Training-Free Residual-Guided Retrieval for Multi-hop KGQA with Single-Step LLM Inference

Keito Sei[1], Sunil Kumar Maurya[2], Yohei Kobashi[2], Yutaka Matsuo[2]

[1]Tokyo City University, [2]The University of Tokyo

Correspondence: kemachi0314@gmail.com

Abstract

Graph-based retrieval is effective for knowledge graph question answering (KGQA), but retrieving the correct reasoning path remains challenging in multi-hop settings because the retrieval objective changes as reasoning progresses. Existing methods often address this issue with repeated LLM-based planning, which improves adaptivity but increases inference cost. We present a training-free residual-guided retrieval framework that updates the retrieval signal according to the current reasoning path in embedding space. The method performs multi-hop graph retrieval without repeated hop-wise LLM calls and invokes the LLM only once for final answer generation. Experiments on WebQSP and CWQ show that our method achieves superior Hit rates with only a single LLM call. Additional comparisons with supervised graph-based KGQA methods show that the proposed method remains effective without task-specific training data. These results suggest that residual-guided retrieval is an efficient alternative for multi-hop graph-based question answering.

1 Introduction

Retrieval-augmented generation (RAG) extends large language models (LLMs) with external knowledge beyond parametric memory. In graph-based RAG, knowledge graphs (KGs) are well suited for complex question answering because they explicitly represent entities, relations, and multi-hop connections (Mavromatis and Karypis, 2025; Mavromatis et al., 2025). However, retrieving the correct reasoning path remains difficult for multi-hop questions, where the information needed at each step changes as reasoning progresses (Mavromatis and Karypis, 2022; Luo et al., 2024; Sun et al., 2024; Sui et al., 2025; Dong et al., 2025; Lin et al., 2025).

A key challenge is that the next useful expansion depends on both the question and the current reasoning path. As illustrated in Figure 1, static retrieval uses a fixed query throughout graph traversal and can miss changing constraints across hops. Iterative LLM-guided methods address this by adapting the search at each hop, but they require multiple LLM calls. Our goal is to obtain similar adaptivity with substantially lower inference cost by updating the retrieval signal in embedding space.

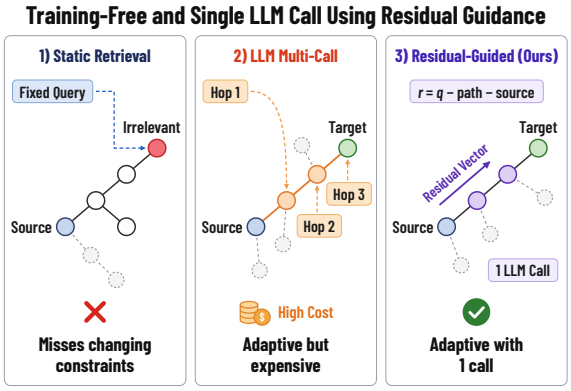


Figure 1: Motivation for residual-guided retrieval. Static retrieval may miss changing constraints during multi-hop reasoning, while LLM multi-call methods adapt retrieval with high inference cost. Our method updates the retrieval signal in embedding space, enabling adaptive retrieval with a single LLM call.

Existing methods mainly address this in two ways. Iterative LLM-guided approaches adapt retrieval during traversal (Sun et al., 2024; Dong et al., 2025; Sui et al., 2025; Lin et al., 2025; Mavromatis et al., 2025), but require repeated LLM calls. Learned graph retrievers such as GNN-RAG (Mavromatis and Karypis, 2025) and SubgraphRAG (Li et al., 2025) improve retrieval quality, but depend on task-specific supervision.

We propose a training-free residual-guided retrieval framework that updates the retrieval objective directly in embedding space. At each hop, the method computes a residual vector from the question embedding and the current reasoning-state em-

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 430–442

November 6-10, 2026 ©2026 Association for Computational Linguistics

bedding to guide the next retrieval step, and calls the LLM only once for final answer generation. We evaluate the method on WebQSP and CWQ (Yih et al., 2016; Talmor and Berant, 2018) using Hit rate.

In summary, our contributions are as follows:

- We propose a training-free residual-guided retrieval framework for multi-hop KGQA that dynamically updates the retrieval objective based on the current reasoning path in embedding space.
- We provide a simple alternative to iterative LLM-guided graph traversal without repeated hop-wise LLM calls or task-specific retriever training.
- Experiments on WebQSP and CWQ show that our method achieves superior Hit rates compared with LLM-guided baselines, while requiring only a single LLM call.

2 Related Work

Recent LLM-based approaches to KGQA and structured-data reasoning can be broadly divided into two lines.

The first line uses LLMs to iteratively guide graph exploration and reasoning. StructGPT (Jiang et al., 2023) is an early tool-augmented framework for structured-data reasoning. In KGQA, RoG (Luo et al., 2024), ToG (Sun et al., 2024), FiDeLiS (Sui et al., 2025), EffiQA (Dong et al., 2025), and RJE (Lin et al., 2025) improve multi-hop reasoning by using LLMs to plan traversal, verify intermediate evidence, or revise exploration when evidence is insufficient. BYOKG-RAG (Mavromatis et al., 2025) is also related, as it iteratively refines graph queries and reasoning artifacts for zero-shot KGQA over custom KGs. While these methods adapt retrieval to the evolving reasoning state, they typically rely on repeated LLM calls during search, leading to high inference cost.

The second line improves KG retrieval with learned retrievers. GNN-RAG (Mavromatis and Karypis, 2025) trains a graph neural retriever to identify question-relevant nodes and extract evidence for downstream reasoning, while SubgraphRAG (Li et al., 2025) learns a lightweight retriever for compact subgraph selection. GGR (Liu et al., 2025) further uses GNN guidance over contextualized subgraphs to improve entity and relation scoring while preserving globally important reasoning paths. Moving toward end-to-end optimization, D-RAG (Gao et al., 2025) jointly optimizes subgraph retrieval and answer generation through differentiable subgraph selection and prompt construction. Although effective, these approaches depend on task-specific training, learned retrieval modules, or supervised optimization.

