triple structures to match against the graph (Cai et al., 2025; Ma et al., 2023); such patterns can narrow search, but useful connections may be missed when the generated structure does not match the stored evidence. Iterative methods use successive exploration steps to find relevant neighbors (Sun et al., 2024; Ma et al., 2025; LUO et al., 2024; Mavromatis et al., 2025), while relevance-propagation methods spread relevance from retrieved seeds (Gutiérrez et al., 2024; Gutiérrez et al., 2025; Zhuang et al., 2025; Wang et al., 2025). In both cases, useful evidence may be missed when initial matches exclude relevant entry points or exploration gives insufficient priority to promising connections. Pre-indexed methods organize communities or other auxiliary structures before a question arrives (Edge et al., 2024; Liang et al., 2025; Su et al., 2025). Such query-independent organization may not align with the evidence needed for a particular question, whose supporting facts may span multiple retrieval units. These limitations motivate a retrieval framework that adapts candidate evidence structures to each question, while preserving plausible concept matches and prioritizing further exploration under a limited budget.

Our Solution. We propose **BubbleRAG**, a training-free graph retrieval framework that retains alternative matches for each query concept and selects exploration starting points after constructing and comparing candidate evidence graphs. This allows matches to be assessed together with their connections, rather than through concept-level similarity alone. We formulate connected evidence selection as Optimal Informative Subgraph Retrieval (OISR), maximizing mean semantic relevance subject to query-concept coverage. Our generate-and-rank heuristic uses Bubble Expansion to construct connected *candidate evidence graphs* (CEGs), then ranks them by relevance and coverage while allowing partial coverage. Starting from the top-ranked subgraphs as promising entry points, LLM-guided reasoning-aware expansion retrieves additional facts needed to answer the question. The expanded subgraphs and source passages support answer generation.

Contributions. *Problem analysis.* We identify grounding diversity, structural diversity, and comparison difficulty in query-time KG retrieval, exposing a trade-off between prematurely pruning concept matches and extensively exploring their possible connections.

Retrieval framework. We propose BubbleRAG, which postpones the selection of exploration starting points until alternative concept matches have been assembled into candidate evidence graphs and compared by relevance and coverage.

Problem formulation and algorithm. We formulate OISR and establish its NP-hardness and APX-hardness. Our generate-and-rank heuristic combines Bubble Expansion with coverage-aware ranking of structurally diverse candidates.

Empirical evaluation. Experiments on three multi-hop QA benchmarks show the highest average F1 and LLM-judged accuracy among the evaluated methods at both backbone scales. Ablations and retrieval analyses assess the core design choices, while cost analysis quantifies the computational overhead.

2 Related Work

In this section, we review existing graph-based RAG methods by their retrieval mechanisms and discuss their limitations.

Query Rewriting and Schema-Aligned Matching. These approaches translate a question into structured conditions or query patterns for matching against a KG (Cai et al., 2025; Kim et al., 2023; Lee et al., 2025). SimGRAG (Cai et al., 2025) uses an LLM to transform the question into an abstract query graph, then searches for aligned subgraphs. KG-GPT (Kim et al., 2023) decomposes a question into smaller reasoning units and retrieves relevant graph triples for subsequent inference. SAFE (Lee et al., 2025) studies schema-driven approximate matching for KG querying. Generated conditions constrain the search space, but may exclude useful evidence when their labels or structural requirements do not align with how facts are represented in the graph.

Iterative Multi-Hop Exploration. Another line of work uses successive reasoning and retrieval steps to construct evidence paths or structures (Sun et al., 2024; Ma et al., 2025; LUO et al., 2024; Mavromatis et al., 2025; Wu et al., 2026). ToG (Sun et al., 2024) uses LLM-guided beam search over graph relations and entities. ToG2 (Ma et al., 2025) combines graph exploration with document retrieval, using textual information to guide the search. RoG (LUO et al., 2024) generates relation sequences as reasoning plans and retrieves graph paths consistent with them. Related work also constructs adaptive reasoning structures during retrieval rather than relying on a pre-built graph (Chen et al., 2026). These methods allow evidence collection over multiple steps, but errors in initial matches or intermediate pruning can prevent later steps from reaching relevant facts.

Graph-Based Relevance Propagation. Propagation methods spread relevance from semantic seeds through graph connections, often using Personalized PageRank (PPR) (Page et al., 1999; Gutiérrez et al., 2024; Gutiérrez et al., 2025; Zhuang et al., 2025; Wang et al., 2025). HippoRAG (Gutiérrez et al., 2024) links query entities to graph nodes and uses PPR to support associative passage retrieval. HippoRAG2 (Gutiérrez et al., 2025) incorporates passage nodes into the graph and uses embedding-based retrieval to help initialize relevance propagation. LinearRAG (Zhuang et al., 2025) builds a relation-free graph with entity, sentence, and passage layers, combining local activation with PPR-

based global aggregation. AGRAG (Wang et al., 2025) filters query-matched triples with an LLM, connects their terminals through a Steiner tree, and greedily expands a Minimum Cost Maximum Influence (MCMI) subgraph using PPR influence and semantic edge costs. The resulting subgraph and hybrid-retrieved passages support answer generation. Seed selection and the propagation rule determine how relevance is distributed; poorly matched seeds can direct retrieval toward irrelevant graph regions.

Structure-Augmented Retrieval with Auxiliary Graphs. Other methods construct hierarchical or multi-partite indices before processing individual questions (Edge et al., 2024; Liang et al., 2025; Su et al., 2025; Huang et al., 2025a; Zhu et al., 2025; Luo et al., 2025). GraphRAG (Edge et al., 2024) builds an entity graph, applies Leiden community detection to obtain a hierarchy of communities, and generates summaries for retrieval. KAG (Liang et al., 2025) connects graph knowledge with source chunks through mutual indexing and organizes concepts and instances to support retrieval and reasoning. ClueRAG (Su et al., 2025) constructs a multi-partite index linking text chunks, knowledge units, and entities. Further approaches use hierarchical knowledge organization (Huang et al., 2025a), KG-guided evidence organization (Zhu et al., 2025), or hypergraph representations (Luo et al., 2025). Such structures can be reused across questions, although the evidence needed for a particular query may span multiple precomputed units. Comparative studies examine the retrieval behavior and preparation costs of different graph-based RAG designs (Zhou et al., 2026).

3 METHODOLOGY

3.1 FRAMEWORK OVERVIEW

Figure 2 shows how BubbleRAG turns alternative concept matches into evidence for answering. The framework takes a question q, a knowledge graph $G = (V, E)$, and the associated source passages as input. Semantic anchor grouping maps query concepts to groups of candidate nodes or edges and assigns each group an importance weight. These groups guide Bubble Expansion in constructing candidate evidence graphs, which are ranked by concept coverage and semantic relevance. The selected candidates serve as starting points for LLM-guided expansion. Finally, their expanded evidence is merged and combined with source passages for answer generation. Appendix A summarizes the notation.

