PATHSMOOTHING: MITIGATING SEMANTIC DRIFT AND HUB NOISE VIA PATH EVIDENCE IN KG-BASED RAG

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graph-based retrieval-augmented generation (KG-based RAG) connects information across text passages to support answer generation. Recent studies have shown that the main limitation of KG-based RAG lies in retrieving excessive irrelevant information, researchers have proposed optimization methods centered on path filtering. However, existing methods focus on structural factors when conducting retrieval, while ignoring the semantic factors, which results in the introduction of noise nodes. To overcome this, we propose PathSmoothing that uses personalized PageRank to aggregate the structural evidence scores of entities. It enumerates context paths containing entities in the subgraph, calculates the semantic relevance of the paths, and then smooths these scores by combining the original semantic scores of these triples. PathSmoothing filters triples and retrieves their source text as evidence for answer generation. Experiments on six datasets show that PathSmoothing achieves an average win rate of 54.13% against PathRAG across four evaluation dimensions on domain-specific tasks, and increases the average answer matching score from 50.24 to 52.02 on multi-hop question answering datasets. PathSmoothing also improves LightRAG and PathRAG when applied as a plugin to them. Our code and datasets are available at supplementary material.

1 INTRODUCTION

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) retrieves evidence from external corpus to help large language models (LLMs) answer questions. To fully exploit the structured information in corpus, researchers have proposed Knowledge graph-based Retrieval-Augmented Generation (KG-based RAG), such as GraphRAG (Edge et al., 2024) and LightRAG (Guo et al., 2025). These methods first extract entities and relations from corpus to construct structured index. During retrieval, they anchor on topic entities from user query, locate relevant nodes via vector similarity matching, and perform community-based retrieval or multi-hop neighborhood expansion. Subsequently, the textual descriptions of the retrieved triples are incorporated as contextual evidence into the LLM prompt, guiding LLM to generate factually consistent responses.

Existing KG-based RAG methods retrieve evidence through neighborhood expansion, path pruning, or graph-based propagation (Guo et al., 2025; Chen et al., 2026; Gutiérrez et al., 2024). These methods use graph connections to reach information beyond the entities mentioned in the query, but structural connectivity alone does not helps answer the question in all cases. As retrieval expands, it may follow connections that lead away from the information the query requires, causing semantic drift (Six et al., 2025; Carpineto & Romano, 2012). Highly connected nodes can also introduce unrelated relations into the retrieved context (Zheng et al., 2025). These issues motivate filtering candidate relations according to their relevance to the query, rather than relying on structural signals alone.

A direct approach is to rank candidate triples by their semantic similarity to the query. However, this can overlook how triples connect to form evidence. For a question asking when the manufacturer of a missile ceased operating, the relation linking the missile to its manufacturer is necessary, even though it provides no date, i.e., low semantic similarity. A relation describing the service dates of the

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.

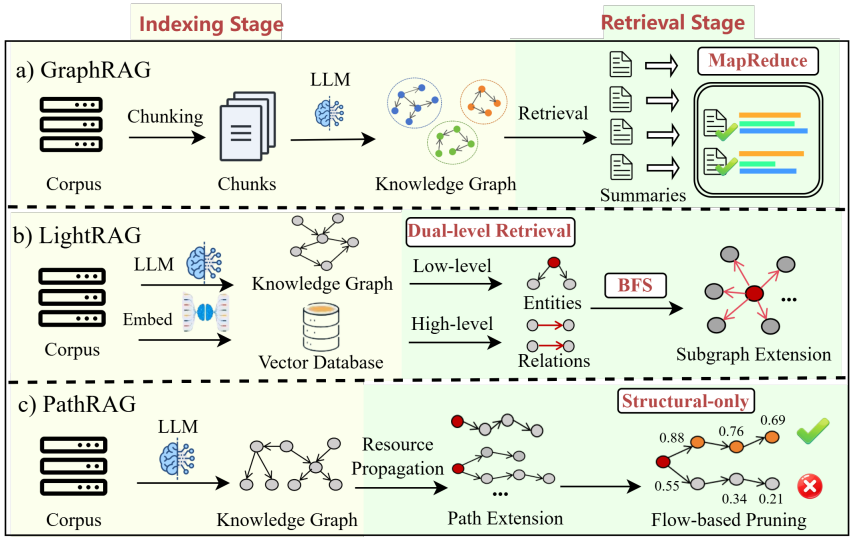

Figure 1: Representative KG-based RAG frameworks. (a) GraphRAG retrieves information using MapReduce strategy based on pre-constructed community summaries. (b) LightRAG uses low-level retrieval to locate topic entities, high-level retrieval to identify high-dimensional relations, and BFS to expand subgraph to gather answers. (c) PathRAG performs path exploration on entities and prunes paths based on flow propagation.

missile is semantically similar but does not answer the question. Distinguishing these cases requires considering how each triple connects to the surrounding relations.

Figure 1 illustrates the different KG-based RAG frameworks. Existing methods have two main limitations: **(i)** Retrieval overemphasizes topology of knowledge, while the semantic relevance between reasoning chain and query is largely ignored; **(ii)** The noisy hub nodes, such as abstract entities like "country" or "enterprise", are often overemphasized due to their structural importance, while they usually contribute little information for answering queries.

To address the above limitations, we propose **PathSmoothing**, a triple-level path smoothing denoising method, performs filtering based on the subgraph context and the paths in which candidate triples reside. A path connects a triple to surrounding entities and relations, providing context for assessing its relevance to the query. PathSmoothing combines the semantic similarity between the triple and the query with that between its containing paths and the query. This allows path evidence to raise the scores of intermediate relations or lower those of triples embedded in unrelated contexts. As a result, PathSmoothing effectively filters triples with both structural and semantic information for KG-based RAG.

PathSmoothing first locates entities mentioned in the query and expands a local subgraph to collect candidate triples. It then uses Personalized PageRank (PPR) (Page et al., 1999; Haveliwala, 2002) to assign structural weights to nodes and constructs short paths containing each triple. These weights guide path selection and weight the path semantic scores when refining triple scores. To control embedding cost, PathSmoothing limits both the total number of paths and the number retained for each triple. It filters triples using the refined scores, retrieves their source text, and trims and merges the text into evidence for answer generation.