Our method is positioned between these two lines of work. It aims to obtain adaptive multi-hop retrieval without repeated hop-wise LLM calls, while avoiding task-specific retriever training. Instead of using the LLM to re-plan at each hop, we update the retrieval objective directly in embedding space with a residual guidance vector.

3 Method

Given a question and its topic entities, our goal is to retrieve reasoning paths from a knowledge graph that support answer generation. Our key idea is state-dependent retrieval: instead of reusing the question embedding at every hop, we compute a residual vector from the current reasoning path and use it to guide relation and entity selection.

3.1 Residual-Guided Search

Let x be the input question, $v_q \in \mathbb{R}^d$ its embedding, z a retrieval state, p_z the current reasoning path associated with z, $v_{p_z} \in \mathbb{R}^d$ its path embedding, and $v_{\text{src}} \in \mathbb{R}^d$ a source-entity embedding. We define the residual guidance vector as

$$r_z = \text{norm}\big(v_q - \alpha_p \text{norm}(v_{p_z}) - \alpha_{\text{src}} \text{norm}(v_{\text{src}})\big), \quad (1)$$

where $\text{norm}(\cdot)$ denotes ℓ_2 normalization. The weights α_p and α_{src} control how strongly the current path and source entity are subtracted. Intuitively, r_z represents the question semantics not yet covered by the current path.

Path state. We do not serialize p_z into text. Instead, we embed each relation from its preprocessed text and each entity from its name. If hop t follows relation r_t from the current entity to a neighbor u_t, we update v_{p_z} as

$$v_{p_z}^{(t)} = \text{norm}\Big(v_{p_z}^{(t-1)} + w_{\text{rel}} v_{r_t} + w_{\text{ent}} v_{u_t}\Big), \quad (2)$$

where v_{r_t} and v_{u_t} are their embeddings, $w_{\text{rel}} = 0.6$, and $w_{\text{ent}} = 0.4$.

We use this residual directly for graph search. For an outgoing relation r from the current entity, we compute

$$s_{\text{rel}}(r \mid z) = \lambda_{\text{res}} \cos(v_r, r_z) + \lambda_q \cos(v_r, v_q) + \lambda_{\text{type}} \cos(v_r, v_{\text{type}}), \quad (3)$$

where v_r is the relation embedding and v_{type} embeds a question-type anchor built from the first word of the question (Appendix A). If no anchor can be built, e.g., for imperative questions, we drop the λ_{type} term. For a candidate edge (z, r, u),

$$\begin{aligned} s_{\text{ent}}(u \mid z) &= \lambda_{\text{res}} \cos(v_u, r_z) + \lambda_q \cos(v_u, v_q) \\ &\quad + \lambda_{\text{type}} \cos(v_u, v_{\text{type}}), \\ s_{\text{edge}}(z, r, u) &= s_{\text{rel}}(r \mid z) + \beta_{\text{edge}} s_{\text{ent}}(u \mid z). \end{aligned} \quad (4)$$

Both relation selection and edge expansion are guided by the residual signal. The coefficient β_{edge} controls the contribution of the reached entity to the final edge score, so expansion is not based only on relation similarity. At each hop, the retriever keeps top-$3K$ coarse relations, where $3K$ is three times the relation budget K, reranks them using top-K_n neighbor evidence, and extends paths with the resulting edge scores. After each expansion, we update the path state with Eq. (2) and recompute r_z. This shifts retrieval toward the remaining constraint.

Unlike LLM-guided step-by-step search, our method updates the search target through deterministic vector arithmetic, without hop-wise LLM calls or retriever training. The overall algorithm is provided in Appendix A.

4 Experimental Setup

Datasets and metric. We evaluate our method on the official test sets of WebQSP (Yih et al., 2016) and CWQ (Talmor and Berant, 2018), which contain 1,639 and 3,531 questions, respectively. WebQSP mainly includes shorter questions, while CWQ includes more compositional multi-hop questions. Following prior KGQA work such as GNN-RAG (Mavromatis and Karypis, 2025), we use Hit rate as the evaluation metric. Appendix F also reports macro-F1 and an LLM-as-a-Judge evaluation. Dataset and evaluation details are provided in Appendices B and C.

Models and settings. Unless otherwise noted, we use Qwen3-Embedding-4B (Zhang et al., 2025) for retrieval and Qwen3-8B (Yang et al., 2025) for answer generation. The retriever requires no training. Hyperparameters (Appendix E) were selected once on the WebQSP development set. They were reused unchanged for CWQ, the other embedding models, and the noisy-graph experiments. Implementation, hyperparameter, and prompt details are provided in Appendices D–G. Our code is publicly available.[1]

Method	WebQSP		CWQ	
	Hit (%)	#LLM Calls	Hit (%)	#LLM Calls
Vanilla LLM	62.0	1	42.1	1
CoT LLM	72.1	1	53.0	1
StructGPT (Jiang et al., 2023)	72.6	4	54.2	4
ToG (Sun et al., 2024)	82.6	11.2	67.6	14.3
FiDeLiS (Sui et al., 2025)	<u>84.1</u>	10.7	<u>71.4</u>	15.2
Proposed	**86.0**	1	**73.0**	1

Table 1: Comparison with representative zero/few-shot KGQA baselines on WebQSP and CWQ in terms of Hit rate and LLM calls. Results for competing methods are taken from the corresponding papers.

Method	Hit		#Train Data
	WebQSP	CWQ	
RoG	85.7	62.6	30.5K
SubgraphRAG	**89.4**	<u>68.6</u>	27.6K
GNN-RAG	85.7	66.8	27.6K
Proposed	<u>86.0</u>	**73.0**	0

Table 2: Comparison with representative KGQA methods in terms of Hit rate and training requirements.

5 Experimental Results

Table 1 compares the proposed method with representative baselines for KGQA, including prompt-based LLM methods and LLM-guided graph retrieval methods. We report Hit rate and the mean number of LLM calls on WebQSP and CWQ.

