

Large Language Model-based Textual Graph Retrieval Agent with Minimal Toolset

Wooyoung Kim¹, Doyoung Kwon¹, Haemin Jung² and Wooju Kim^{1,*}

¹Department of Industrial Engineering, Yonsei University, Seoul, Republic of Korea

²Korea National University of Transportation, Chungju, Republic of Korea

Abstract

This paper introduces TeGRA, a large language model-based **Textual Graph Retrieval Agent** that combines textual and graph-structured information. To overcome the limitations of unimodal retrieval systems, TeGRA integrates textual similarity search and graph relation expansion within a reasoning loop controlled by a language model. The agent leverages a compact set of high-level tools inspired by the complex instruction set computer design philosophy to reduce action space complexity and enable interpretable multi-step reasoning. Experiments conducted on the STaRK benchmark demonstrate that TeGRA achieves superior retrieval performance compared to traditional retrievers and reduced instruction set computer-style agents, particularly in tasks requiring multi-hop relational inference. These findings highlight the effectiveness of structured tool composition for retrieval-augmented reasoning over hybrid knowledge sources.

Keywords

Large Language Models, Semi-Structured Knowledge Base, Graph Retrieval, Tool-Augmented Agents

1. Introduction

Natural language queries have become the default interface for information seeking in complex, real-world domains such as e-commerce, academic search, and precision medicine. These information needs often combine free-form language with structured constraints—for example, requesting items that match certain properties, belong to specific categories, or satisfy multi-step relational conditions. In such settings, the underlying knowledge takes the form of semi-structured knowledge bases (SKBs) composed of textual documents and entity-relation graphs, and retrievers based solely on dense or sparse embeddings often fall short when queries demand relational reasoning or the integration of multiple modalities. This has motivated a growing interest in agent-based retrieval systems built on large language models (LLMs) that incorporate explicit tool usage for structured reasoning.

Existing agent systems iteratively invoke a sequence of tools, such as text retrieval and graph traversal, to progressively narrow down the answer space. In many implementations, however, the action space consists of numerous fine-grained tools, requiring the agent to make decisions at every low-level step; this significantly increases search complexity and places a heavy cognitive load on the controller, especially in multi-hop reasoning tasks.

In this work, we propose a structured agent framework that reduces this burden by adopting a high-level tool abstraction inspired by the principles of Complex Instruction Set Computer (CISC) design. Rather than decomposing reasoning into dozens of micro-operations, we design a compact toolset in which each tool encapsulates a semantically meaningful and reusable operation, such as text similarity search or graph relation expansion. We instantiate this idea in the **Textual Graph Retrieval Agent** (TeGRA), which integrates text and graph-based retrieval in a unified agent loop combining a memory module retaining the reasoning trace, a modular tool interface, and an LLM controller conditioned on accumulated context.

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

*Corresponding author.

✉ timothy@yonsei.ac.kr (W. Kim); dykwon0312@yonsei.ac.kr (D. Kwon); hmjung@ut.ac.kr (H. Jung); wkim@yonsei.ac.kr (W. Kim)

ORCID 0009-0001-5930-9892 (W. Kim); 0009-0000-5358-4968 (D. Kwon); 0000-0002-1182-9303 (H. Jung); 0000-0001-5828-178X (W. Kim)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

We evaluate TeGRA on the STaRK benchmark, whose semi-structured queries demand both semantic matching and relational reasoning across multiple domains. Results show that our CISC-style agent outperforms traditional retrievers and achieves competitive performance compared to existing agent-based methods, particularly in tasks that benefit from graph-structured reasoning, while reducing action space complexity and maintaining reasoning traceability.

2. Related Work

Early attempts to integrate LLMs with graph-structured knowledge injected structured knowledge during pretraining, e.g., by linearizing knowledge graph triples into textual sequences [1]; such methods improved factual coverage but scaled poorly to long-tail and compositional queries. The field has since moved toward retrieval-augmented generation (RAG), which retrieves external knowledge at inference time and decouples knowledge access from model parameters. Graph information can also be injected through prompting, via discrete prompt optimization for knowledge graph question answering [2] or continuous prompt vectors that summarize graph context in graph-augmented LLMs [3].

The STaRK benchmark [4] evaluates retrieval over SKBs that combine unstructured text with relational graph information, and shows that traditional retrievers and LLMs often fail on such hybrid queries. Several systems bridge structured graph reasoning and text-based retrieval to address this challenge. The Mixture of Retrieval approach [5] interleaves structural traversal and textual matching over a reasoning graph, while HybGRAG [6] couples a retriever bank spanning structured and unstructured sources with a self-reflective critic. QA-GNN [7] connects questions to candidate nodes and performs message passing for context-aware, multi-hop reasoning, and Think-on-Graph [8] grounds LLM generation in beam search over knowledge graph edges to construct explicit reasoning chains.