Specifically, it incorporates both multi-hop structural and semantic information from the subgraph to alleviate mentioned issues: **(i)** Instead of evaluating solely based on structural factors, it utilizes semantic consistency at the path-level to assess whether the context structure of the triple is relevant to the query, avoiding semantic drift caused by over-reliance on single dimension. **(ii)** Structural importance serves as weighting factors in the path smoothing scoring function instead of being directly used. This ensures that even high-weight hub nodes hardly affects retrieval results if the path containing them are not semantically aligned with query. Overall, our contributions are summarized as follows:

- **Triple-level PathSmoothing Reranking:** We design a novel scoring function that integrates three key components: the original semantic score of a triple, the semantic relevance of its neighboring paths, and the structural importance weights of its nodes. This mechanism dynamically reranks triples, favoring those that better match the query structurally and semantically while suppressing noise from local optima or hub nodes.
- **PathSmoothing is Effective in Denoising**: We conduct extensive experiments demonstrates that, compared with the latest baselines, our method achieves an average winning rate of 54.13% in question answering on domain-specific datasets and improves the Answer Match Score metric by 3.54% on multi-hop question answering datasets.
- **Lightweight Plug-and-Play Denoising Module:** This module works independently of both the index construction and the graph retrieval pipeline. As a result, it can be easily plugged into existing KG-based RAG workflows without any architectural changes. Experiments show that the state-of-the-art method with PathSmoothing achieves 54.60% win rate in question answering against the version without it.

2 Related Work

In this section, we review existing KG-based RAG approaches and identify a critical problem: the failure to jointly utilize semantic and structural cues for fine-grained noise filtering, which usually degrades answer quality. This observation directly motivates our proposed triple-level PathSmoothing denoising method.

2.1 Text-based RAG

Text-based RAG (Lewis et al., 2020) alleviates the limitations of parametric models in factual accuracy and knowledge timeliness by combining external knowledge retrieval with LLM generation. (Gao et al., 2023) RAG typically uses a pre-trained dense retriever to retrieve several text passages most relevant to the query from external vector database, and then constructs new prompt with the retrieved evidences and the user query as the context for the model. This method enables LLMs to utilize dynamic external knowledge. However, when dealing with complex multi-hop questions across documents, the retrieved evidences often lack global structural information and are prone to introducing noise irrelevant to the query (Wei et al., 2026).

2.2 KG-based RAG

KG-based RAG aims to address the reasoning limitation that RAG system face when processing cross-document questions, which arise from their failure to capture the complex logical associations (Peng et al., 2025). GraphRAG (Edge et al., 2024) uses LLM to extract entities and relations from documents to build knowledge graph, and constructs hierarchical community summaries through community detection algorithm (Traag et al., 2019). For global understanding questions, it aggregates answers from each community in MapReduce strategy, using them as context for LLM to generate comprehensive answers. In addition, LightRAG (Guo et al., 2025) constructs a dual-level retrieval system, extracts keywords and high-dimension relations from question, finds anchor nodes in the graph by vector matching, and performs breadth-first search(BFS) to obtain neighboring subgraphs as evidence. These methods effectively retrieve information, but with substantial noise, resulting in poor answer quality.

2.3 Graph-Based Retrieval Optimization

KG-based RAG has a core challenge: the information retrieved is noisy (Zheng et al., 2025). To optimize this, PathRAG (Chen et al., 2026) proposes to retrieve key relation paths from the indexed graph only, reduce redundant information through flow-based pruning, and guide the LLM to generate more logical answers in the form of paths. HippoRAG (Gutiérrez et al., 2024) constructs a lightweight graph and computes node specificity via TF-IDF (Salton & Buckley, 1988), uses PPR to complete multi-hop retrieval in one round of graph computation, replacing expensive iterative retrieval. SubgraphRAG (Li et al., 2025) further integrates a lightweight MLP with a parallel triple-scoring mechanism and encodes directional structural distances for efficient subgraph retrieval, yet

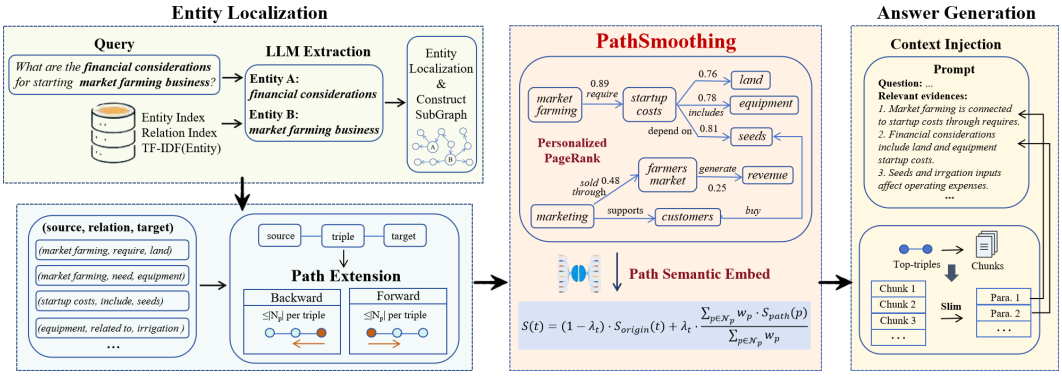

Figure 2: The overall pipeline of PathSmoothing. (a) Entity Localization: Extracts key entities and localizes their corresponding nodes in the graph to construct subgraph. (b) PathSmoothing: Enumerates candidate paths and performs smoothed scoring on triples by combining PPR structural weights and path semantic relevance. (c) Answer Generation: Compresses text chunks corresponding to high-confidence triples and injects them into context to guide LLM generation.

it likewise prioritizes structural signals over semantic relevance. However, these methods focus on structural factors while neglecting semantic information for triple-level filtering, introducing irrelevant noise into the evidence context.

3 METHODOLOGY

In this section, we propose a novel KG-based RAG denoising method called PathSmoothing, which filters the triples semantically and structurally based on path evidence. As shown in Figure 2, this method performs three main operations on the graph: entity localization, PathSmoothing filtering, and answer generation. We first use PPR to capture the entity weights in subgraph, and then perceives semantic relevance and structural importance based on the path evidence of the triples. It also adopts the cliff truncation strategy to retain the most reliable triples.

3.1 ENTITY LOCALIZATION

Given a query q, we use an LLM to extract a set of entity names $K = \{k_1, k_2, \ldots, k_n\}$. We then match these names to entities in the knowledge graph $G = (E, R)$, where E and R denote the entity and relation sets, respectively. The matched entities form a seed set E_{loc} for subsequent subgraph expansion.