Our method achieves 86.0% Hit rate on WebQSP and 73.0% on CWQ with just a single LLM call. Compared with reported FiDeLiS, our method improves CWQ performance by 1.6 points and reduces the number of LLM calls from 15.2 to 1.

Table 2 compares the proposed method with representative graph-based KGQA methods regarding accuracy and training requirements, highlighting its effectiveness without task-specific supervision.

Supervised methods such as RoG, SubgraphRAG, and GNN-RAG use 27.6K–30.5K training examples. In contrast, our method uses no training data, achieves the best Hit rate on CWQ among the compared methods, and maintains comparable

[1] github.com/SeiKdesu/Training-Free-Residual-Guided-Retrieval-for-Multi-hop-KGQA

Setting	WebQSP Hit	CWQ Hit
Component / weight ablation ($\lambda_{\text{res}}/\lambda_q/\lambda_{\text{type}}$)		
Proposed (0.3/0.5/0.2)	**86.0**	73.0
w/o residual ($\lambda_{\text{res}} = \lambda_{\text{type}} = 0$)	84.1	68.5
w/o type anchor ($\lambda_{\text{type}} = 0$)	85.5	**73.2**
Q-focused (0.2/0.6/0.2)	85.1	72.8
Res.–Q balanced (0.4/0.4/0.2)	84.4	72.9
Embedding model		
Qwen3-Embedding-4B (default)	**86.0**	73.0
Qwen3-Embedding-8B	85.1	**75.1**
Jina embeddings	85.7	72.8

Table 3: Ablations and embedding-model comparison. Parentheses give the residual/question/type weights in Eqs. (3) and (4).

performance on WebQSP.

Ablations. Table 3 reports component/weight ablations and embedding-model comparisons. Setting $\lambda_{\text{res}} = \lambda_{\text{type}} = 0$ lowers Hit by 1.9 points on WebQSP and 4.5 on CWQ. Removing only the type anchor changes Hit by -0.5 on WebQSP and $+0.2$ on CWQ. Removing the rules from the answer-generation prompt changes CWQ Hit by -0.2 (Appendix G). Thus, neither the anchor nor the prompt rules explain the main improvement. The alternative weightings stay within 0.2 points of the default on CWQ but lose up to 1.6 points on WebQSP. With BGE-large-en-v1.5 and unchanged hyperparameters, the default weights still add 4.5 points on CWQ over $\lambda_{\text{res}} = \lambda_{\text{type}} = 0$ (66.2 $\rightarrow$ 70.7; Appendix F.1). Additional hyperparameter sensitivity results are provided in Appendix E.1.

Residual case study. Table 4 shows one representative CWQ example illustrating how the residual signal changes the next-hop retrieval target. Both variants first identify John F. Kennedy as the person educated at Edward Devotion School. Without the residual signal, retrieval continues along education-related paths because the original question still contains the school context. The proposed method suppresses already covered information and shifts the next hop toward the remaining constraint, "brothers," retrieving the sibling relation and the correct answers. Additional path-level residual-ablation cases are provided in Appendix K.

Qualitative comparison with LLM-guided search. Both our method and LLM-guided iterative search adapt retrieval across hops, but they adapt different objects. LLM-guided methods such as ToG (Sun et al., 2024) repeatedly ask an LLM to select, rank, or score candidate relations/entities

Item	Retrieved evidence
Question	Who are brothers of the individual educated at Edward Devotion School?
w/o residual	Edward Devotion School $\rightarrow$ John F. Kennedy $\rightarrow$ *education* $\rightarrow$ Harvard University, Noble and Greenough School
Proposed	Edward Devotion School $\rightarrow$ John F. Kennedy $\rightarrow$ *sibling_s* $\rightarrow$ Joseph P. Kennedy, Jr., Ted Kennedy, Robert F. Kennedy

Table 4: Representative residual-guided retrieval case from CWQ. Red: correct, blue: incorrect.

Method	Reasoning path / retrieved evidence
ToG	Lou Seal $\rightarrow$ *sports.mascot.team* $\rightarrow$ San Francisco Giants $\rightarrow$ *championships* $\rightarrow$ World Series event candidates.
Proposed	Lou Seal $\rightarrow$ *sports.mascot.team* $\rightarrow$ San Francisco Giants $\rightarrow$ *sports.sports_team.championships* $\rightarrow$ 2014 World Series; also retrieves related championship paths such as 2012 World Series and 2010 World Series.

Table 5: Diagnostic example on the Lou Seal question. The table contrasts ToG's LLM-guided traversal with the evidence retrieved by our residual-guided method. With a compact quantized LLM, ToG reaches the relevant championship relation but does not explicitly preserve the final grounded reasoning path. In contrast, our method retrieves the gold reasoning path through explicit KG search.

generated from the current node. Thus, the update is a model-dependent local decision over candidates. In contrast, our method updates the retrieval objective itself through deterministic vector arithmetic: it subtracts the semantics already covered by the current path from the question representation, producing a residual vector that represents the remaining information need.

Table 5 shows a diagnostic example on the Lou Seal question using ToG with Qwen3-8B-AWQ. ToG reaches the relevant team and championship relation through hop-wise LLM-guided traversal, but the final grounded reasoning path is not explicitly preserved. This suggests that when the controller LLM is compact or quantized, hop-wise LLM-guided search can become sensitive to local candidate-selection errors or may rely on parametric knowledge at the final answering stage. Our residual-guided method retrieves the corresponding KG evidence path, including the gold championship event, without additional LLM calls during traversal. The key distinction is whether multi-hop reasoning is handled as an implicit search process inside the LLM, whose behavior depends on model capability, or as an explicit retrieval process over the KG, whose intermediate evidence can be directly inspected and controlled. Further robustness,

error, and case-study analyses are provided in Appendices I–K.

6 Conclusion

We proposed a training-free residual-guided retrieval framework for complex KGQA. By subtracting the semantics already covered by the current reasoning path from the original question representation, the residual vector shifts retrieval toward the remaining unresolved constraint. This enables adaptive multi-hop retrieval as an explicit KG search process, without repeated hop-wise LLM planning.