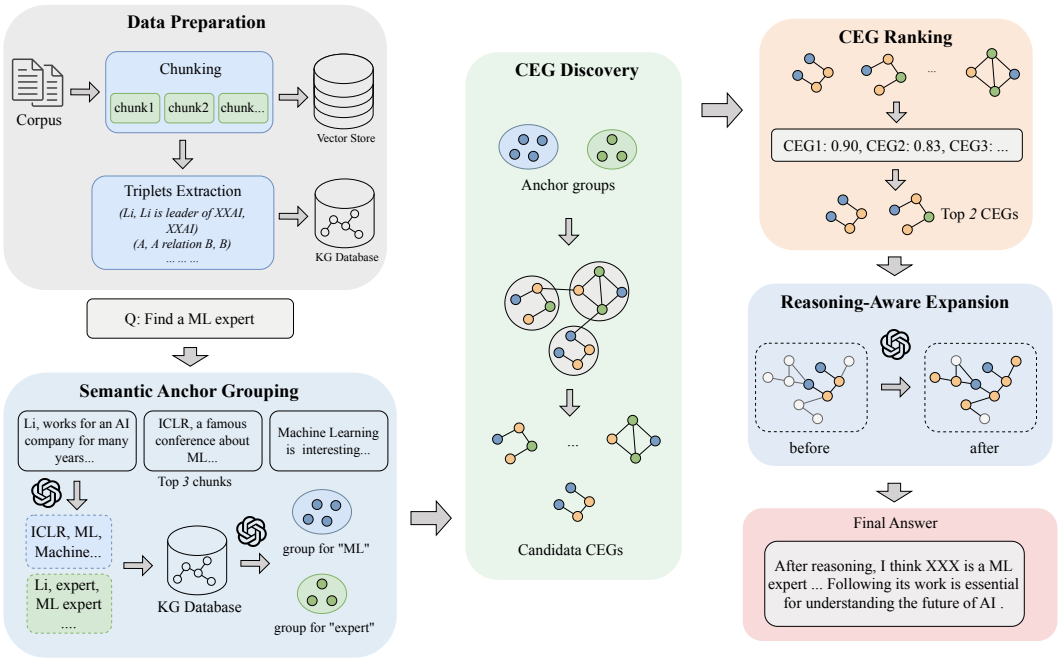


Figure 2: Pipeline of BubbleRAG.

Data Preparation. Following standard LLM-based graph construction (Guo et al., 2025; Gutiérrez et al., 2024), we segment the corpus into chunks and extract factual triples $(\text{head}, \text{relation}, \text{tail})$ from each chunk. The triples are consolidated into a graph whose nodes represent entities and whose edges represent relations. Each edge retains a searchable string containing its head, relation, and tail, together with its source-chunk identifier. These links allow the generator to use the original passages alongside the retrieved triples, including details that graph extraction may omit. The graph extraction prompts are provided in Appendix B.1.

3.2 Semantic Anchor Grouping

For each query concept, BubbleRAG retains alternative graph matches as *anchors* and organizes them into semantic anchor groups. Before retrieving graph candidates, we retrieve text chunks relevant to the question and ask the LLM whether they provide sufficient evidence to answer it. If so, the system returns the answer without graph traversal; otherwise, the retrieved chunks provide context for the next steps.

We next identify candidate graph elements for the query concepts. An LLM extracts entities, relations, and attributes from the question and refines them into retrieval phrases using the question context and retrieved chunks. Directly using the original query terms can produce overly broad matches or miss evidence expressed differently in the graph. We address these two issues through anchor specialization and schema relaxation.

Anchor Specialization. BubbleRAG makes broad terms more specific by attaching the relevant query context. For *When did Lothair II's mother die?*, it rewrites *mother* as *Lothair II's mother*, focusing candidate retrieval on the intended relationship.

Schema Relaxation. BubbleRAG uses the retrieved chunks to identify expressions that fully or partially match the query conditions and includes them as additional retrieval phrases. For example, a condition such as *married twice* may be supported by a passage naming two spouses without using that exact expression. These expressions broaden candidate retrieval while the original question retains the full condition for answering. Query concepts without a passage match remain in their original or specialized form.

The resulting phrases $\{t_i\}$ are used to retrieve top-k candidate nodes or edges by embedding similarity.

Anchor Grouping and Weighting. We merge and deduplicate the retrieved candidates. Given the question and the merged candidates, the LLM organizes alternative matches for each query concept into a semantic anchor group $S_i \subseteq V \cup E$ and assigns an importance weight w_i to each group. The weights reflect the concept's importance to the question and the specificity of its candidate matches, favoring informative matches over generic ones. The nonnegative weights are normalized to $\sum_i w_i = 1$ and used to penalize missing concepts during subsequent subgraph ranking.

The groups $\mathcal{S} = \{S_1, \ldots, S_m\}$ preserve alternative concept matches for subsequent subgraph construction and ranking.

3.3 Initial Evidence Subgraph Retrieval

We connect alternative concept matches into *candidate evidence graphs* (CEGs) and rank them as starting points for further exploration. We first define the selection objective, then describe candidate discovery and ranking.

Initial Selection Objective. Let z_q and z_x denote embeddings of the query and a graph element. Nodes and edges share the relevance measure $\text{val}(x) = \max\{0, \cos(z_q, z_x)\} \in [0, 1]$, with edges represented by their head–relation–tail text.

Definition 1 (Optimal Informative Subgraph Retrieval)**.** *Given $G = (V, E)$ and candidate groups $\mathcal{S} = \{S_1, \ldots, S_m\}$, find a connected subgraph $T = (V_T, E_T)$ covering every group, $S_i \cap (V_T \cup$*

$E_T) \neq \varnothing$ *for all* i*, that maximizes*

$$\text{Value}(T) = \frac{1}{|V_T \cup E_T|} \sum_{x \in V_T \cup E_T} \text{val}(x). \tag{1}$$

This objective reflects our assumption that a useful initial evidence subgraph should cover the key query concepts while limiting irrelevant content. Since OISR is NP-hard and APX-hard (Theorem 1, Appendix B.3), we adopt a generate-and-rank heuristic: Bubble Expansion constructs candidates through low-cost connections, and candidate ranking evaluates their relevance and concept coverage.

Phase 1: CEG Discovery via Bubble Expansion. Bubble Expansion generates candidate subgraphs by exploring low-cost connections among alternative anchors. Inspired by shortest-path heuristics for Group Steiner Tree, it uses the following relevance gap as a traversal cost:

$$\delta(x) = 1 - \text{val}(x), \qquad x \in V \cup E. \tag{2}$$

The implementation uses positive traversal costs $c(x) = \max\{\epsilon, 1 - \cos(z_q, z_x)\}$, with $\epsilon = 0.01$, and a default cost when an embedding is unavailable (Appendix B.4). Lower-cost connections are explored first. Figure 3 illustrates the discovery process, which proceeds in three steps.

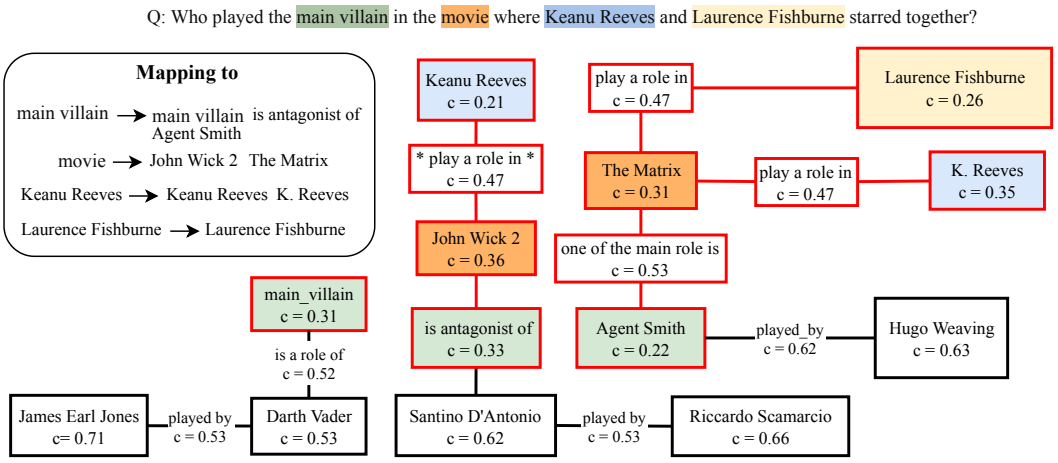


Figure 3: Candidate evidence discovery with Bubble Expansion. Query colors identify semantic anchor groups; red borders mark graph elements incorporated into a candidate evidence graph.