We match the extracted name to graph entities by first exact string match then semantic match. For exact string matching, Let $\text{label}(e)$ denote the name of entity e. The initial seed set of entities is

$$E_{\text{loc}} = \{e \in E \mid \exists k \in K,\ \text{label}(e) = k\}. \tag{1}$$

For names without an exact match, we retrieve graph entities by semantic similarity (Karpukhin et al., 2020). Specifically, during the index construction process, we use the semantic embedding model f to encode the entities and normalize the resulting embeddings to unit length. These normalized entity vectors are stored in a pre-calculated matrix, where each entity embedding is $x_e = f(e)$ and loaded only once during the initialization stage. For each unmatched key entity $k' \in \mathcal{K}$, the system first calculates its normalized embedding $x'_k = f(k')$, and then computes the semantic similarity score via the inner product of the normalized embeddings, which is equivalent to cosine similarity: $s(k', e) = \mathbf{x}'^{\top}_k \mathbf{x}_e$. We retrieve the top-$N$ entities with the highest similarity, and then retains those entities whose scores exceed the set threshold μ. Let $\mathcal{T}'_k$ denote the set of top-N entities ranked by similarity to the unmatched key entity k'. The final set of matched entities is represented as:

$$E_{\text{loc}} \leftarrow E_{\text{loc}} \cup \bigcup_{k' \in \mathcal{K}} \{e \in T'_k \mid s(k', e) > \mu\}. \tag{2}$$

3.2 TRIPLE-LEVEL PATH SMOOTHING

We first retrieve all relational edges adjacent to $\mathcal{E}_{loc}$ from $\mathcal{R}$, and merge them with their associated head and tail entities to form the triple candidate set $\mathcal{T}_{cand}$. Meanwhile, for each entity $e \in \mathcal{E}_{loc}$, we treat it as a seed node and perform graph extension to extract local subgraph $\mathcal{G}_{sub}$ with a maximum size of n-hop. We then perform path smoothing filtering on the candidate triples in $\mathcal{G}_{sub}$.

For each candidate triple $t = (e_h, r, e_t) \in T_{\text{cand}}$, where e_h, r, and e_t denote the head entity, relation, and tail entity, respectively, we convert it into text $\text{str}(t)$. Using the same embedding model f as in entity localization, we compute the cosine similarity between the triple and the query as its initial semantic score:

$$S_{\text{origin}}(t) = \frac{f(\text{str}(t)) \cdot f(q)}{\|f(\text{str}(t))\| \, \|f(q)\|}. \tag{3}$$

We use PPR to assign structural weights to entities in G_{sub}. PPR propagates probability mass along graph edges and periodically restarts according to a specified distribution. The resulting node weights guide path selection and determine the contribution of each path to triple scoring. We compute the node weight vector $\mathbf{w_e}$ by iterating until convergence:

$$\mathbf{w_e}^{(k+1)} = (1-\alpha) M_{ij} \mathbf{w_e}^{(k)} + \alpha \mathbf{v}_{\text{init}}, \tag{4}$$

where M_{ij} is the transition probability from entity e_i to e_j, and α is the restart probability. The restart distribution combines a uniform distribution over E_{loc} with a TF-IDF distribution over all entities in the subgraph:

$$\mathbf{v}_{\text{init}} = \lambda_q \mathbf{v}_q + (1-\lambda_q)\mathbf{v}_t. \tag{5}$$

Let E_{sub} denote the entity set of G_{sub}. For $e \in E_{\text{sub}}$, we define

$$v_q(e) = \begin{cases} 1/|E_{\text{loc}}|, & e \in E_{\text{loc}}, \\ 0, & \text{otherwise}, \end{cases} \qquad v_t(e) = \frac{\text{TF-IDF}(e)}{\sum_{e' \in E_{\text{sub}}} \text{TF-IDF}(e')}, \tag{6}$$

where λ_q balances the two components. TF-IDF(e) denotes the product of term frequency and inverse document frequency for entity e. Eq.(5) satisfies the normalization condition: $\sum_{e \in \mathcal{E}_{loc}} v_{\text{init}}(e) = 1$. The topological propagation diffuses probabilistically along edges of $\mathcal{G}_{sub}$, and restarts at the initial seed with probability α, iterating until convergence. Finally, we define the smooth weight of a path as the sum of importance scores of all entities on the path, expressed as:

$$w_p = \sum_{e \in p} w_e. \tag{7}$$

To bound the cost of path embedding, we retain at most 600 paths across all candidate triples and at most 20 paths per triple. We pool the paths constructed for all triples and sort them by w_p in descending order. We scan this list and retain a path if the corresponding triple has fewer than 20 retained paths; otherwise, we skip it. Selection stops when 2000 paths have been retained or the list is exhausted. Let $N_p(t)$ denote the paths retained for triple t.

For each retained path, we arrange its entities and relations in path order to form a text representation $\text{str}(p)$. We then compute its semantic relevance to the query using the embedding model f:

$$S_{\text{path}}(p) = \frac{f(\text{str}(p)) \cdot f(q)}{\|f(\text{str}(p))\| \, \|f(q)\|}. \tag{8}$$

We refine the semantic score of each triple using its retained paths. For $N_p(t) \neq \varnothing$, we compute a weighted average of the path semantic scores and combine it with the original triple score:

$$S(t) = (1-\lambda_t) S_{\text{origin}}(t) + \lambda_t \frac{\sum_{p \in N_p(t)} w_p S_{\text{path}}(p)}{\sum_{p \in N_p(t)} w_p}, \tag{9}$$

where $\lambda_t \in [0,1]$ controls the contribution of path evidence. The original score measures the relevance of the triple in isolation, while the path scores account for its surrounding relations. Combining them allows path context to raise or lower the triple score. If no path is retained for t, we set $S(t) = S_{\text{origin}}(t)$.

We sort the candidate triples by $S(t)$ in descending order and use gaps between adjacent scores to determine how many triples to retain. Let $m = |T_{\text{cand}}|$, and let $t_{(i)}$ denote the triple at rank i, with score $S_{(i)}$. We compute

$$\Delta_i = S_{(i)} - S_{(i+1)}, \qquad i = 1, \ldots, m-1, \tag{10}$$

and locate the largest gap at $j = \arg\max_{1 \le i < m} \Delta_i$. If $\Delta_j \ge \theta_{\text{rel}} S_{(1)}$, we retain the triples up to rank j. Otherwise, we retain the top 60% of candidates. In both cases, we remove triples whose scores fall below θ_{abs}:

$$T_{\text{out}} = \begin{cases} \{t_{(i)} \mid 1 \le i \le j,\ S_{(i)} \ge \theta_{\text{abs}}\}, & \Delta_j \ge \theta_{\text{rel}} S_{(1)}, \\ \{t_{(i)} \mid 1 \le i \le \lceil 0.6m \rceil,\ S_{(i)} \ge \theta_{\text{abs}}\}, & \text{otherwise.} \end{cases} \tag{11}$$

This rule adapts the number of retained triples to the score distribution while excluding low-scoring candidates.

3.3 Answer Generation

To generate answers faithful to the knowledge sources, PathSmoothing retrieves the original corpus, uses the original text chunks associated with high-confidence triples as evidence context, and sends them to LLM for answer generation.