On WebQSP and CWQ, the method achieves higher Hit rates than iterative LLM-guided baselines with a single LLM call and no task-specific training data. These results suggest that residual-guided retrieval is an efficient alternative to such methods for multi-hop KGQA.

7 Limitations

Our work has three main limitations. First, the final LLM over-generates answers. Macro precision is 46.5 on WebQSP and 30.4 on CWQ, far below recall (75.1 and 69.0). Macro-F1 (47.2 and 36.1) is also lower than that reported by GNN-RAG (71.3 and 59.4; Appendix F.2). Our prompt asks for all possible answers, as in GNN-RAG. Unlike GNN-RAG, however, we do not fine-tune the generator, so it does not learn dataset-specific answer selection or cardinality. Second, the current retrieval process uses a fixed maximum hop length and does not automatically determine when sufficient evidence has been collected. Third, we evaluate only on WebQSP and CWQ, two English benchmarks over Freebase. We have not tested KGs with different graph structures or domains.

Future work should address these issues by filtering path-supported candidates without extra LLM calls or training, developing adaptive stopping criteria, and evaluating on other KGs. Hop-dependent weighting is another direction. Our weights are fixed across hops. Neither alternative weighting becomes clearly better on deeper CWQ questions (Appendix E.2), but we have not tested hop-dependent weights directly.

References

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. 2026. jina-embeddings-v5-text: Task-targeted embedding distillation. *Preprint*, arXiv:2602.15547.

Zixuan Dong, Baoyun Peng, Yufei Wang, Jia Fu, Xiaodong Wang, Xin Zhou, Yongxue Shan, Kangchen Zhu, and Weiguo Chen. 2025. EffiQA: Efficient question-answering with strategic multi-model collaboration on knowledge graphs. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7180–7194, Abu Dhabi, UAE. Association for Computational Linguistics.

Guangze Gao, Zixuan Li, Chunfeng Yuan, Jiawei Li, Wu Jianzhuo, Yuehao Zhang, Xiaolong Jin, Bing Li, and Weiming Hu. 2025. D-RAG: Differentiable retrieval-augmented generation for knowledge graph question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 35398–35417, Suzhou, China. Association for Computational Linguistics.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251, Singapore. Association for Computational Linguistics.

Mufei Li, Siqi Miao, and Pan Li. 2025. 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, pages 6061–6089.

Can Lin, Zhengwang Jiang, Ling Zheng, Qi Zhao, Yuhang Zhang, Qi Song, and Wangqiu Zhou. 2025. RJE: A retrieval-judgment-exploration framework for efficient knowledge graph question answering with LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17277–17294, Suzhou, China. Association for Computational Linguistics.

Haochen Liu, Song Wang, and Jundong Li. 2025. Knowledge graph retrieval-augmented generation via GNN-guided prompting. In *Second Conference on Language Modeling*.

Linhao Luo, Yuan-Fang Li, Reza Haffari, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, volume 2024, pages 14400–14423.

Costas Mavromatis, Soji Adeshina, Vassilis N. Ioannidis, Zhen Han, Qi Zhu, Ian Robinson, Bryan Thompson, Huzefa Rangwala, and George Karypis. 2025. BYOKG-RAG: Multi-strategy graph retrieval for knowledge graph question answering. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27881–27898, Suzhou, China. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2025. GNN-RAG: Graph neural retrieval for efficient large language model reasoning on knowledge graphs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 16682–16699, Vienna, Austria. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.8-Max: A new bar for coding and cowork.

Yuan Sui, Yufei He, Nian Liu, Xiaoxin He, Kun Wang, and Bryan Hooi. 2025. FiDeLiS: Faithful reasoning in large language models for knowledge graph question answering. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 8315–8330, Vienna, Austria. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, volume 2024, pages 3868–3898.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Xingyu Tan, Xiaoyang Wang, Qing Liu, Xiwei Xu, Xin Yuan, and Wenjie Zhang. 2025. Paths-over-graph: Knowledge graph empowered large language model reasoning. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, page 3505–3522, New York, NY, USA. Association for Computing Machinery.

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, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

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. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *Preprint*, arXiv:2506.05176.

A Full Retrieval Algorithm

Algorithm 1 summarizes the retrieval procedure. For each frontier state, the retriever computes the residual vector and uses it to score relations, neighboring entities, and candidate edges.

Why linear residual subtraction is appropriate for KGQA. The residual update in Eq. (1) is used as a retrieval-query update, not as an exact symbolic decomposition of the question. Multi-hop KGQA questions usually contain several constraints, and once a partial path p_z satisfies some of them, repeatedly using the original question embedding v_q can keep retrieval biased toward already covered semantics. Subtracting the normalized path embedding $\mathrm{norm}(v_{p_z})$ reduces the direction explained by the current evidence path, while subtracting $\mathrm{norm}(v_{\mathrm{src}})$ weakens the dominance of the topic entity. The residual vector r_z therefore serves as a soft query for the remaining unresolved constraint.

This linear update is effective in our setting because relation and entity ranking are based on cosine similarity in the same embedding space. Changing the query vector directly changes the next-hop ranking direction: after identifying an educated person, for example, the retriever should move away from education-related evidence and toward the remaining relation such as siblings or teams. The weights α_p and α_{src} avoid fully removing these semantics, and the final normalization reduces scale variation before cosine-based scoring.

Answer-type anchor construction. The answer-type anchor is rule-based and derived only from the surface form of the input question. It is used as a lightweight answer-type prior, not as the main retrieval signal. We lowercase the question, remove punctuation, tokenize it by whitespace, and inspect the first token. For questions beginning with *who*, *where*, or *when*, we use fixed anchor texts: "who person", "where location", and "when time", respectively. For *what* and *which* questions, we extract at most three early tokens after skipping generic function words such as auxiliaries, articles,

name, *type*, and *kind*; extraction stops at common predicate, relative-pronoun, or prepositional words such as *that*, *which*, *who*, *is*, *are*, *did*, *in*, *on*, *for*, *from*, and *with*. The final anchor concatenates the first word with the extracted phrase, e.g., "what city" or "which university". If no valid anchor is found, no anchor term is used.