1. **Localize the search.** We collect neighborhoods around the anchors using a hop limit h. Within the resulting undirected search graph, each triple is represented by head–relation–tail links, allowing both entity and relation anchors to initiate exploration.
2. **Expand from anchors.** Each anchor independently expands its search region in increasing order of accumulated traversal cost. Each round settles at most one new vertex per anchor and records predecessor links for path recovery.
3. **Merge intersecting searches.** When searches from multiple anchors reach the same entity vertex, their recovered paths are merged into a connected candidate subgraph. Merging multiple paths allows branching structures. Candidates are deduplicated by their edge sets, and search stops after B rounds or when all queues are exhausted.

If no candidate subgraph is found within the search budget, we fall back to the initial anchors as starting points for reasoning-aware expansion. Appendix B.4 details candidate discovery and the fallback procedure.

Phase 2: CEG Ranking. Candidate subgraphs may cover different subsets of the query concepts. We therefore relax the strict coverage requirement of OISR into an importance-weighted penalty. Let $\bar{c}(T)$ be the mean traversal cost of the entity and relation vertices representing T in the search graph. We assign each $T \in \mathcal{C}$ the score:

$$\mathrm{Score}(T) = \frac{1}{\bar{c}(T)\exp(\alpha r_{\mathrm{miss}}(T))}, \tag{3}$$

where the missing concept weight is

$$r_{\mathrm{miss}}(T) = \sum_{i:S_i \cap (V_T \cup E_T) = \varnothing} w_i. \tag{4}$$

Lower mean cost and more complete concept coverage yield higher scores. For fully covered candidates, $r_{\mathrm{miss}}(T) = 0$, so ranking depends only on mean cost. Under the ideal relevance gap $c(x) = \delta(x)$, minimizing this mean is equivalent to maximizing the OISR value. Partial candidates remain eligible, but missing a more important concept incurs a larger penalty.

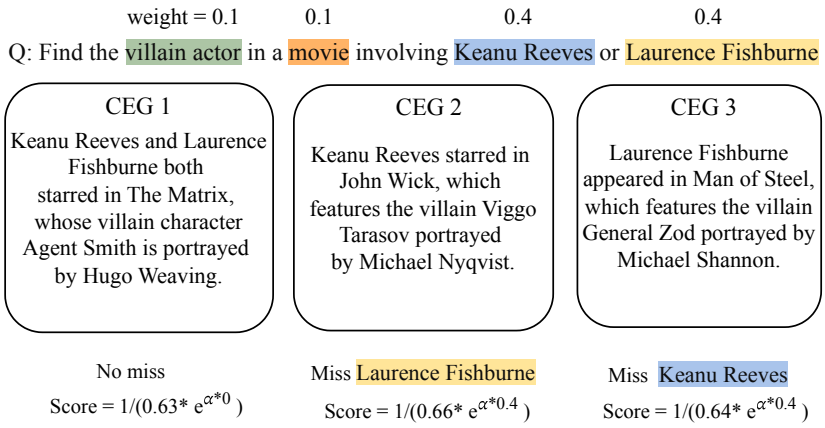


Figure 4: Ranking candidate evidence graphs by semantic relevance and weighted concept coverage. Colors indicate query concepts, and α controls the penalty for missing concepts.

Figure 4 illustrates the coverage preference: larger α favors coverage, while $\alpha = 0$ ranks by mean cost alone. We rank the candidates in descending score order and retain the top-n as starting points for subsequent expansion.

3.4 Reasoning-Aware Expansion and Answer Generation

Reasoning-aware expansion uses the selected candidates to guide further evidence retrieval. For the question in Figure 3, an initial CEG may connect Keanu Reeves and Laurence Fishburne to *The Matrix*. Given this evidence, the LLM can focus subsequent exploration on the film's main villain and the actor portraying that character.

For each top-ranked T^*, BubbleRAG retrieves the immediate neighbors of the current evidence boundary and prompts the LLM with the question and accumulated evidence to select relevant additions. At step ℓ, the evidence becomes

$$T^*_\ell = T^*_{\ell-1} \cup \mathrm{Selected}(\mathcal{N}_\ell),$$

where $\mathcal{N}_\ell$ contains the frontier neighbors and connecting relations. Expansion processes candidates in rank order and stops at depth d or when the LLM judges the evidence sufficient. It can retrieve facts beyond the original anchors and repair omissions from initial retrieval.

Answer Generation. We merge expanded candidates into G_{final}, remove duplicate nodes and edges, and recover source passages C_{text} through their stored identifiers. The final prompt combines the question, serialized graph triples, and passages, allowing the LLM to use both explicit connections and their textual context.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and Metrics. We evaluate on 1,000 validation questions each from HotpotQA (Yang et al., 2018), MuSiQue (Trivedi et al., 2022), and 2WikiMultiHopQA (Ho et al., 2020), using their associated corpora for retrieval. Following prior evaluation protocols (Gutiérrez et al., 2024; Su et al., 2025), we measure token-overlap F1 and LLM-as-a-Judge Accuracy (ACC_L). Qwen3-8B (Yang et al., 2025) judges whether an answer is semantically equivalent to the reference, complementing lexical overlap when valid answers differ in wording.

Baselines. We compare with Vanilla LLM, CoT prompting (Wei et al., 2023), and passage-based NaiveRAG (Gao et al., 2024). The graph-based baselines include schema-oriented KG-GPT (Kim et al., 2023) and SimGRAG (Cai et al., 2025); iterative graph exploration with ToG (Sun et al., 2024); and relevance propagation with HippoRAG2 (Gutiérrez et al., 2025). Baselines using auxiliary retrieval structures include RAPTOR (Sarthi et al., 2024), ClueRAG (Su et al., 2025), and the local, global, and hybrid modes of LightRAG (Guo et al., 2025).

Implementation. The main comparison uses 8B and 30B backbone scales, with Qwen3-Embedding-8B (Zhang et al., 2025) for embedding retrieval across methods. All methods are provided with 15 text chunks per query. BubbleRAG uses $h = 6$, $B = 10$, $k = 8$, $n = 2$, $d = 6$, and $\alpha = 1.0$ by default. Component ablations use the Qwen 8B backbone on 1,000 questions each from 2Wiki and HotpotQA; parameter sensitivity uses the same backbone on 1,000 2Wiki questions. All other baseline settings follow their respective defaults. The machine configuration is provided in Appendix C.

4.2 OVERALL PERFORMANCE

Table 1: Comparison of Different Methods on Multi-hop QA Benchmarks. The best results are highlighted in bold. The rightmost two columns show the average F1 and ACC_L across all six dataset–backbone settings.