For each triple $t \in T_{out}$, we look up the set of text unit identifiers $ID(t)$ associated with the triple from pre-built index, and retrieve its source text chunks from the original corpus, yielding the raw evidence chunk set $C_t = \{\text{text}(u) \mid u \in ID(t)\}$, where $\text{text}(u)$ denotes the raw content of text unit u. Since directly concatenating all raw chunks would expand the context and slow down generation. To mitigate this, before assembling the final context, we split each chunk into paragraphs and retain only the paragraph that contains the evidence along with its immediate neighboring paragraphs. If a unit contains no evidence entity, we keep its first paragraph as a fallback to avoid losing all information. This trimming operation transforms C_t into a filtered evidence chunk set C_s, , denoted as $C_t \xrightarrow{\text{Trim}} C_s$. The final evidence context C is then obtained by taking the union over all triples of these filtered sets, with duplicates removed by text-unit identifiers, expressed as:

$$C = \bigcup_{t \in T_{out}} C_s, \tag{12}$$

which, together with the original question q, forms the generation prompt and sends to LLM to produce the final answer.

4 Experiments

We conduct comprehensive experiments to validate the effectiveness of PathSmoothing, specifically aiming to answer the following questions: Q1: How does our method perform compared to state-of-the-art approaches? Q2: Does each component of our method contribute to retrieval performance? Q3: How do different hyperparameters affect the performance of our method? Q4: How does our method perform when embedded as a module into other KG-based RAG methods?

4.1 Experimental Setting

Datasets. We evaluate PathSmoothing on six datasets. Agriculture, Computer Science, and Legal from UltraDomain (Qian et al., 2024) cover domain-specific question answering. SQuALITY (Wang et al., 2022) evaluates question-focused summarization of long documents. We also use HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022) to evaluate multi-hop question answering. We tune hyperparameters on 80 examples from Art, another UltraDomain dataset, and 80 examples from HotpotQA. Dataset details are provided in Appendix A.

Baselines. We compare PathSmoothing with four representative methods: GraphRAG (Edge et al., 2024) using local retrieval, LightRAG (Guo et al., 2025) with Chain-of-Thought prompting, HippoRAG (Gutiérrez et al., 2024), and PathRAG (Chen et al., 2026). These methods include the latest KG-based RAG approaches as well as retrieval denoising methods.

Evaluation Metrics. For domain-specific question answering and long-document summarization, we use DeepSeek-V4-Flash to score answers on four dimensions (Liu et al., 2023): Comprehensiveness, Diversity, Empowerment, and Accuracy (Zheng et al., 2023). The first three assess coverage, breadth of perspectives, and support for informed judgments, respectively; Accuracy measures factual consistency with the reference answer. We compare these scores to report the win rate of PathSmoothing against each baseline on each dimension.

For multi-hop question answering, we report Answer Matching Score (AMS) and LLM score. AMS assigns 100 when the generated and reference answers match exactly, 50 when either answer contains the other without an exact match, and 0 otherwise. The LLM evaluator assigns scores ranging from 0 to 100 conveying the consistency from generated answers to the ground truth. We average both scores over queries. To measure token consumption, we count prompt and completion tokens across all online LLM calls. Evaluation prompts are provided in Appendix H.

Implementation Setup. To ensure experimental consistency, we construct offline graph indices based on the LightRAG pipeline, with each text chunk segmented at a budget of 1,200 tokens. We employ Deepseek-V4-Flash as the base model for all methods (Liu et al., 2023) and use Text-Embedding-V4 for text vector embedding. During the retrieval phase, to guarantee fair treatment across different datasets, we set the key hyperparameters as follows: $\alpha = 0.15$, $\lambda_q = 0.4$, $\lambda_t = 0.6$, $\theta_{\text{abs}} = 0.1$, and $\theta_{\text{rel}} = 0.15$. Other experimental configurations are detailed in the appendix.

Table 1: Comparison between PathSmoothing and baseline across four domains. In the table, Comprehensiveness is abbreviated as Comp., Empowerment is abbreviated as Empo., and PathSmoothing is abbreviated as PS. The best results are underlined.

	Agriculture		**Computer Science**		**Legal**		**SQuALITY**		**Average**	
	GraphRAG	**PS.**	GraphRAG	**PS.**	GraphRAG	**PS.**	GraphRAG	**PS.**	Baseline	**PS.**
Comp.	32.10%	67.90%	36.67%	63.33%	31.03%	68.97%	39.33%	60.67%	34.78%	65.22%
Diversity	30.86%	69.14%	34.44%	65.56%	31.03%	68.97%	34.67%	65.33%	32.75%	67.25%
Empo.	33.33%	66.67%	41.11%	58.89%	32.18%	67.82%	38.00%	62.00%	36.16%	63.85%
Accuracy	38.27%	61.73%	35.56%	64.44%	34.48%	65.52%	36.67%	63.33%	36.25%	63.76%
	LightRAG	**PS.**	LightRAG	**PS.**	LightRAG	**PS.**	LightRAG	**PS.**	Baseline	**PS.**
Comp.	41.98%	58.02%	47.78%	52.22%	43.68%	56.32%	44.00%	56.00%	44.36%	55.64%
Diversity	45.68%	54.32%	44.44%	55.56%	45.98%	54.02%	40.67%	59.33%	44.19%	55.81%
Empo.	49.38%	50.62%	48.89%	51.11%	43.68%	56.32%	42.00%	58.00%	45.99%	54.01%
Accuracy	39.51%	60.49%	41.11%	58.89%	42.53%	57.47%	41.33%	58.67%	41.12%	58.88%
	HippoRAG	**PS.**	HippoRAG	**PS.**	HippoRAG	**PS.**	HippoRAG	**PS.**	Baseline	**PS.**
Comp.	45.68%	54.32%	48.89%	51.11%	41.38%	58.62%	43.33%	56.67%	44.82%	55.18%
Diversity	50.62%	49.38%	52.22%	47.78%	40.23%	59.77%	49.33%	50.67%	48.10%	51.90%
Empo.	44.44%	55.56%	48.89%	51.11%	41.38%	58.62%	40.67%	59.33%	43.85%	56.16%
Accuracy	46.91%	53.09%	43.33%	56.67%	47.13%	52.87%	44.00%	56.00%	45.34%	54.66%
	PathRAG	**PS.**	PathRAG	**PS.**	PathRAG	**PS.**	PathRAG	**PS.**	Baseline	**PS.**
Comp.	45.68%	54.32%	44.44%	55.56%	43.68%	56.32%	44.67%	55.33%	44.62%	55.38%
Diversity	46.91%	53.09%	45.56%	54.44%	44.83%	55.17%	44.00%	56.00%	45.33%	54.68%
Empo.	44.44%	55.56%	48.89%	51.11%	47.13%	52.87%	45.33%	54.67%	46.45%	53.55%
Accuracy	45.68%	54.32%	47.78%	52.22%	48.28%	51.72%	46.67%	53.33%	47.10%	52.90%

4.2 Retrieval Analysis (Q1)

Table 1 compares PathSmoothing with four baselines on domain-specific question answering and long-document summarization. PathSmoothing achieves an average win rate above 50% against

Table 2: Performance comparison on multi-hop question answering datasets. In the table, PathSmoothing is abbreviated as PS., The best results are highlighted in **bold**.