The anchor text is embedded with the same encoder as questions, relations, and entities, and is used as an auxiliary similarity term in both relation and neighbor-entity scoring. In the default setting, the anchor weight is 0.2, while the residual and question weights are 0.3 and 0.5, respectively. Thus, the anchor weakly biases retrieval toward the expected answer type, while the main retrieval behavior remains determined by the residual and question embeddings. For example, "Who wrote Hamlet?" yields "who person", "What city is the capital of France?" yields "what city", and "How many people live in Tokyo?" yields no anchor.

Fallback without an anchor. Some questions, e.g., "Identify the brothers of the individual educated at Edward Devotion School," do not start with a supported interrogative word. They receive no anchor, so we drop the λ_{type} term from Eqs. (3) and (4) and use only the residual and question terms. Removing the anchor for all questions changes Hit from 86.0 to 85.5 on WebQSP and from 73.0 to 73.2 on CWQ (Table 3). The anchor is a heuristic that relies on English interrogative words. Multilingual and implicitly typed questions remain a limitation.

B Dataset Statistics

Table 6 summarizes the statistics of the two benchmark datasets used in our experiments.

Table 7 shows the distribution of reasoning-hop lengths in each dataset. WebQSP is dominated by one- and two-hop questions, whereas CWQ includes a substantial proportion of questions requiring three or more hops.

C Evaluation Details

We evaluate all methods using a string-match-based Hit metric. A prediction is counted as correct if it contains at least one ground-truth answer string. When multiple gold answers are available, the prediction is counted as correct if any one of them appears in the output. All reported results are based on this Hit metric.

Algorithm 1: Overall Algorithm

Input: Question x, topic entities E_0, KG $\mathcal{G}$, max hops T
Output: Final answer a

Encode x into question embedding v_q;
Initialize frontier $\mathcal{F}_0$ from topic entities E_0;
Initialize retrieved paths $\mathcal{P} \leftarrow \emptyset$;
for $t = 1$ **to** T **do**
  Initialize next frontier $\mathcal{F}_t \leftarrow \emptyset$;
  foreach *state* $z \in \mathcal{F}_{t-1}$ **do**
    Compute the residual r_z with Eq. (1);
    Score relations with Eq. (3) and keep top-$3K$;
    foreach *coarse relation* r **do**
      Score neighboring entities with Eq. (4);
      Rerank r using its top-K_n neighbor evidence;
    Keep top-K relations by neighbor-aware score;
    Score candidate edges with Eq. (4);
    Extend the path, update its state with Eq. (2), and add new states to $\mathcal{F}_t$ and $\mathcal{P}$;
  Keep top-scoring states in $\mathcal{F}_t$;
Rank paths in $\mathcal{P}$, select top paths $\mathcal{P}^*$, build a prompt, and generate answer a with one LLM call;
return a

F1. For each question, we compute precision, recall, and F1 between the predicted and gold answer sets. Macro scores average them over questions, as in GNN-RAG (Mavromatis and Karypis, 2025). Micro-F1 pools true positives, false positives, and false negatives over all questions. Results are given in Appendix F.2.

LLM-as-a-Judge. Predictions that fail strict matching are re-evaluated with GPT-4o-mini as a judge (Appendix F.4).

D Implementation Details

For dense retrieval, we use Qwen3-Embedding-4B to encode questions, relations, entities, and the answer-type anchor into a shared embedding space. Relations are encoded from their preprocessed text, and entities from their names. Paths are not serialized into text, and the path state follows Eq. (2).

Dataset	Train	Dev	Test	avg. $\lvert\mathcal{V}_q\rvert$	coverage (%)
WebQSP	2,848	250	1,639	1,429.8	94.9
CWQ	27,639	3,519	3,531	1,305.8	79.3

Table 6: Dataset statistics. avg. $\lvert\mathcal{V}_q\rvert$ denotes the average number of entities in the extracted subgraph, and coverage denotes the percentage of questions for which the extracted subgraph contains at least one gold answer entity.

Dataset	1-hop	2-hop	≥3-hop
WebQSP	65.49%	34.51%	0.00%
CWQ	40.91%	38.34%	20.75%

Table 7: Distribution of reasoning-hop lengths in WebQSP and CWQ.

The `head -[relation]-> tail` format in Appendix G is used only in the answer-generation prompt. All embeddings are normalized before similarity computation, and cosine similarity is used for retrieval scoring. For final answer generation, we use Qwen3-8B. The retrieval process is training-free and does not require task-specific supervision.

At each hop, the method computes a residual guidance vector from the question embedding, the current reasoning-state embedding, and the source-entity embedding. Relations are first ranked using residual-guided relation scoring, and the top-$3K$ relations are retained as coarse candidates, where $3K$ denotes three times the final edge budget K. The retriever then scores neighboring entities of these relations and reranks candidate edges with a neighbor-aware edge score. The final top-K edges are used to extend reasoning paths.

E Hyperparameter Details

Table 8 summarizes the main hyperparameters used in the proposed retrieval framework. In Eq. (1), α_p and α_{src} control the subtraction strength of the current path and source entity. In Eqs. (3) and (4), λ_{res}, λ_q, and λ_{type} balance residual, question, and type-anchor similarities for relation and entity scoring. The coarse multiplier is set to 3, meaning that the retriever first keeps top-$3K$ relations before neighbor-aware reranking. The coefficient β_{edge} controls the contribution of the entity score to the final edge score in Eq. (4). In Eq. (2), w_{rel} and w_{ent} weight the relation and entity embeddings added to the path state.

Symbol	Role	Value
α_p	Path subtraction weight	0.60
α_{src}	Source-entity subtraction weight	0.60
λ_{res}	Residual weight for scoring	0.30
λ_q	Question weight for scoring	0.50
λ_{type}	Type-anchor weight for scoring	0.20
β_{edge}	Entity bonus for edge scoring	0.20
w_{rel}	Relation weight in the path state	0.60
w_{ent}	Entity weight in the path state	0.40

Table 8: Main hyperparameters used in the proposed retrieval framework.