Method	30B Model						8B Model						Average	
	HotpotQA		MuSiQue		2Wiki		HotpotQA		MuSiQue		2Wiki		F1	ACC_L
	F1	ACC_L	F1	ACC_L	F1	ACC_L	F1	ACC_L	F1	ACC_L	F1	ACC_L		
Vanilla LLM	17.84	29.70	5.80	10.40	20.14	30.30	11.41	30.10	5.18	12.90	17.46	29.90	12.97	23.88
CoT + LLM	18.83	33.50	8.42	13.20	18.25	25.50	16.35	30.40	7.11	12.00	18.11	25.50	14.51	23.35
Naive RAG	73.07	80.50	43.68	42.20	52.15	54.80	67.41	75.00	34.53	35.00	50.23	52.80	53.51	56.72
KG-GPT	49.57	61.90	30.74	31.50	55.52	60.40	31.87	57.50	19.08	27.80	45.22	47.50	38.67	47.77
SimGRAG	30.71	36.10	12.85	12.60	33.98	35.70	20.62	31.40	7.12	10.50	31.50	32.00	22.80	26.38
ToG	40.07	48.50	16.76	21.60	35.88	38.60	23.56	35.30	8.39	13.90	20.22	23.10	24.15	30.17
HippoRAG2	71.72	80.20	45.04	47.20	67.65	71.70	**73.33**	79.00	42.92	41.60	62.40	66.70	60.51	64.40
RAPTOR	66.75	68.00	33.10	32.20	40.11	43.90	62.66	68.60	29.35	28.90	36.25	40.20	44.70	46.97
Clue-RAG	62.87	67.30	33.01	33.90	45.63	47.40	54.95	69.20	27.39	27.80	35.87	46.30	43.29	48.65
LightRAG (local)	57.05	61.80	41.72	43.80	60.17	64.90	54.25	54.30	36.57	37.70	54.61	55.10	50.73	52.93
LightRAG (global)	35.28	39.10	20.56	22.50	23.20	26.30	30.79	30.80	20.11	20.30	23.41	24.00	25.56	27.17
LightRAG (hybrid)	54.87	59.10	39.57	42.20	55.67	60.10	54.32	54.50	34.21	35.10	47.79	48.50	47.74	49.92
BubbleRAG	**74.35**	**82.60**	**53.03**	**53.10**	**72.54**	**76.60**	71.82	**79.10**	**45.15**	**44.00**	**64.97**	**67.60**	**63.64**	**67.17**

Table 1 reports results across three datasets and two backbone scales. BubbleRAG achieves the highest F1 in five of the six settings and the highest ACC_L in all six, with average scores of 63.64 and 67.17, respectively. Compared with HippoRAG2, the strongest baseline on average, BubbleRAG improves F1 and ACC_L by 5.17 and 4.40 points with the 30B backbone, and by 1.10 and 1.13 points with the 8B backbone.

The gains vary across datasets and backbone scales. The largest F1 improvement over HippoRAG2 is on MuSiQue with the 30B backbone, from 45.04 to 53.03. On HotpotQA with the 8B backbone, BubbleRAG obtains lower F1 (71.82 versus 73.33) and similar ACC_L (79.10 versus 79.00).

4.3 COMPONENT ABLATIONS

Table 2 examines candidate preparation, ranking, and reasoning-aware expansion (RAE). Without specialization, generic keywords are used without query-conditioned refinement. Without relaxation, the preliminary passage retrieval used to guide relaxation is removed. Without ranking, discovery stops after producing the first two candidates, which are used directly for expansion. Without RAE, the selected candidates are passed directly to answer generation.

Table 2: Component ablations with the 8B backbone. ACC denotes LLM-judged accuracy.

Method Variant	**2Wiki**		**HotpotQA**	
	F1	**ACC**	**F1**	**ACC**
BubbleRAG (Full)	**64.97**	**67.60**	**71.82**	**79.10**
w/o Anchor Specialization	60.45	65.00	69.28	76.10
w/o Schema Relaxation	53.62	59.20	66.65	75.30
w/o CEG Ranking	58.76	64.40	70.20	78.90
w/o RAE	58.30	60.30	71.60	78.70

Removing schema relaxation produces the largest F1 decrease: 11.35 points on 2Wiki and 5.17 on HotpotQA; this variant also removes preliminary passage retrieval, so the effect reflects both changes. Removing specialization lowers F1 by 4.52 and 2.54 points, respectively, supporting query-conditioned refinement of candidate matches. Removing ranking lowers F1 by 6.21 and 1.62 points, supporting candidate comparison before exploration. Removing RAE lowers F1 by 6.67 and 0.22 points and ACC_L by 7.30 and 0.40 points, showing that its contribution varies across datasets.

4.4 PARAMETER SENSITIVITY

We vary B, d, α, and k one at a time on 2Wiki with the 8B backbone. The default configuration is $h = 6$, $B = 10$, $k = 8$, $n = 2$, $d = 6$, and $\alpha = 1.0$; all other parameters remain fixed in each run.

Table 3: Parameter sensitivity on 2Wiki with the 8B backbone. Parameters are varied individually; bold values indicate defaults.

Search budget B			Expansion depth d			Coverage penalty α			Candidate count k		
Value	F1	ACC_L	Value	F1	ACC_L	Value	F1	ACC_L	Value	F1	ACC_L
5	63.72	67.60	2	62.49	66.40	0.1	63.35	67.20	4	64.38	68.30
10	64.97	67.60	4	63.31	67.20	**1**	64.97	67.60	**8**	64.97	67.60
20	62.56	66.50	**6**	64.97	67.60	2	64.61	68.50	16	68.97	72.90
50	64.66	68.60	8	66.52	70.40	5	66.59	70.50			

Table 3 shows that increasing B does not consistently improve F1, whereas increasing d from 2 to 8 raises F1 from 62.49 to 66.52. A stronger coverage penalty ($\alpha = 5$) yields 66.59 F1, compared with 64.97 at the default. Increasing k from 8 to 16 improves F1 by 4.00 points and ACC_L by 5.30 points.

Additional analyses of retrieval routing, statistical significance, candidate ranking, reasoning-aware expansion, schema information, and efficiency are provided in Appendix D.

5 CONCLUSION

BubbleRAG retains alternative concept matches, constructs and ranks connected evidence subgraphs, and uses LLM-guided expansion to retrieve additional facts. Across three multi-hop QA benchmarks, it achieves the highest average F1 and LLM-judged accuracy among the evaluated methods with both 8B and 30B backbones.

AI USE STATEMENT

During manuscript preparation, we used GPT and Claude for editorial assistance and critical feedback on technical arguments, including proofs, and on experimental design and interpretation. The authors are responsible for the final content.

REFERENCES

Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1–2):123–134, 2000. doi: 10.1016/S0304-3975(98)00158-3.

Yuzheng Cai, Zhenyue Guo, YiWen Pei, WanRui Bian, and Weiguo Zheng. SimGRAG: Leveraging similar subgraphs for knowledge graphs driven retrieval-augmented generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, July 2025.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(36):30270–30278, 2026. doi: 10.1609/aaai.v40i36.40278. URL https://doi.org/10.1609/aaai.v40i36.40278.

Mingyue Cheng, Yucong Luo, Jie Ouyang, Qi Liu, Huijie Liu, Li Li, Shuo Yu, Bohou Zhang, Jiawei Cao, Jie Ma, Daoyu Wang, and Enhong Chen. A survey on knowledge-oriented retrieval-augmented generation, 2025. URL https://arxiv.org/abs/2503.10677.

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.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, pp. 6491–6501, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3671470. URL https://doi.org/10.1145/3637528.3671470.

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, 2024. URL https://arxiv.org/abs/2312.10997.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.568. URL https://aclanthology.org/2025.findings-emnlp.568/.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: neurobiologically inspired long-term memory for large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=LWH8yn4HS2.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational*

Linguistics, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Haoyu Huang, Yongfeng Huang, Junjie Yang, Zhenyu Pan, Yongqiang Chen, Kaili Ma, Hongzhi Chen, and James Cheng. Retrieval-augmented generation with hierarchical knowledge, 2025a. URL https://arxiv.org/abs/2503.10150.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55, January 2025b. ISSN 1558-2868. doi: 10.1145/3703155. URL http://dx.doi.org/10.1145/3703155.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, February 2022. ISSN 2162-2388. doi: 10.1109/tnnls.2021.3070843. URL http://dx.doi.org/10.1109/TNNLS.2021.3070843.