	HotpotQA		Musique		Average	
	AMS	LLM	AMS	LLM	AMS	LLM
GraphRAG	44.22	61.39	25.55	40.51	34.89	50.95
LightRAG	49.01	65.51	35.04	50.36	42.03	57.94
HippoRAG	57.59	71.62	38.32	54.74	47.96	63.18
PathRAG	59.24	**75.41**	41.24	56.57	50.24	65.99
PS.	**60.23**	**75.41**	**43.80**	**57.30**	**52.02**	**66.36**

Table 3: Comparison of token consumption on Legal dataset. In the table, PathSmoothing is abbreviated as PS., The best results are highlighted in **bold**.

	Prompt	Completion	Total
GraphRAG	**3452**	608	**3978**
LightRAG	31681	2566	34247
HippoRAG	8868	1080	9948
PathRAG	19403	1519	20922
PS.	21687	**263**	21950

each baseline across datasets and evaluation dimensions. Against PathRAG, the best baseline, its average win rate is 54.13% and wins on all four dimensions. The only exceptions are that HippoRAG performs better on Diversity in Agriculture and Computer Science, with win rates of 50.62% and 52.22%, respectively. Overall, PathSmoothing improves answer quality across various datasets and evaluation dimensions.

Table 2 reports the results on multi-hop question answering. PathSmoothing achieves the highest AMS on both datasets, improving over PathRAG by 0.99 points on HotpotQA and 2.56 points on MuSiQue. The average AMS increases from 50.24 to 52.02. For LLM scores, PathSmoothing matches PathRAG on HotpotQA and improves the score from 56.57 to 57.30 on MuSiQue, raising the average from 65.99 to 66.36. These results show larger improvements on MuSiQue and in answer matching than in LLM-based evaluation.

Table 3 compares token consumption across all online LLM calls on the Legal dataset. PathSmoothing uses 21,950 tokens, 4.9% more than PathRAG and 35.9% fewer than LightRAG. Compared with PathRAG, it uses more prompt tokens (21,687 vs. 19,403) but fewer completion tokens (263 vs. 1,519). Together with the answer quality results, this shows that PathSmoothing improves answers over PathRAG with 4.9% more online LLM token consumption.

4.3 Ablation Studies (Q2)

We compare PathSmoothing with two variants to assess how path semantics and structural weighting contribute to triple scoring. **Triple-only Scoring** uses $S_{\text{origin}}(t)$ directly, without incorporating path semantic scores. **Uniform Path Weighting** retains the same selected paths but assigns them equal weights when averaging their semantic scores, i.e., eliminating structural information. We evaluate both variants on Agriculture, Computer Science, Legal, and MuSiQue. Figure 3 reports the win rate of the full method against each variant on the first three datasets and AMS on MuSiQue.

The full method outperforms both variants. On the three domain-specific datasets, its win rates against Uniform Path Weighting are higher than those against Triple-only Scoring, with the largest difference on Legal. On MuSiQue, removing path semantics reduces AMS from 43.80 to 39.21, while using uniform path weights reduces it to 40.92. These results show that both path semantics and PPR-based weighting effectively improve triple scoring.

4.4 Hyperparameter Study (Q3)

We examine the subgraph expansion depth and the smoothing weight λ_t on Legal and MuSiQue. We report the mean LLM score across the four evaluation dimensions for Legal and AMS for MuSiQue. Figure 4 shows the results.

For subgraph expansion, we compare depths of one to four hops with a random candidate selection setting labeled 0-hop. Legal reaches its highest score of 75.57 at two hops, with no further improvement at greater depths. MuSiQue improves across the tested depths and reaches 45.99 at four hops. These results indicate that a broader candidate set benefits MuSiQue, whereas expanding beyond two hops offers little benefit on Legal.

We vary λ_t from 0.2 to 1.0 in increments of 0.2. Legal reaches its highest score of 76.64 at $\lambda_t = 0.6$, while MuSiQue reaches 45.26 at $\lambda_t = 0.8$. At $\lambda_t = 1.0$, the scores decrease to 70.67 and 44.45,

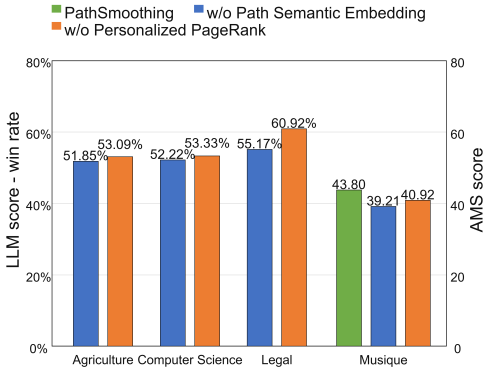

Figure 3: Ablation study on key modules of PathSmoothing under four different datasets.

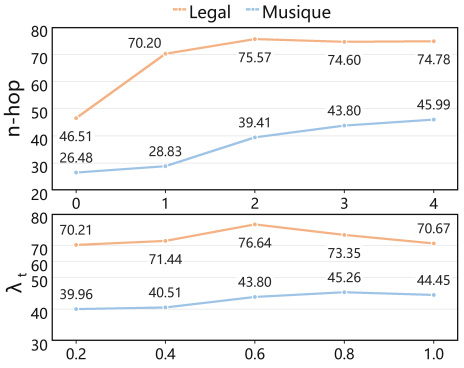

Figure 4: The impact of different n-hop and λ_t on retrieval performance.

Table 4: The performance of PathSmoothing as a plugin on Legal and Musique platforms. In the table, Comprehensiveness is abbreviated as Comp., Empowerment is abbreviated as Empo., and PathSmoothing is abbreviated as PS., The best results are underlined.

	Legal				**Musique**
	Comp.	Diversity	Empo.	Accuracy	AMS
LightRAG	43.68%	45.98%	43.68%	42.53%	35.04
LightRAG+PS.	56.32%	54.02%	56.32%	57.47%	38.30
PathRAG	43.68%	49.43%	42.53%	45.98%	41.24
PathRAG+PS.	56.32%	50.57%	57.47%	54.02%	46.04

respectively. Combining triple and path semantic scores therefore performs better than relying solely on path semantics for triples with retained paths.

4.5 Transferable Denoising Modules (Q4)

We integrate PathSmoothing into LightRAG and PathRAG after subgraph retrieval. Each baseline constructs its own subgraph, and PathSmoothing further filters the triples within it. We compare the augmented systems with their original versions on Legal and MuSiQue, using win rates across four evaluation dimensions for Legal and AMS for MuSiQue.