Tool-augmented LLM agents have become a dominant paradigm more broadly: MRKL [9] routes subtasks to specialized modules, ReAct [10] interleaves natural language reasoning with tool actions, Toolformer [11] learns when to invoke external APIs, Gorilla [12] routes inputs across a large space of open-source APIs, and AvaTaR [13] improves tool usage via contrastive reasoning over successful and failed tool sequences. An important design axis in these systems is the structure of the action space: some follow a Reduced Instruction Set Computer (RISC)-style paradigm of fine-grained tool invocations, while others—including TeGRA—adopt a CISC-style approach that encapsulates complex operations into a few high-level tools.

3. Methodology

3.1. Problem Formulation

The goal is to design a retrieval agent that, given a natural language query q , dynamically extracts relevant information from an SKB and returns a candidate answer set of nodes $\mathcal{A} \subseteq \mathcal{V}$. The SKB consists of a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and directed edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a set of free-form textual documents $\mathcal{D} = \{\mathcal{D}_i \mid i \in \mathcal{V}\}$ associated with individual nodes. A correct answer node $i \in \mathcal{A}$ must (1) satisfy the relational conditions specified in q within the structural context of $\mathcal{G}$, and (2) have an associated document $\mathcal{D}_i$ that semantically matches the textual requirements implied by q . This problem cannot be addressed by pure text-based retrieval or isolated graph traversal alone; it requires a hybrid approach that integrates both structural and semantic signals.

3.2. Textual Graph Retrieval Agent

TeGRA is composed of three modular components—the Memory Module, the Tool Module, and the LLM Module—that interact in an iterative reasoning loop inspired by the ReAct framework [10]. Figure 1 presents the high-level architecture, and Algorithm 1 outlines the core decision-making loop: the LLM observes the current memory trace, selects a high-level tool from a predefined set, and updates the memory with the tool’s output, until it determines that retrieval is complete.

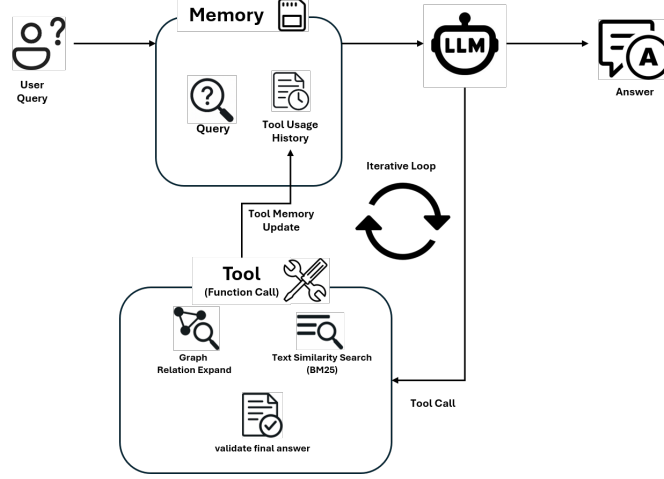


Figure 1: Overall architecture of **TeGRA**. The agent operates in an iterative ReAct-style loop in which the LLM observes the current memory state, selects a high-level tool, and invokes it with appropriate arguments; the memory stores the full trace of prior tool invocations and outcomes.

Algorithm 1 TeGRA: Tool-Augmented Textual Graph Retrieval Agent

Require: Knowledge Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Document Set $\mathcal{D}$, Query q

Ensure: Answer node set $\mathcal{A} \subseteq \mathcal{V}$

- 1: Initialize Memory $M \leftarrow \{q\}$
 - 2: Actions $\leftarrow \{\text{text_similarity_search}, \text{expand_graph_relation}, \text{validate_final_answer}, \text{stop}\}$
 - 3: **while** True **do**
 - 4: LLM observes current memory M
 - 5: LLM selects next action $\in$ Actions
 - 6: **if** action == text_similarity_search **then**
 - 7: Retrieve candidate nodes $C \leftarrow \text{TEXT_SIMILARITY_SEARCH}(q, \mathcal{D})$
 - 8: Update $M \leftarrow M \cup \{C\}$
 - 9: **else if** action == expand_graph_relation **then**
 - 10: Select node $v \in \mathcal{V}$ and relation r
 - 11: Expand neighbors $N_v^r \leftarrow \text{EXPAND_GRAPH_RELATION}(v, r, \mathcal{G})$
 - 12: Update $M \leftarrow M \cup \{N_v^r\}$
 - 13: **else if** action == validate_final_answer **then**
 - 14: Validate candidate set $\mathcal{A} \leftarrow \text{VALIDATE_FINAL_ANSWER}(M)$
 - 15: Update $M \leftarrow M \cup \{\mathcal{A}\}$
 - 16: **else if** action == stop **then**
 - 17: **return** $\mathcal{A}$
 - 18: **end if**
 - 19: **end while**
-

The **Memory Module** maintains the internal state of the agent: starting from the initial query q , it is incrementally updated with the inputs and outputs of each tool invocation in chronological order. This trace is actively used as part of the evolving prompt at each decision step, so the LLM conditions its next action on the accumulated context, enabling coherent and memory-aware tool orchestration.