Jiho Kim, Yeonsu Kwon, Yohan Jo, and Edward Choi. Kg-gpt: A general framework for reasoning on knowledge graphs using large language models, 2023. URL https://arxiv.org/abs/2310.11220.

Sangoh Lee, Sungho Park, and Wook-Shin Han. SAFE: Schema-driven approximate distance join for efficient knowledge graph querying. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 17464–17489, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.883. URL https://aclanthology.org/2025.emnlp-main.883/.

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, 2021. URL https://arxiv.org/abs/2005.11401.

Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, Wen Zhang, and Huajun Chen. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 334–343, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713316. doi: 10.1145/3701716.3715240. URL https://doi.org/10.1145/3701716.3715240.

Teng Lin, Yuyu Luo, Honglin Zhang, Jicheng Zhang, Chunlin Liu, Kaishun Wu, and Nan Tang. MEBench: Benchmarking large language models for cross-document multi-entity question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China, November 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.emnlp-main.77/.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. Hypergraphrag: Retrieval-augmented generation via hypergraph-structured knowledge representation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 152206–152234. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/df55ee6e59f8ac4a625219e11fe9ddba-Paper-Conference.pdf.

LINHAO LUO, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ZGNWW7xZ6Q.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=oFBu7qaZpS.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5303–5315, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.322. URL https://aclanthology.org/2023.emnlp-main.322/.

Varun Magesh, Faiz Surani, Matthew Dahl, Mirac Suzgun, Christopher D. Manning, and Daniel E. Ho. Hallucination-free? assessing the reliability of leading ai legal research tools, 2024. URL https://arxiv.org/abs/2405.20362.

Costas Mavromatis, Soji Adeshina, Vassilis N. Ioannidis, Zhen Han, Qi Zhu, Ian Robinson, Bryan Thompson, Huzefa Rangwala, and George Karypis. BYOKG-RAG: Multi-strategy graph retrieval for knowledge graph question answering. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 27881–27898, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1417. URL https://aclanthology.org/2025.emnlp-main.1417/.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford infolab, 1999.

Boci Peng, Yongchao Liu, Xiaohe Bo, Jiaxin Guo, Yun Zhu, Xuanbo Fan, Chuntao Hong, and Yan Zhang. M³GQA: A multi-entity multi-hop multi-setting graph question answering benchmark. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria, July 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.acl-long.1478/.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. Raptor: Recursive abstractive processing for tree-organized retrieval, 2024. URL https://arxiv.org/abs/2401.18059.

Yaodong Su, Yixiang Fang, Yingli Zhou, Quanqing Xu, and Chuanhui Yang. Clue-rag: Towards accurate and cost-efficient graph-based rag via multi-partite graph and query-driven iterative retrieval, 2025. URL https://arxiv.org/abs/2507.08445.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel 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*, 2024. URL https://openreview.net/forum?id=nnVO1PvbTv.

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/.

Yubo Wang, Haoyang Li, Fei Teng, and Lei Chen. Agrag: Advanced graph-based retrieval-augmented generation for llms, 2025. URL https://arxiv.org/abs/2511.05549.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Xiaojun Wu, Cehao Yang, Xueyuan Lin, Chengjin Xu, Xuhui Jiang, Yuanliang Sun, Hui Xiong, Jia Li, and Jian Guo. Think-on-graph 3.0: Efficient and adaptive llm reasoning on heterogeneous graphs via multi-agent dual-evolving context retrieval, 2026. URL https://arxiv.org/abs/2509.21710.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Yingli Zhou, Yaodong Su, Youran Sun, Shu Wang, Taotao Wang, Runyuan He, Yongwei Zhang, Sicong Liang, Xilin Liu, Yuchi Ma, and Yixiang Fang. In-depth analysis of graph-based rag in a unified framework. *Proc. VLDB Endow.*, 18(13):5623–5637, January 2026. ISSN 2150-8097. doi: 10.14778/3773731.3773738. URL https://doi.org/10.14778/3773731.3773738.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.naacl-long.449. URL https://aclanthology.org/2025.naacl-long.449/.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-based approaches and functionalities in retrieval-augmented generation: A comprehensive survey. *ACM Comput. Surv.*, February 2026. ISSN 0360-0300. doi: 10.1145/3795880. URL https://doi.org/10.1145/3795880.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora, 2025. URL https://arxiv.org/abs/2510.10114.

A NOTATION

Table 4: Notation used in BubbleRAG.

Notation	Description
$G = (V, E)$	Knowledge graph with nodes V, edges E
$\mathrm{val}(\cdot)$	Per-element relevance (nodes and edges)
q	Input user query
k	Number of candidate graph elements retrieved per phrase t_i
t_i	Retrieval phrase derived from the query and preliminary chunks
$\mathcal{S} = \{S_1, \ldots, S_m\}$	Semantic anchor groups
w_i	Importance weight of group S_i
$T = (V_T, E_T)$	Candidate evidence subgraph
$H_{\mathrm{loc}} = (U, F)$	Localized entity–relation search graph
h	Neighborhood hop limit for candidate graph construction
B	Maximum number of Bubble Expansion rounds
$\mathrm{Value}(T)$	Mean per-element relevance of CEG T
$\delta(x)$	Relevance gap $1 - \mathrm{val}(x)$
$c(x)$, $\bar{c}(T)$	Traversal cost and its mean over a candidate
$\mathcal{C}$	Candidate pool
$r_{\mathrm{miss}}(T)$	Total weight of query concepts not covered by T
α	Coverage penalty coefficient
$\mathrm{Score}(T)$	Candidate ranking score in Eq. (3)
n	Number of top-ranked candidates selected
d	Max depth for multi-hop expansion
G_{final}	Evidence graph obtained by merging expanded candidates
C_{text}	Text chunks linked with evidence graph

B METHOD DETAILS

B.1 GRAPH EXTRACTION PROMPTS

Entity Extraction Prompt

```
## Goal

Your task is to extract **named entities** from the given
paragraph.
Respond ONLY with a single string containing the **valid JSON
list of entities** inside the <entities> tag.
DO NOT add any other text, reasoning, or markdown.

[Examples omitted for brevity.]

## Paragraph to Process
{text}

## Your Response
```

Triple Extraction Prompt

```
## Task Goal

Your task is to construct an **RDF (Resource Description
Framework) Knowledge Graph** from the given paragraph and list
of named entities.
**Output Requirement:** Respond with a **valid JSON list of
triples** (Subject-Predicate-Object) enclosed within a single
`<triples>` tag.
DO NOT add any other text, reasoning, or markdown.
---

## Extraction Guidelines

1.  **Entity Usage:** Each triple should contain at least one,
but **preferably two**, entities taken directly from the
provided list of named entities.
2.  **Pronoun Resolution:** Clearly resolve any **pronouns**
(e.g., "It") in the passage to their specific named entity to
ensure the clarity and independence of the resulting triples.

---

[Examples omitted for brevity.]

## Input Data

**Given Passage:**
{text}

**Given Entities:**
{entity}

## Your Response
```