As shown in Table 4, adding PathSmoothing improves both baselines. On Legal, LightRAG + PS. and PathRAG + PS. achieve average win rates of 56.03% and 54.60%, respectively, against their original versions. On MuSiQue, PathSmoothing increases AMS from 35.04 to 38.30 for LightRAG and from 41.24 to 46.04 for PathRAG, yielding gains of 3.26 and 4.80 points. These results show that PathSmoothing can improve answer quality by filtering subgraphs retrieved by existing KG-based RAG methods.

5 Conclusion

We present PathSmoothing, a KG-based RAG method that uses path context to refine triple relevance scores. By combining triple and path semantics with structural weights, it selects triples and retrieves their source text to support answer generation. Experiments on six datasets show improvements in answer quality, while ablation studies demonstrate the contributions of path semantics and structural weighting. PathSmoothing, as a plug-in to existing methods, also improves LightRAG and PathRAG through additional filtering of their retrieved subgraphs.

AI USE STATEMENT

We utilize AI coding tools for the modular implementation, debugging, testing, and iteration of certain modules of PathSmoothing. In addition, we use generative AI to provide suggestions for optimizing paragraph layout and correct grammatical errors and awkward phrasing arising from non-native English writing. AI does not provide ideas or claims for our research. We ensure that all research contributions are original and that all experiments are reliable.

ETHICS STATEMENT

The datasets used in our experiments are all widely used public datasets. The baseline methods used in the experiments are all open source project code repository. Our research does not involve privacy or sensitive data, and we strictly comply with the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide the main code for reproducing the experiments, including dataset preprocessing, method implementation, hyperparameter configuration, and extension plugin integration. All datasets used are sourced from open-source public databases.

REFERENCES

Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *Acm Computing Surveys (CSUR)*, 44(1):1–50, 2012.

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. In *Proceedings of the AAAI conference on artificial intelligence*, volume 40, pp. 30183–30191, 2026.

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.

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 J 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.

Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pp. 517–526, 2002.

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

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.

Mufei Li, Siqi Miao, and Pan Li. Simple is effective: The roles of graphs and large language models in knowledge-graph-based retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 6061–6089, 2025.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 2511–2522, 2023.

Larry Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web: Stanford infolab. *Navigation, findability and the usage of cultural heritage on the web: an exploratory study*, pp. 1508503, 1999.

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.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*, 1, 2024.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Valentin Six, Evan Dufraisse, and Gaël de Chalendar. The structure-content trade-off in knowledge graph retrieval. *arXiv preprint arXiv:2506.13380*, 2025.

Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):5233, 2019.

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.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R Bowman. Squality: Building a long-document summarization dataset the hard way. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1139–1156, 2022.

Jiaqi Wei, Hao Zhou, Xiang Zhang, Di Zhang, Zijie Qiu, Noah Wei, Jinzhe Li, Wanli Ouyang, and Siqi Sun. Retrieval is not enough: Enhancing rag through test-time critique and optimization. *Advances in Neural Information Processing Systems*, 38:21484–21520, 2026.

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 *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Yilun Zheng, Dan Yang, Jie Li, Lin Shang, Lihui Chen, Jiahao Xu, and Sitao Luan. Less is more: denoising knowledge graphs for retrieval augmented generation. *arXiv preprint arXiv:2510.14271*, 2025.

A Dataset Details

Our experiments employ six datasets, including three datasets from UltraDomain benchmark (Qian et al., 2024)—Agriculture, Computer Science, and Legal, one long-document summarization dataset—SQuALITY and two multi-hop question answering datasets—HotpotQA and Musique. Brief descriptions of the key characteristics of the above datasets are provided as follows:

(a) Agriculture: The dataset is derived from agricultural domain texts, covering topics such as beekeeping, small-scale farm management, grazing management, and urban agriculture. The questions involve domain-specific practical knowledge, e.g., "How to extract and process honey" and "What is the definition of grazing intensity." This dataset is primarily designed to evaluate models' retrieval capability over specialized agricultural subdomains.

(b) Computer Science: This dataset is characterized by breadth reasoning, emphasizing coverage of a wide range of concepts and their interrelations. Its graph structure encompasses diverse entities spanning from fundamental data structures and algorithms to distributed systems and cloud services. The questions require models to integrate conceptual associations across different subdomains, e.g., understanding the relationships among Kafka and multiple programming languages.

(c) Legal: The dataset consists of legal contract texts and focuses on corporate legal practice, covering corporate restructuring, legal agreements, compliance management, and corporate governance, with an emphasis on professional texts in the legal and financial domains. It requires models to comprehend the precise and nuanced language of legal instruments.

(d) SQuALITY (Wang et al., 2022): This is a question-oriented long-document summarization dataset, in which the source documents consist of English science fiction short stories. Each document is paired with one overarching question and several targeted questions, and multiple independent reference summaries are manually authored for each question, emphasizing faithful compression of and precise response to the entire narrative content.

(e) HotpotQA (Yang et al., 2018): This is a Wikipedia-based multi-hop question answering dataset, in which questions are designed to require multi-step reasoning across multiple supporting documents in order to be answered. Sentence-level supporting fact annotations are provided for each question, endowing the model's reasoning path with interpretability.

(f) Musique (Trivedi et al., 2022): This is a multi-hop question answering dataset constructed in a bottom-up manner via a single-hop question composition strategy. Its core characteristics include chain-mandated reasoning, a leak-free design across reasoning steps, and the incorporation of unanswerable contrastive question pairs to further increase the level of challenge.

B Baseline Details

Our experiments use four baseline methods: GraphRAG, LightRAG, HippoRAG and PathRAG. The key information of these baseline methods is briefly described as follows:

(a) GraphRAG (Edge et al., 2024): GraphRAG constructs a knowledge graph by extracting entities and relationships from documents, followed by community detection and hierarchical summarization. Given a query, it first identifies the relevant communities and then performs local retrieval using community-level summaries and graph-structured information. In our experiments, we adopt the Local Search variant of GraphRAG, which is more closely aligned with our retrieval setting.

(b) LightRAG (Guo et al., 2025): LightRAG introduces a low-cost and lightweight graph-enhanced retrieval mechanism over the knowledge graph, jointly leveraging entities and relationships for retrieval. After locating query-relevant entities or relationships in the graph, it performs local BFS based graph traversal and combines the retrieved graph information with vector based text retrieval. The resulting structured knowledge and textual evidence are then provided to the large language model for answer generation. In addition, we employ LightRAG with its default configuration in combination with Chain-of-Thought prompting.

(c) HippoRAG (Gutiérrez et al., 2024): HippoRAG draws inspiration from the human hippocampal memory mechanism and organizes knowledge into a semantically associated memory network. It employs PPR to propagate query activation from initially relevant memory nodes to their neighboring nodes, thereby discovering potentially relevant knowledge through graph-based associative propagation. This mechanism enables the retrieval of indirectly connected information and facilitates multi-hop knowledge retrieval.