The **Tool Module** provides the agent with a set of external retrieval operations implemented as callable functions. Rather than decomposing functionality into fine-grained atomic actions, each tool is a high-level, semantically coherent operation:

- `text_similarity_search`: retrieves entity nodes whose associated texts are similar to the user query q (or a query generated by the LLM), using an off-the-shelf BM25 retriever [14] over

the documents $\mathcal{D}$ linked to the knowledge graph.

- `expand_graph_relation`: given an entity node and a relation type, explores adjacent nodes in the knowledge graph; it supports recursive expansion along multiple relation types, enabling multi-hop traversal.
- `validate_final_answer`: verifies whether the final output complies with the evaluation criteria and formatting constraints before the agent halts.

This design philosophy draws on the CISC architecture, in which frequently used complex operations are encapsulated within single high-level instructions: `text_similarity_search` internally performs document vectorization, similarity scoring, and entity ID extraction, while `expand_graph_relation` performs relation-based filtering, neighbor traversal, and duplicate elimination. In contrast, RISC-style decomposition into many fine-grained tools inflates the number of options, increases the tool-selection load on the LLM, and fragments the reasoning flow. By encapsulating each conceptually unified operation as one tool, TeGRA minimizes the number of interactions required per reasoning task while preserving the traceability of reasoning paths and the interpretability of tool usage.

The **LLM Module** governs the reasoning process. At each iteration, the LLM observes the current memory state and generates a natural language action: either selecting a tool to invoke and its parameters, or halting. The LLM is not trained to predict final answers directly; it orchestrates retrieval through tool selection and memory interpretation.

4. Experiments

4.1. Experimental Setup

We use the STaRK benchmark [4], a high-difficulty benchmark grounded in SKBs, comprising three domain-specific sub-datasets: **STaRK-Amazon** (product recommendation; entities such as products, brands, categories, and colors, with attribute- and review-based constraints), **STaRK-MAG** (academic paper recommendation over papers, authors, and institutions, frequently requiring multi-hop reasoning), and **STaRK-PRIME** (biomedicine; anatomical structures, genes, proteins, diseases, drugs, and pathways, with scarce documents and thus a stronger reliance on graph reasoning). Performance is measured with standard ranking-oriented retrieval metrics: Hits@ K ($K = \{1, 5\}$), Recall@20, and Mean Reciprocal Rank (MRR). The agent is implemented with the OpenAI o3-mini API model (o3-mini-2025-01-31) under a ReAct-style reasoning loop over the three tools described above, with tool outputs stored in the structured memory module. Baselines span five categories of retrieval techniques: sparse retrieval (BM25 [14]), dense retrieval (DPR [15], GritLM-7b [16], LLM2Vec [17], and text-embedding-ada-002), multivector retrieval (ColBERTv2 [18]), graph-based retrieval (QA-GNN [7]), and LLM-based re-rankers (GPT-4 and Claude 3 rerankers following the STaRK protocol [4]).

4.2. Main Results

Table 1 compares retrieval performance across baselines and datasets. Our method achieves consistently strong results on all four metrics and the highest overall average performance, with Hits@1 of 47.87, Hits@5 of 59.48, Recall@20 of 49.44, and MRR of 55.10. Compared to single-modality methods, our model exhibits clear improvements across datasets, highlighting the effectiveness of jointly reasoning over linguistic relevance and the structural relations of the knowledge graph.

The relatively lower performance on AMAZON can be attributed to its higher proportion of textual information [4]: many AMAZON queries can be resolved through surface-level textual matching, where single-modality LLM rerankers hold a slight edge. Our method nevertheless remains competitive overall and performs best precisely where structural reasoning is required.

Table 1

Retrieval performance of various baselines and our proposed agent across STaRK-AMAZON, MAG, and PRIME. The proposed model outperforms all baselines on average.