B.2 Query-Time Retrieval and Answering Prompts

Preliminary Answering Prompt

```
## Goal
You are an expert Question Answering system. Your task is to
carefully reason step by step based on the provided context and
derive the most accurate and concise answer.

## Instructions
1. Read and analyze all provided contexts thoroughly.
2. Combine relevant pieces of information logically to infer
the answer.
3. Clearly explain your reasoning process in the `thought`
field.
4. If you cannot derive the answer directly from the current
context, please do not guess, set `"answer": ""`.
5. Your output **must strictly follow** the JSON format below.
6. Your answer should be as short as possible.

## Output Format
```json
```
```

Preliminary Answering Prompt (continued)

```
{
    "thought": "Briefly describe your reasoning process --
    which contexts were relevant, how you combined them, and
    why this answer was chosen (or why no answer was
    possible).",
    "answer": "The final concise answer to the user query, or
    '' if it cannot be determined."
}
```

```
[Examples omitted for brevity.]

## User Query
{user_query}

## Contexts Provided
{context}

## Your Output
```

Query Concept Refinement Prompt

```
## Context
{context}

## User Query
{user_query}

## Goal
You are an expert in information retrieval and semantic
understanding.
Your task is to:
1. Analyze the **User Query** to extract its core semantic
**nodes (entities)** and **relations
(properties/relationships)**.
2. Treat these nodes and relations as **filters**. Scan the
*entire* **Context** and find **ALL** entities or phrases that
match these filters -- including both **exact** and **partial**
matches.
3. Extract the **exact, corresponding phrases** from the
Context for all matches found.
4. If certain query concepts have **no direct match** in the
context, you must still **preserve** them (or their most
essential words) in the final keyword list to reflect the
query's intent.

This acts as a "community filter" to ensure all relevant or
potentially relevant parts of this chunk are identified -- even
if only partial matches exist.

## Guidelines

1. **Analyze Query (Nodes & Relations):**
   Break down the User Query into its semantic components
   (entities, relations, attributes).
```

Query Concept Refinement Prompt (continued)

```
   Identify both the **main entity focus** and any **dependent
   relations** (e.g., "place of birth of X").
   Each node or relation will serve as a conceptual filter.

2. **Constraint Rewriting (for underspecified relations):**
   When a node or relation lacks a clear subject or constraint,
   you must expand it to make it semantically complete.
   Always attach the most relevant entity from the query to the
   relation.
   For example:
   * "place of birth" → "place of birth of Lisbeth Palme's
   husband"
   * "occupation" → "occupation of Henry Lau"
   * "date of death" → "date of death of Olof Palme"
   * "spouse" → "spouse of Brad Newman"
   These rewritten, contextually anchored phrases should also
   appear in the final `keywords` list.
   This prevents overly broad matches and keeps the reasoning
   grounded in the query intent.

3. **Scan & Map All Matches:**
   - Locate every instance that matches or partially aligns
   with the extracted (and possibly rewritten) concepts.
   - Include **exact phrases** from the Context.
   - If a phrase indirectly implies the concept (partial
   match), still extract it.
   - If nothing matches, retain the concept's rewritten or
   original form.

4. **Extract Mapped Keywords:**
   - Always use the **exact wording** from the Context for any
   found matches.
   - For missing or partially matched concepts, keep their
   **rewritten or original form**.
   - The `keywords` list should therefore include both context
   phrases and retained query terms.
   - You should keep the mapping of all keywords.

## Output Format

Return the result strictly in the following JSON format:

```json
{
  "reason": "A step-by-step reasoning. **First, state the nodes
  and relations (concepts) extracted from the User Query.**
  Second, for *each* concept, explain how it fully or partially
  matched the context. If some concepts were not found, explain
  that they were retained or constraint-rewritten to preserve
  the query's intent.",
  "keywords": ["context_specific_keyword1",
  "context_specific_keyword2", ...]
}
```

[Examples omitted for brevity.]
```

Semantic Anchor Grouping Prompt

```
## Role & Objective

**Role:** You are an advanced RAG (Retrieval-Augmented
Generation) query processor with knowledge-graph reasoning
ability.

**Goal:**
Given a **User Query** and a flat list of **Candidate Nodes**,
you must:

1. Parse key **concepts / relations (keywords)** directly from
the query.
2. **Group** all candidate nodes according to their most
semantically relevant keyword.
3. **Assign normalized weights** to each group, based on how
likely nodes in that group are to contribute to the final
answer.
4. Output a concise JSON object containing your reasoning,
grouped nodes, and weights.

---

## Execution Logic

### 1. Parse Query Keywords
Extract the main entities, attributes, and relations from the
User Query (e.g.,
"illegitimate daughter", "caused many wars", "married twice").
Each parsed keyword defines one group, and the order of groups
follows the logical order in the query.

### 2. Group Candidate Nodes
Map each candidate node to the **single most relevant**
keyword.
- Keep group order identical to the parsed keyword order.
- Each node appears in only one group.
- The structure is:
```

```
nodes = [
[nodes for keyword_1],
[nodes for keyword_2],
...
]
```

```

### 3. Assign Group Weights
Assign floating-point weights (sum = 1.0) reflecting the
**probability** that each group leads to the final answer.
Weights depend on two interacting factors:

**Factor 1 - Keyword Centrality (TF-like importance):**
How essential the keyword is to the query's meaning.

**Factor 2 - Node Specificity (IDF-like uniqueness):**
Reward groups with rare, specific, relational nodes.
```

Semantic Anchor Grouping Prompt (continued)

````
Penalize groups dominated by generic or overly broad nodes
(e.g., “human”, “person”, “artist”), which lower answer
precision.

Approximate rule:
```

weight(keyword_i) ∝ centrality_i × (1 / genericity_i)

````

Output Format

Return a single valid JSON object (no additional text):

```json
{
  "reason": "Step-by-step explanation of how keywords were
  parsed, nodes grouped, and weights assigned. Justify higher
  weights for groups with specific, high-value nodes and lower
  weights for generic ones.",
  "keywords": ["keyword1", "keyword2", "keyword3"],
  "nodes": [
    ["node_for_keyword1", "another_node_for_keyword1"],
    ["node_for_keyword2"],
    ["node_for_keyword3"]
  ],
  "group_weights": [weight1, weight2, weight3]
}
````

[Examples omitted for brevity.]

Inputs

User Query:
{user_query}

Candidate Nodes:
{candidate_nodes}

Your response
````

Reasoning-Aware Expansion Prompt

```
## Role & Goal
You are a **Knowledge Graph Reasoning Strategist**.
Your job is **not to answer** the query but to guide a
multi-hop reasoning process by selecting the **most promising
next nodes** from the given candidates.
From the `candidates`, choose **at least {k} nodes** that best
bridge the gap between the **current context** and the **query
goal**. **More is okay**.

If you know how to answer, you can just return null.
```

Reasoning-Aware Expansion Prompt (continued)

```
---

## Reasoning Process

1. **Understand the Query**
   * Identify the **Subject Entity** (starting point).
   * Identify the **Target Information** (the property or
   relation being sought).
   * Define the **missing link** that connects them.

2. **Review Context**
   * Summarize what's already known.
   * Identify the current **semantic gap** -- what's missing to
   reach the target.

3. **Evaluate Candidates**
   * **Tier 1 (High Priority):** Directly reveal or connect to
   the missing entity or target info.
   * **Tier 2 (Medium Priority):** Strong contextual link
   expanding toward the target (e.g., family, education,
   profession).
   * **Tier 3 (Low Priority):** General or peripheral info with
   weak relevance.

Select all Tier 1 nodes and any Tier 2 nodes that meaningfully
reduce the gap.

---

## Output Format
Return only valid JSON:

```json
{
  "reason": "Explain the goal (subject + target), the current
  knowledge gap, and why the chosen candidates are the most
  promising next steps.",
  "candidates": ["candidate1", "candidate2", ...]
}
```

[Examples omitted for brevity.]

User Query
{user_query}

Context

{context}

Candidates (pick from these)

{candidates}

Your Output
```

Answer Generation Prompt

```
You are an expert Question Answering system. Your task is to
synthesize the provided context to generate a direct and
accurate response to the user query.

## User Query
{user_query}

## Contexts Provided

### Graph Context
{graph_context}

### Chunk Context
{chunk_context}

## Output Format
Your response must strictly follow this JSON format:

```json
{
    "thought": "Briefly describe your reasoning process: which
    contexts were used, how the information was combined, and
    why the final answer was chosen.",
    "answer": "The final concise answer to the user query. Keep
    it as short and factual as possible."
}
```

B.3 Hardness of Initial Evidence Selection

Theorem 1. *OISR is NP-hard and APX-hard under approximation-preserving reductions, even when all anchor groups are disjoint singletons and all element relevance values belong to* $\{0, 1\}$. *Consequently, OISR admits no polynomial-time approximation scheme unless* $\mathrm{P} = \mathrm{NP}$.

Proof. We reduce from Minimum Vertex Cover on nonempty graphs of maximum degree three, which is APX-complete (Alimonti & Kann, 2000). Let $H = (U, F)$ be such a graph, let $M = |F|$, and let $k^* \geq 1$ be its minimum vertex-cover size. Since a vertex covers at most three edges, $M \leq 3k^*$.

Construction. Create an undirected graph G with a root r, a choice vertex x_u for each $u \in U$, and a terminal t_e for each $e \in F$. Add edges (r, x_u) for all u, and (x_u, t_e) whenever u is an endpoint of e. Use singleton anchor groups $\{r\}$ and $\{t_e\}$ for every $e \in F$. Assign relevance one to r and all terminals, and zero to all choice vertices and edges. These values are compatible with the cosine-based relevance definition: assign the query vector to the positive-relevance elements and an orthogonal unit vector to all other elements. The construction is polynomial.

Feasible solutions and optimal value. Any feasible OISR subgraph T contains r and all M terminals. If it contains k choice vertices, their corresponding vertices form a cover of H: connectivity requires each t_e to connect to at least one of its endpoint choices. Thus $k \geq k^*$. There are $M+k+1$ vertices in T, and connectivity requires at least $M+k$ edges. Only the root and terminals contribute relevance, so

$$\mathrm{Value}(T) \leq \frac{M+1}{2(M+k)+1}.$$

Conversely, a vertex cover C of size k yields a feasible tree: connect each selected x_u to r and connect each terminal to exactly one selected endpoint. Its value is $(M+1)/(2(M+k)+1)$. Therefore

$$\mathrm{OPT}_{\mathrm{OISR}} = \frac{M+1}{2(M+k^*)+1}.$$
```

For any vertex-cover bound b, a cover of size at most b exists if and only if the OISR optimum is at least $(M+1)/(2(M+b)+1)$. This proves NP-hardness.

Approximation preservation. For maximization, use the convention that a ρ-approximate solution, $\rho \geq 1$, has value at least $\mathrm{OPT}_{\mathrm{OISR}}/\rho$. Extract the k selected choice vertices from such a solution. The bounds above imply

$$\frac{M+1}{2(M+k)+1} \geq \mathrm{Value}(T) \geq \frac{M+1}{\rho\,[2(M+k^*)+1]},$$

and hence

$$k \leq \rho k^* + (\rho-1)\left(M + \tfrac{1}{2}\right).$$

Using $M \leq 3k^*$ and $k^* \geq 1$ gives

$$\frac{k}{k^*} \leq 1 + \frac{9}{2}(\rho-1).$$

Thus the construction and solution extraction form an approximation-preserving reduction with a constant error factor. In particular, a $(1+2\varepsilon/9)$-approximation for OISR would give a $(1+\varepsilon)$-approximation for degree-three Vertex Cover. APX-hardness follows; a PTAS for OISR would imply a PTAS for this APX-complete problem, which is impossible unless $\mathrm{P} = \mathrm{NP}$. □

The theorem applies to the strict-coverage OISR objective in Definition 1.

B.4 Bubble Expansion: Implementation and Complexity

Algorithm 1 details candidate generation on the localized search graph H_{loc}. Let A_i denote the anchor vertices for group i, and $A = \bigcup_i A_i$. The neighborhood limit h counts KG-triple hops before the entity–relation transformation.

The candidate pool is initially empty and receives subgraphs produced by path merging. If the pool remains empty when search terminates, the initial anchors A are used as fallback starting points for reasoning-aware expansion.

Computational scope. Let $N = |U|$, $M = |F|$, and $K = |A|$. There are K independent searches, each settling at most $\min(B, N)$ vertices. Excluding visited-membership checks and candidate construction, binary-heap search work is bounded conservatively by $O(K(N+M)\log(N+M+1))$ over complete traversals. The implementation stores visited records as lists, so membership checks can add $O(KMN)$ work in the worst case. Rebuilding the meeting map each round visits at most KN records, giving $O(BKN)$ work, with additional source-signature sorting for meeting keys. Each new meeting can require tracing up to K paths of length at most N, followed by edge deduplication and sorting; the candidate count is not bounded by B. Per-source distances and predecessors require $O(KN)$ storage, in addition to queues, visited records, meeting signatures, and candidate storage.

Coverage checks scan group anchors for each candidate, while relevance aggregation scans candidate vertices. The localized graph can span the full KG in the worst case. These costs exclude embedding retrieval, LLM-based candidate preparation, and subsequent reasoning-aware expansion.

Algorithm 1: Bubble Expansion with independent anchor searches and path merging

Input : Local search graph $H_{\text{loc}} = (U, F)$, anchor groups $\{A_i\}_{i=1}^{m}$, costs c, round budget B
Output: Candidate pool $\mathcal{C}$, or initial anchors A as fallback expansion seeds

$A_i \leftarrow A_i \cap U$ for each i; $A \leftarrow \bigcup_i A_i$
$\mathcal{C} \leftarrow \emptyset$
$\mathcal{I} \leftarrow \emptyset$; $\mathcal{D} \leftarrow \emptyset$ `// processed meetings; emitted edge signatures`
for each $s \in A$ **do**
  $D_s[u] \leftarrow \infty$ for all $u \in U$; $D_s[s] \leftarrow c(s)$
  $P_s \leftarrow \emptyset$; $R_s \leftarrow \emptyset$; $Q_s \leftarrow \{(c(s), s)\}$
for $b = 1, \ldots, B$ **do**
  $active \leftarrow$ false
  for each $s \in A$ **do**
    Pop entries from Q_s until an unvisited vertex $v \notin R_s$ is found or Q_s is empty
    if *no such v is found* **then continue**
    $active \leftarrow$ true; $R_s \leftarrow R_s \cup \{v\}$
    for each $u \in \mathcal{N}(v) \setminus R_s$ **do**
      if $D_s[v] + c(u) < D_s[u]$ **then**
        $D_s[u] \leftarrow D_s[v] + c(u)$; $P_s[u] \leftarrow v$
        Push $(D_s[u], u)$ into Q_s
  if $active =$ false **then break**
  Build $J(v) = \{s : v \in R_s\}$ from all visited records
  for each *entity vertex v with* $|J(v)| > 1$ **do**
    if $(v, J(v)) \in \mathcal{I}$ **then continue**
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{(v, J(v))\}$
    Recover each path $s \rightsquigarrow v$ using P_s, for $s \in J(v)$
    if *any recovered path fails to reach its source* **then continue**
    $T \leftarrow$ union of the recovered paths; $\sigma \leftarrow$ canonical undirected edge signature of T
    if $\sigma \notin \mathcal{D}$ **then**
      Append T to $\mathcal{C}$; $\mathcal{D} \leftarrow \mathcal{D} \cup \{\sigma\}$
if $\mathcal{C} = \emptyset$ **then return** A as fallback expansion seeds
return $\mathcal{C}$

C MACHINE CONFIGURATION

The hardware configuration used in our experiments is summarized in Table 5.