(d) PathRAG (Chen et al., 2026): PathRAG treats relational paths in the knowledge graph as the fundamental units for retrieval and reasoning. It searches for query-relevant paths in the graph and applies flow based filtering and pruning strategy to select high-quality candidate paths. By organizing scattered but valuable multi-hop information into semantically coherent evidence chains, PathRAG facilitates more effective retrieval and reasoning over interconnected knowledge.

C CASE STUDY

To evaluate the effectiveness of PathSmoothing on multi-hop questions, we conducted a comparative analysis of the retrieval process against PathRAG on Musique dataset. As shown in Table 5, PathRAG overly relied on local path weight scores, resulting in incorrect retrieval results. Despite extracting entities such as `SS.11` and `NORD AVIATION`, it incorrectly adopts a high scoring path (e.g., the path passing through `AA.20` pointing to 1960) as the core reasoning basis, resulting in the incorrect answer of 1960. In contrast, by focusing on facts that are both semantically and structurally relevant via its path smoothing mechanism, PathSmoothing accurately extracts the path `NORD AVIATION merged with Sud Aviation in 1970` (with a path relevance score of 0.66211), successfully deriving the correct answer of 1970.

Table 5: Case study on the retrieval process of PathSmoothing and PathRAG.

Query	In which year did the company that made SS.11 end?
Truth	1970
PathRAG	**Entity extraction:** `SS.11/SS.12 SERIES` (0.7025), `SS.11` (0.5493), `NORD AVIATION` (0.5138), and `AGM-22` (0.5127). **Path scoring:** `bfs_weighted_paths` (threshold 0.3, $\alpha = 0.8$) scores each path by its mean edge weight; 308 scored paths were deduplicated to 165 and ranked, giving 1.0000, 0.9000, and 0.8133 for 1-, 2-, and 3-hop paths. **Key path:** `SS.11` $\xrightarrow{\text{weapon designation, manufacturing}}$ `NORD AVIATION` $\xrightarrow{\text{missile development, defense}}$ `AA.20` $\xrightarrow{\text{service life, year}}$ `1960`, `NORD AVIATION` $\xrightarrow{\text{corporate merger}}$ `SNIAS` **Answer:** 1960.
PathSmoothing	**Entity extraction:** `NORD AVIATION` (0.03514), `SS.11` (0.03003), `FRANCE` (0.00916), and `SNIAS` (0.00693). **Key path:** `SS.11` $\xrightarrow{\text{made by}}$ `NORD AVIATION` $\xrightarrow{\text{merged with Sud Aviation in 1970}}$ `SUD AVIATION`, with a path relevance score of 0.66211. **Path smoothing:** The path evidence promotes the triple "Nord Aviation merged with Sud Aviation in 1970," connecting the missile manufacturer to the year in which the company ceased to exist independently. **Answer:** 1970.

D IMPACT OF DIFFERENT MODELS ON RETRIEVAL PERFORMANCE

Due to the potential impact of LLMs with different capability levels on experiments or evaluation, we replace Deepseek-V4-Flash with GPT-5.5 and GPT-5.6-sol as the reasoning generation model and the evaluation model, respectively. As shown in Table 6, when the model used is GPT-5.5, the average win rates of PathSmoothing on the two datasets are 51.55% and 53.73%, respectively; when the model used is GPT-5.6-sol, the average win rates are 53.09% and 55.46%, respectively. Based on the above experiments, the stronger the model capability, the better the retrieval performance of the method.

Table 6: Performance comparison against PathRAG on the Legal and Agriculture datasets using different models as judges. In the table, Comprehensiveness is abbreviated as Comp., PathSmoothing is abbreviated as PS., The best results are underlined.

	Agriculture		Legal		Average	
GPT-5.5	PathRAG	PS.	PathRAG	PS.	PathRAG	PS.
Comp.	49.38%	50.62%	47.13%	52.87%	48.26%	51.75%
Diversity	46.91%	53.09%	48.28%	51.72%	47.60%	52.41%
Empowerment	49.38%	50.62%	43.68%	56.32%	46.53%	53.47%
Accuracy	46.91%	53.09%	45.98%	54.02%	46.45%	53.56%
GPT-5.6-sol	PathRAG	PS.	PathRAG	PS.	PathRAG	PS.
Comp.	48.15%	51.85%	44.83%	55.17%	46.49%	53.51%
Diversity	44.44%	55.56%	42.53%	57.47%	43.49%	56.52%
Empowerment	46.91%	53.09%	44.83%	55.17%	45.87%	54.13%
Accuracy	48.15%	51.85%	45.98%	54.02%	47.07%	52.94%

E ALGORITHM PSEUDOCODE

Algorithm 1: Triple Path Smoothing

Input: Query q, Knowledge graph $G = (V, E)$
Output: Answer y
$K \leftarrow \texttt{ExtractEntities}(q)$
$\mathcal{E}_{loc} \leftarrow \texttt{Match}(K, V)$
$G_{sub} \leftarrow \texttt{BFS}(G, \mathcal{E}_{loc}, G, \text{Max hops} = n - hop)$
$\mathcal{T}_{cand} \leftarrow \texttt{ExtractTriples}(G_{sub})$
$w_e \leftarrow \texttt{PPR}(G_{sub}, \text{Query weight} = \lambda_q, \text{TF}-\text{IDF weight} = 1 - \lambda_q)$
foreach $t \in \mathcal{T}_{cand}$ **do**
  $s_{origin}(t) \leftarrow \texttt{CosineSimilarity}(\text{Embed}(q), \text{Embed}(t))$
  $p \leftarrow \texttt{SelectPath}(t, G_{sub})$
  $N_p(t) \leftarrow \texttt{AddPath}(p)$
  $s_{path}(p) \leftarrow \texttt{CosineSimilarity}(\text{Embed}(q), \text{Embed}(p))$
end
foreach $t \in \mathcal{T}_{cand}$ **do**
  foreach $p \in N_p(t)$ **do**
    $w_p \leftarrow \sum_{e \in p} w_e$
  end
  $s(t) \leftarrow (1 - \lambda_t) \cdot s_{origin}(t) + \lambda_t \cdot \frac{\sum_{p \in N_p(t)} w_p \cdot s_{path}(p)}{\sum_{p \in N_p{}^{?}(t)} w_p}$
end
$\mathcal{T}_{out} \leftarrow \texttt{Rank}(\mathcal{T}_{cand}, s(t), \text{Cliff detection} = \text{true}, \theta_{\text{abs}}, \theta_{\text{rel}})$
$C_t \leftarrow \texttt{Retrieve}(\mathcal{T}_{out}, G_{sub})$
$C_s \leftarrow \texttt{SemanticTrim}(C_t, \text{Evidence entities})$
$y \leftarrow \texttt{Generate}(q, C_s)$
return y

Our algorithm first extracts and matches key entities from the query, constructs a local subgraph via BFS, and evaluates entity importance using PPR that integrates query information and TF-IDF weights. On this basis, the algorithm enumerates paths containing candidate triples, computes the semantic similarity of each path and the sum of PPR weights of all entities along the path, and fuses them with the original triple score to form a smoothed score. Finally, relevant triples are filtered according to cliff detection and a threshold, and evidence text is retrieved and compressed to generate the final answer.