Selection protocol. We selected all values in Table 8 once on the WebQSP development set (250 questions). We reused them unchanged for CWQ, for the other embedding models (Jina (Akram et al., 2026), Qwen3-Embedding-8B, and BGE-large-en-v1.5), and for the noisy-graph experiments (Appendix I). Table 3 varies the λ weights, and Table 9 varies α_p, α_{src}, and β_{edge}.

E.1 Hyperparameter Sensitivity

We further examine the sensitivity of the proposed method to two retrieval-specific hyperparameters on WebQSP: the residual-update weights α_p and α_{src} in Eq. (1), and the edge entity bonus β_{edge} in Eq. (4). For the residual update, we set both α_p and α_{src} to 1.0. For the edge entity bonus, we vary β_{edge} from the default value 0.20 to 0.40 and 0.60.

The results show that performance changes only moderately under these hyperparameter variations. Setting the residual-update weights to $\alpha_p = \alpha_{\text{src}} = 1.00$ decreases WebQSP Hit from 86.0% to 85.4%. Similarly, increasing the edge entity bonus from the default $\beta_{\text{edge}} = 0.20$ to 0.40 and 0.60 yields 85.7% and 85.0%, respectively. Although all of these changes slightly reduce performance compared with the default setting, the drops remain within 1.0 point. This suggests that the default setting gives the best result among the tested values, while the proposed method is not highly sensitive to moderate changes in these hyperparameters.

E.2 Hit by Reasoning Hops

Table 10 reports CWQ Hit by reasoning-hop count for the two alternative weightings in Table 3. We take the hop count of a question from its annotated SPARQL query. It is the number of edges on the shortest path from a topic entity to the answer. Constraint edges, such as the additional conditions of conjunction questions, and intermediate mediator nodes are not counted. The resulting distribution therefore differs from the one in Table 7. The two

Setting	Value	WebQSP Hit
Default	$\alpha_p = 0.60$ $\alpha_{\text{src}} = 0.60$ $\beta_{\text{edge}} = 0.20$	**86.0**
Update weight	$\alpha_p = 1.00$ $\alpha_{\text{src}} = 1.00$	85.4
Edge bonus	$\beta_{\text{edge}} = 0.40$	85.7
Edge bonus	$\beta_{\text{edge}} = 0.60$	85.0

Table 9: Hyperparameter sensitivity analysis on WebQSP.

settings differ by at most 0.7 points in any group and by 0.4 points on $\geq$3-hop questions. Neither becomes clearly better as reasoning depth increases. We did not test hop-dependent weights.

Hops	#Questions	Q-focused	Res.–Q balanced
1	1,539	75.2	75.0
2	1,442	70.0	70.7
$\geq$3	550	73.3	72.9
All	3,531	72.8	72.9

Table 10: CWQ Hit (%) by reasoning-hop count for the Q-focused (0.2/0.6/0.2) and Res.–Q balanced (0.4/0.4/0.2) weightings. The default weighting was not evaluated per hop group.

F Additional Analyses

F.1 Residual Effect Across Embedding Models

Table 11 compares the default weights with $\lambda_{\text{res}} = \lambda_{\text{type}} = 0$ on CWQ for two encoders. Both use the hyperparameters in Table 8.

Embedding model	w/o residual	Proposed	Δ
Qwen3-Embedding-4B	68.5	73.0	+4.5
BGE-large-en-v1.5	66.2	70.7	+4.5

Table 11: CWQ Hit (%) with and without the residual term ($\lambda_{\text{res}} = \lambda_{\text{type}} = 0$) for two embedding models. WebQSP was not evaluated with BGE-large-en-v1.5.

F.2 Answer-Set Precision, Recall, and F1

Table 12 reports our macro precision, recall, and F1, our micro-F1, and the macro-F1 reported by GNN-RAG (Mavromatis and Karypis, 2025). Precision is far below recall on both datasets. Our prompt (Appendix G) asks for all possible answers, and the generator is not fine-tuned. The output therefore often includes non-gold candidates. For example, the output in Table 19 includes the intermediate entity David Beckham along with the gold answer. GNN-RAG, in contrast, fine-tunes its generator on the training set and thus learns dataset-specific answer selection and cardinality. Appendix F.3 tests a larger generator with a revised prompt.

	Macro			Micro	GNN-RAG
Dataset	P	R	F1	F1	Macro-F1
WebQSP	46.5	75.1	47.2	31.2	71.3
CWQ	30.4	69.0	36.1	27.3	59.4

Table 12: Answer-set quality of the proposed method (%) and the macro-F1 reported by GNN-RAG.

F.3 Larger Generator and Revised Prompt

We test whether a larger generator and a revised prompt reduce over-generation. We replace Qwen3-8B with Qwen3.8-27B (Qwen Team, 2026), using the UD-Q5_K_XL GGUF quantization released by Unsloth. We also make two changes to the prompt in Box G. First, we remove "all" from the instruction to return all possible answers, so that the model lists fewer candidates. Second, we add a sixth rule. The original setting sometimes shortened answers, for instance answering "2014" when the gold answer was "2014 World Series". The new rule therefore asks the model to copy answer strings from the paths as written and not to output machine IDs such as `m.xxxx` or `g.xxxx`.

Table 13 shows the results. Macro-F1 rises by 26.2 points on WebQSP and 26.5 on CWQ, and exceeds the macro-F1 reported by GNN-RAG on both datasets. Hit drops by 2.0 points on WebQSP and 2.7 on CWQ. CWQ Hit stays above that of GNN-RAG, while WebQSP Hit falls below it. The method still uses a single LLM call and no task-specific training data. We changed the generator and the prompt together, so this experiment does not separate their effects.

	WebQSP		CWQ	
Setting	Hit	F1	Hit	F1
GNN-RAG	85.7	71.3	66.8	59.4
Qwen3-8B (default)	86.0	47.2	73.0	36.1
Qwen3.8-27B + new prompt	84.0	73.4	70.3	62.6

Table 13: Hit and macro-F1 (%) of the default setting (Qwen3-8B), Qwen3.8-27B with the revised prompt, and GNN-RAG. GNN-RAG values are those reported by Mavromatis and Karypis (2025).