Table 5: Machine configuration used in our experiments.

Component	Specification
GPU	4× NVIDIA GeForce RTX 4090 (24 GB VRAM each)
CPU	2× Intel Xeon Gold 6226R @ 2.90 GHz

D ADDITIONAL EXPERIMENTS

D.1 RETRIEVAL ROUTING

We examine the proportion of queries that proceed to graph retrieval after preliminary passage retrieval. As shown in Table 6, 42.1% of queries on 2Wiki and 17.6% on HotpotQA enter the graph stage.

Table 6: Proportion of evaluation queries entering graph retrieval with the 30B backbone.

Dataset	Graph retrieval (%)
2Wiki	42.1
HotpotQA	17.6

D.2 STATISTICAL COMPARISON

We compare BubbleRAG with HippoRAG2, the strongest baseline on average, using 10,000 paired bootstrap resamples. With the 30B backbone, BubbleRAG achieves F1 gains of 2.63–7.99 points, with positive 95% confidence intervals on all three datasets (Table 7). Across all six dataset–backbone settings, the mean gain is 3.13 F1 points, with a 95% confidence interval of $[2.14, 4.14]$.

Table 7: Paired bootstrap comparison against HippoRAG2 with the 30B backbone. Differences and intervals are in F1 points.

Dataset	F1 gain	95% confidence interval	p
HotpotQA	2.63	$[0.72, 4.55]$	.007
MuSiQue	7.99	$[5.44, 10.58]$	$< .001$
2Wiki	4.89	$[2.47, 7.35]$	$< .001$

D.3 CANDIDATE RANKING ANALYSIS

We analyze candidate pools from the 30B configuration. Supporting-document recall measures the fraction of gold supporting-document titles recovered through a candidate's source chunks. For the rank analysis in Table 8, we select queries with at least eight candidates and evaluate all ranks on the same query subset within each dataset.

Table 8 shows that earlier-ranked candidates generally recover more supporting evidence. Rank 1 achieves 56.37% recall on 2Wiki and 65.28% on HotpotQA, compared with 39.01% and 40.28% at rank 8.

D.4 CASE STUDY OF REASONING-AWARE EXPANSION

Table 9 traces how BubbleRAG retrieves a mosque's opening date from a candidate anchored at an industrial facility.

Table 8: Supporting-document recall (%) by candidate rank; rank 1 is best.

Rank	2Wiki	HotpotQA
1	56.37	65.28
2	55.41	54.17
3	50.48	47.22
4	51.27	45.83
5	45.86	40.28
6	44.11	40.28
7	42.83	44.44
8	39.01	40.28

Table 9: A case study of reasoning-aware expansion in BubbleRAG.

Item	Content
Question	When did the first mosque open in the place where Tobolar Copra plant is located?
Ground truth	September 2012
Supporting evidence	Location: "...Tobolar Copra processing plant in the Marshall Islands capital of Majuro..." Opening date: "...a small community of Ahmadiyya Muslims based in Majuro, with the first mosque opening in the capital in September 2012."
Initial retrieval	Retrieved passages describe mosques in other locations, but provide no opening date for the mosque in the plant's location.
Selected candidate	Top-ranked candidate from Bubble Expansion: Tobolar Copra processing plant Tobolar Copra processing plant located in Majuro
RAE	Step 1: Expand to Majuro, the plant's location. Step 2: Select "Ahmadiyya Muslims based in Majuro," following the question's focus on a local mosque. Step 3: Expand to Ahmadiyya Muslims. Retrieved evidence: "...the first mosque opening in the capital in September 2012."
Prediction	September 2012 (correct)

Starting from the plant, RAE follows its location relation to Majuro and selects a relevant religious-community relation over unrelated neighbors such as airports and airlines. The expanded graph provides access to the passage containing the opening date, which is absent from the initial retrieval.

D.5 Graph Characteristics and Schema Information

Graph statistics. Table 10 summarizes the constructed KGs. Approximately 70% of predicate labels occur only once.

Table 10: Knowledge graph statistics.

Dataset	Triples	Distinct predicates
2Wiki	76.9K	14.6K
HotpotQA	150.8K	36.5K
MuSiQue	157.8K	45.0K

Effect of schema information. We randomly sample 200 questions from the 2Wiki dataset. We use Claude to read the KG and generate 20 pieces of KG-related knowledge, which serve as schema notes. We vary the number of notes supplied to each method and evaluate all settings on the same sampled questions. BubbleRAG and KG-GPT use the Qwen 30B backbone, with notes supplied to graph expansion and relation retrieval, respectively. For HippoRAG2, the notes are included in the prompt.

Table 11 shows that BubbleRAG obtains 72.05 F1 without notes and 69.26–72.59 with them. KG-GPT reaches its highest F1 of 57.52 with ten notes, compared with 53.04 without notes. HippoRAG2 deviates from its no-note baseline by at most 0.50 F1 points. Performance does not improve monotonically with the number of notes.

Table 11: Effect of schema information on 200 randomly sampled 2Wiki questions. s denotes the number of supplied notes; Δ is the F1 change from the corresponding $s = 0$ run.

BubbleRAG			KG-GPT			HippoRAG2		
s	F1	Δ	s	F1	Δ	s	F1	Δ
0	72.05	0.00	0	53.04	0.00	0	62.28	0.00
2	69.26	-2.79	2	50.18	-2.86	2	62.78	$+0.50$
5	70.60	-1.45	5	54.37	$+1.33$	4	62.72	$+0.44$
8	72.59	$+0.54$	8	53.42	$+0.38$	6	62.28	0.00
10	71.26	-0.79	10	57.52	$+4.48$	8	62.22	-0.06
15	71.29	-0.76	15	54.87	$+1.83$	10	62.72	$+0.44$

D.6 Efficiency and Cost Analysis

Table 12 reports average query latency and query-time token use for 1,000 questions on 2Wiki, using the hardware in Appendix C.

Table 12: Efficiency on 2Wiki. Query tokens cover 1,000 questions; index tokens cover offline corpus preparation.

Method	Latency (s/query)	Query tokens	Index tokens
NaiveRAG	4.36	2,417,815	–
HippoRAG2	10.82	4,623,098	4,575,580
ToG	48.12	7,659,150	–
BubbleRAG	25.54	9,342,275	3,840,320

BubbleRAG takes 25.54 seconds per query, compared with 10.82 for HippoRAG2 and 48.12 for ToG. Its query-time token use is 9.34M, compared with 4.62M for HippoRAG2. Offline indexing uses 3.84M and 4.58M tokens, respectively.

Table 13 reports the stage breakdown for queries that enter graph retrieval.

Table 13: Stage shares of query-time latency and token use for queries entering graph retrieval (%).

Stage	Time (%)	Tokens (%)
Anchor grouping	43.84	39.96
CEG retrieval	23.05	0.00
RAE	26.00	45.77
Answer generation	7.11	14.27

Among queries entering graph retrieval, semantic anchor grouping accounts for the largest latency share (43.84%), while reasoning-aware expansion accounts for the largest token share (45.77%). Initial evidence subgraph retrieval uses 23.05% of latency and no LLM tokens.