F HYPERPARAMETER SETTINGS

We tuned the hyperparameters on Art and HotpotQA datasets. The specific parameter settings and brief explanations in the codebase are presented in Table 7.

Table 7: Hyperparameter settings and their brief explanations.

Hyperparameter	**Value/Configuration**	**Brief Explanation**
subgraph_max_hops	3	Maximum number of hops in the subgraph.
max_triples	2000	Maximum number of candidate triples.
max_back_ext	20	Maximum number of backward path extensions.
max_fwd_ext	20	Maximum number of forward path extensions.
max_total_path_embeddings	600	Budget for path embeddings.
lambda	0.50	Weight for path smoothing.
alpha	0.15	Restart probability for PPR.
max_iter	100	Maximum number of PPR iterations.
lambda_query	0.40	Weight for query seeds.
entity_vec_top_k	5	Top-K for entity recall.
entity_vec_score	0.35	Similarity threshold for entities.
matched_max_entities	15	Maximum number of matched entities.
gap_abs_threshold	0.10	Absolute threshold for score cliff detection.
gap_rel_ratio	0.15	Relative ratio threshold for score cliff detection.
fallback_top_ratio	0.60	Retention ratio for fallback triples.
chunk_vec_top_k	15	Top-K for text chunk recall.
context_slim	1	Enable paragraph compression.
answer_max_tokens	1024	Maximum output length.
model	Deepseek-V4-Flash	Large Language Model.
embedding model	Text-Embedding-V4	Embedding model.

G DISCUSSION ON PATHSMOOTHING

Despite the retrieval advantages our method exhibits over baselines on the datasets, we acknowledge its limitation in retrieval time cost. In contrast to other methods, our method introduces semantic vector embeddings of paths on the local subgraph, which endows triples with semantic awareness of their neighborhoods. However, as the graph scale grows, it becomes necessary to compute semantic vector embeddings for multiple paths containing each triple, resulting in increased computational time cost.

H Overview of the Prompt Templates in PathSmoothing

In Table 8, Table 9, Table 10, Table 11 and Table 12, the LLM prompt words for the key steps are presented respectively. Table 11 and Table 12 follow the prompt for building the index as specified by LightRAG.

Table 8: Answer generation based on evidence.

Prompt	You are a knowledgeable assistant. Based on the following text passages, provide a concise, accurate, factually consistent answer to the question. Relevant Text Passages: {context_text} Question: {prompt}
Function	After the knowledge retrieval is completed, the LLM is required to generate a concise, accurate and factually consistent response solely based on the user's query and the provided evidence.

Table 9: Document entity and relation extraction.

Prompt	Given a text document that is potentially relevant to this activity and a list of entity types, identify ALL entities of those types from the text. Be thorough - extract every entity you can find, even if it seems minor, obscure, or only appears once. 1. Identify all entities. For each identified entity, extract: - entity_name - entity_type - entity_description 2. From the entities identified in step 1, identify all pairs of source_entity and target_entity that are clearly related to each other. For each pair, extract: - source_entity - target_entity - relationship_description - relationship_strength 3. Extract as many entities and relationships as possible. 4. Return output in English as a single list of all entities and relationships identified. Entity_types: {entity_types} Text: {input_text} Output:
Function	Extract entities, entity types, entity descriptions, and relationships between entity pairs from raw documents.

Table 10: Entity description merging.

Prompt	You are a helpful assistant responsible for generating a comprehensive summary of the data provided below. Given one or two entities, and a list of descriptions, all related to the same entity or group of entities. Please concatenate all of these into a single, comprehensive description. Make sure to include information collected from all the descriptions. If the provided descriptions are contradictory, please resolve the contradictions and provide a single, coherent summary. Make sure it is written in third person, and include the entity names so we have the full context. Entities: {entity_name} Description List: {description_list} Output:
Function	Aggregate multiple descriptions of the same entity or entity group into one comprehensive, coherent summary.

Table 11: Answer scoring for domain-specific datasets.

Prompt	**—Role—** You are an expert tasked with scoring an answer to a question based on four criteria: **Comprehensiveness**, **Diversity**, **Empowerment**, and **Accuracy**. - **Comprehensiveness**: How much detail does the answer provide to cover all aspects and details of the question? - **Diversity**: How varied and rich is the answer in providing different perspectives and insights on the question? - **Empowerment**: How well does the answer help the reader understand and make informed judgments about the topic? - **Accuracy**: How factually consistent is the system answer with the reference (real) answer? For each criterion, score the answer on a scale from 0 to 100 (inclusive), where higher scores indicate the answer better satisfies the criterion. Scores must be precise to two decimal places. **Important scoring rules — read carefully:** - Only evaluate the substance and quality of the response content. The length, verbosity or word count of the response shall not affect any scoring. - The first three ratings should be entirely based on the extent to which the response addresses the question, the richness of the viewpoints provided, and its ability to help readers understand and make reasonable judgments. These three ratings should be assessed solely based on the match between the answer itself and the question; references (real) answers shall not influence the scoring results. Here is the question: {query} Here is the answer: {answer} Here is the reference (real) answer: {reference} Do NOT provide any explanations, reasoning, or additional text — output ONLY the JSON object below.
Function	For domain-specific datasets, we use four indicators to evaluate the system's answers against the reference answers: comprehensiveness, diversity, weighting, and accuracy. Strictly follow the scoring rules, giving priority to content quality and factual consistency. The output must be JSON format and should not include any reasoning or explanations.

Table 12: Answer scoring for multi-hop datasets.

Prompt	You are a strict but fair semantic judge. Determine whether the "System Answer" contains the same meaning as the "Standard Answer" for the given question. Scoring Rules (3-level): - score = 1: The System Answer conveys the same core information as the Standard Answer, even if phrased differently or with extra context. - score = 0.5: The System Answer does not contain the exact Standard Answer, but mentions related/similar content that is relevant to the question. - score = 0: The System Answer is completely different from or contradicts the Standard Answer. Strictly evaluate whether the system answer and the standard answer convey the same meaning and are consistent with factual accuracy. Question: {question_text} System Answer: {ans_text} Standard Answer: {answer_text} Return ONLY a number: 0, 0.5, or 1. No other text, no JSON, no explanation.
Function	For the multi-hop question answering dataset, the system's answers are compared with the standard answers to assess semantic consistency and factual accuracy. The output is given in three grades: 1: The core meaning is consistent, even if the wording or additional background is different; 0.5: The answer does not exactly match the standard answer, but mentions content related or similar to the question; 0: The answer is completely different from or contradictory to the standard answer.