F.4 LLM-as-a-Judge Evaluation

Strict Hit misses correct predictions that differ in surface form, such as aliases or formatting variants. We therefore re-evaluate strict misses with GPT-4o-mini as a judge. The judge receives the question, the prediction, and one gold answer, and returns YES or NO. A YES counts the question as correct. Strict matches count as correct without judging. The judge prompt is shown below.

Judge prompt

Question: {question}
Predicted Answer: {predicted}
Ground Truth: {single_ground_truth}

Answer YES only if the predicted answer and the ground truth refer to the same entity, value, or date as an answer to the question.
Accept aliases, paraphrases, abbreviations, and equivalent expressions.
Do not accept answers that are only related, or that refer to a broader or narrower entity than the ground truth (e.g. a country when the ground truth is a state or city in it).
Do not accept text that merely contains or mentions the ground truth without being the answer itself.

Respond with ONLY: YES or NO

Table 14 shows the results. The gains confirm that some strict misses are correct. However, they are small, so surface-form differences explain only part of the remaining errors. We keep strict Hit as the main metric for reproducibility and comparability with prior work.

Dataset	Strict Hit	Judge Hit	Δ
WebQSP	86.0	88.5	+2.5
CWQ	73.0	75.7	+2.7

Table 14: Strict Hit and LLM-as-a-Judge Hit (%). Only strict misses are judged.

G Prompt

This appendix provides the prompt template used for final answer generation. In the proposed method, the LLM is invoked only once after retrieval, and the prompt is constructed from the question, topic entities, and the retrieved reasoning paths. The model is instructed to answer the question using only the provided evidence and to return only the final answer.

Prompt Template The prompt used in our experiments is shown in Box G. Each retrieved path is textualized as a sequence of triples in the form `head -[relation]-> tail`. Paths are grouped by hop length and enumerated within each group. If multiple triples belong to the same path, they are concatenated with the separator `|`.

Prompt template used for final answer generation

You are a knowledge-graph QA system.
Based on the knowledge paths, please answer the given question.
Please keep the answer as simple as possible and return all possible answers as a list.
==========
Rules:
1. TEMPORAL FIRST: If the question contains first/earliest/last/latest, compare candidates by time.
2. Answer the full question; prefer the final answer over intermediate nodes.
3. Match the exact role/entity type requested.
4. Use dates, roles, and qualifiers in paths when they affect the answer.
5. Return all the possible answers.
==========
Output:
Multiple answers separated by |

Question: {question}
Topic Entities: {topic_entities}

=== Knowledge Paths ===

— 1-hop paths —
{1-hop paths}

— 2-hop paths —
{2-hop paths}

— 3-hop paths —
{3-hop paths}

=== Output Rules ===
Return only the final answer.
Do not output JSON.
Do not output explanations.
Do not output reasoning.
If multiple distinct answers are returned, separate them with |.
If the paths do not uniquely identify a single answer, return all path-supported candidates rather than forcing one answer.

The *w/o prompt rules* setting removes the rule-based instructions from this prompt and keeps all other settings unchanged. It changes CWQ Hit from 73.0 to 72.8.

H LLM Usage Cost

Table 15 provides a detailed comparison of token-level LLM usage cost on CWQ and WebQSP. For ToG, PoG, and RJE, we report the efficiency statis-

tics from RJE (Lin et al., 2025), which compares these iterative LLM-guided methods in terms of average LLM calls, input tokens, output tokens, and runtime per question. For our method, we report the statistics measured under our own implementation. In addition to requiring only a single LLM call, the proposed method uses substantially fewer input and output tokens than iterative LLM-guided baselines.

Dataset	Method	Input Tokens	Output Tokens
CWQ	ToG	8182.9	1486.4
	PoG	7803.0	353.2
	RJE	5769.1	247.2
	Ours	**1731.1**	**61.9**
WebQSP	ToG	6031.2	987.7
	PoG	5234.8	282.9
	RJE	2763.3	148.7
	Ours	**1747.9**	**75.2**

Table 15: Comparison of token-level LLM usage cost on CWQ and WebQSP. The statistics for ToG, PoG, and RJE are taken from RJE (Lin et al., 2025); our statistics are measured under our own implementation. Lower is better for all metrics.

Compared with ToG (Sun et al., 2024), PoG (Tan et al., 2025), and RJE (Lin et al., 2025), our method reduces prompt length because graph traversal is performed in embedding space rather than through repeated LLM interaction at each hop. The reduction in output tokens is particularly large, since the LLM is used only once for final answer generation instead of producing intermediate reasoning traces or retrieval guidance multiple times. These results show that the efficiency advantage of the proposed framework extends beyond call count to overall token consumption.

I Robustness Analysis on Noisy Graphs

We conduct a robustness analysis to examine how the proposed retriever behaves under noisy graph inputs. To simulate graph noise, we perturb the KG edges while keeping the original questions and gold answers unchanged. Specifically, we apply degree-aware edge rewiring so that the local graph structure is partially randomized while preserving the overall scale of the retrieved graph.

This setting evaluates whether the retriever can still recover useful reasoning paths when the graph contains misleading edges. Since the questions and gold answers are fixed, performance changes mainly reflect the robustness of the retrieval process to graph perturbations.

We compare the original graph with noisy graphs generated using different perturbation ratios. The perturbation ratio p is the fraction of KG edges that are rewired. A higher perturbation ratio introduces more incorrect or misleading edges into the local graph neighborhood. We report Hit rate on WebQSP and CWQ under each setting.

Noise Setting	WebQSP Hit (%)	CWQ Hit (%)
Original graph	86.0	73.0
Noisy graph, $p = 0.20$	85.2	72.9
Noisy graph, $p = 0.40$	85.1	72.9

Table 16: Robustness analysis under noisy graph perturbations.

Table 16 shows the results for the original graph and for the two perturbation ratios. Compared with the original graph, noisy graph perturbations lead to a moderate performance drop. On CWQ, Hit decreases from 73.0% to 72.9%, while increasing the perturbation ratio from $p = 0.20$ to $p = 0.40$ causes little additional degradation. This suggests that residual-guided retrieval is relatively robust to local graph noise.

J Fine-Grained Error Analysis

To better understand the limitations of the proposed residual-based retrieval, we conduct a fine-grained analysis of the errors on CWQ. The proposed method achieves 2,579 matches out of 3,531 questions, while 952 questions are not matched. We first distinguish whether the gold answer appears in the retrieved paths. Among the 952 errors, 659 cases do not contain the gold answer in the retrieved evidence, while 293 cases contain the gold answer but are still answered incorrectly. Table 17 summarizes this breakdown. This suggests that the main bottleneck is evidence retrieval rather than final answer extraction.

These results motivate a dynamic weighting direction that remains consistent with our training-free setting. Rather than training a full task-specific retriever or adding supervised modules, future work could estimate question-level retrieval difficulty or compositional complexity directly from the input question. This signal could dynamically adjust the balance between the residual signal and the original question signal, allowing the retriever to use stronger residual guidance for highly compositional questions while preserving more original question semantics for simpler cases.

Error Type	#Errors	Ratio
Gold answer not in retrieved paths	659	69.2%
Gold answer in retrieved paths but missed	293	30.8%
Total errors	952	100.0%

Table 17: Breakdown of errors based on whether the gold answer appears in the retrieved paths.

Failure Pattern	Observed Behavior	Example
Missing constraint evidence	A relevant intermediate entity is retrieved, but the final constraint relation is missing.	The method identifies Brad Paisley but does not retrieve Belmont University.
Relation drift	Retrieval follows a semantically related but incorrect relation.	The path moves toward nationality, birthplace, or general location instead of the required relation.
Premature path shrinkage	A correct early branch is found, but later pruning removes the branch leading to the gold answer.	The method retrieves San Francisco Giants and some World Series paths, but misses 2014 World Series.

Table 18: Representative retrieval-side failure patterns observed in the error analysis.

We further categorize retrieval-side failures into three representative patterns, as shown in Table 18. First, *missing constraint evidence* occurs when the method retrieves a relevant intermediate entity but fails to expose the final relation or entity required by the question. Second, *relation drift* occurs when retrieval follows a semantically related but incorrect relation, such as nationality, birthplace, or general containment, instead of the relation required by the question. Third, *premature path shrinkage* occurs when the correct reasoning branch is not retained through later hops, even though an earlier hop reaches a plausible intermediate entity.

K Case Study: Effect of the Residual Signal

To better understand why the residual signal improves multi-hop retrieval, we provide a case study on CWQ. The goal of this analysis is not to show another benchmark result, but to inspect how the retrieved reasoning paths change when the residual signal is removed.

We compare the proposed method with a variant that does not use the residual signal. Without the residual signal, retrieval is mainly guided by the original question and topic entities throughout all hops. As a result, the search often continues in the same semantic direction even after part of the question has already been resolved. In contrast, the residual signal updates the retrieval target after each expansion by reducing the influence of information already covered by the current path. This makes the next-hop retrieval focus on the remaining unresolved constraint.

In the tables below, red indicates the correct answer or correct reasoning evidence, while blue indicates an incorrect answer or misleading reasoning evidence.

In Table 19, both methods first identify David Beckham as the person educated at Chase Lane Primary School. However, after this information is covered, the next search step should focus on the remaining constraint: the team he played for around 2011. The variant without residual continues to retrieve generally relevant football-career relations and follows an earlier loan relation, leading to Preston North End F.C. The proposed method instead shifts the search target toward the unresolved team constraint and retrieves a path that contains LA Galaxy. This illustrates how the residual signal prevents the search from repeatedly following the same broad semantic direction.

Table 20 shows a case where the non-residual variant is biased toward the prominent topic entity, Angelina Jolie. Since many paths from Angelina Jolie are semantically related to films, retrieval without residual keeps expanding her general filmography. However, the question requires a film that also includes the role Maida. The proposed method reduces the influence of already-covered information and retrieves paths from the remaining role constraint, allowing it to connect Maida to the correct film and then verify that the film was directed by Angelina Jolie.

Question	The person educated at Chase Lane Primary School played for what team in 2011?
w/o Residual Paths	Chase Lane Primary School → *students_graduates* → David Beckham → *loans* → 1995–1995 → *borrowing_team* → Preston North End F.C.
Proposed Paths	Chase Lane Primary School → *students_graduates* → David Beckham → *world_cup_team* → current squad → *current_club* → LA Galaxy
Outputs	**w/o Residual**: Preston North End F.C. | Manchester United F.C. **Proposed**: David Beckham | LA Galaxy
Ground Truth	LA Galaxy

Table 19: Case study showing that the residual signal helps avoid continuing in a locally related but outdated direction. Without the residual signal, the retrieval follows David Beckham's earlier loan relation and misses the 2011 team. With the residual signal, the remaining constraint about the team is better reflected in later-hop retrieval.

Question	What film directed by Angelina Jolie is the flick that includes the role of Maida?
w/o Residual Paths	Angelina Jolie → *film.actor.film* → A Mighty Heart, Changeling, Lara Croft Tomb Raider: The Cradle of Life, . . .
Proposed Paths	Maida → *film_character.portrayed_in_films* → Film performance → *performance.film* → In the Land of Blood and Honey → *film.directed_by* → Angelina Jolie
Outputs	**w/o Residual**: A Mighty Heart | Changeling | Lara Croft Tomb Raider: The Cradle of Life | · · · **Proposed**: In the Land of Blood and Honey
Ground Truth	In the Land of Blood and Honey

Table 20: Case study showing that residual-guided retrieval shifts attention from the already-covered topic entity to the missing constraint. Without the residual signal, the search remains biased toward Angelina Jolie's general filmography. With the residual signal, the search moves to the role entity Maida and retrieves the film satisfying both constraints.