Method	AMAZON				MAG				PRIME				Average			
	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR
BM25	27.16	51.85	29.23	18.79	32.14	41.67	32.46	37.42	22.45	41.84	42.32	30.37	27.25	45.12	34.67	28.86
DPR (roberta)	16.05	39.51	15.23	27.21	4.72	9.52	25.00	7.90	2.04	9.18	10.69	7.05	7.60	19.40	16.97	14.05
GritLM-7b	40.74	71.60	36.30	53.21	34.52	44.04	34.57	38.72	25.51	41.84	48.10	34.28	33.59	52.49	39.66	42.07
LLM2Vec	29.63	46.91	21.21	38.61	16.67	28.57	21.74	21.59	9.18	21.43	26.77	15.24	18.49	32.30	23.24	25.15
ada-002	39.50	64.19	35.46	52.65	28.57	41.67	35.95	35.81	17.35	34.69	41.09	26.35	28.47	46.85	37.50	38.27
ColBERTv2	33.33	55.56	29.03	43.77	33.33	36.90	30.50	35.97	15.31	26.53	25.56	19.67	27.32	39.66	28.36	33.14
QA-GNN (roberta)	22.22	49.38	21.54	31.33	20.24	26.19	28.76	25.53	6.12	13.27	17.62	9.39	16.19	29.61	22.64	22.08
GPT4 Reranker	50.62	75.31	35.46	61.06	36.90	46.43	35.95	40.65	28.57	44.90	41.61	34.82	38.70	55.55	37.67	45.51
Claude3 Reranker	53.09	74.07	35.46	62.11	38.10	45.24	35.95	42.00	28.57	46.94	41.61	36.32	39.92	55.42	37.67	46.81
Ours	49.38	62.96	26.45	57.33	45.24	51.19	59.23	50.81	48.98	64.29	62.63	57.16	47.87	59.48	49.44	55.10

Table 2

Comparison between our CISC-style retrieval agent (TeGRA, “Ours”), its reranked variant (“Ours-Rerank”), and the RISC-style agent AvaTaR across the three STaRK datasets.

Method	AMAZON				MAG				PRIME				Average			
	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR	Hits@1	Hits@5	Recall@20	MRR
AvaTaR	58.32	76.54	42.43	65.91	33.33	42.86	35.94	38.62	33.03	51.37	53.34	41.00	41.56	56.92	43.90	48.51
Ours	49.38	62.96	26.45	57.33	45.24	51.19	59.23	50.81	48.98	64.29	62.63	57.16	47.87	59.48	49.44	55.10
Ours-Rerank	49.38	62.96	26.45	57.48	50.00	65.48	62.51	58.14	53.06	68.37	62.29	61.05	50.81	65.60	50.42	58.89

4.3. Reranking and CISC vs. RISC Agent Design

To further improve retrieval quality, we add an LLM-based reranking module on top of TeGRA’s initial retrieval outputs (**Ours-Rerank** in Table 2). Reranking preserves the top-1 predictions on all datasets but consistently improves metrics that consider broader candidate quality: MRR improves from 50.81 to 58.14 on MAG and from 57.16 to 61.05 on PRIME, with only marginal change on the text-dominated AMAZON dataset—i.e., the reranker is most beneficial precisely where complex multi-hop reasoning is required.

Table 2 also compares TeGRA with AvaTaR [13], a RISC-style agent equipped with a suite of 22 distinct tool functions (based on GPT-4-turbo). AvaTaR achieves the highest performance on the text-heavy AMAZON dataset (Hits@1 58.32, MRR 65.91), but TeGRA performs substantially better where deeper structural reasoning is required: TeGRA-Rerank outperforms AvaTaR in MRR by +19.52 on MAG and +20.05 on PRIME, and achieves the highest average performance (58.89 MRR). This supports the hypothesis that CISC-style tool composition—encapsulating semantic expansion and multi-hop reasoning within coherent high-level tools—is better suited to complex retrieval scenarios than RISC-style modularity alone.

5. Conclusion

This study introduced TeGRA, a tool-augmented retrieval agent for question answering over SKBs that dynamically integrates textual matching and graph traversal through iterative tool calls guided by an LLM. Its compact, CISC-style toolset encapsulates semantically coherent operations as high-level tools, reducing action space complexity while enabling efficient and interpretable multi-step reasoning. Experiments on the STaRK benchmark show that TeGRA outperforms unimodal baselines on average across all datasets and surpasses a RISC-style agent precisely in the domains requiring structural reasoning. These results highlight the importance of aligning tool design with reasoning requirements in LLM-based agents, and we believe CISC-style agents like TeGRA offer a principled foundation for integrating symbolic structure and neural computation.

Declaration on Generative AI

During the preparation of this work, the authors used Claude (Anthropic) in order to: check grammar and spelling, and assist with template formatting.

Acknowledgment

This work was supported by research funding from a project conducted in collaboration with the Korea Technology Finance Corporation (KOTEC) (Project No. 2026-11-1415).

References

- [1] W. Kim, H. Jung, W. Kim, Knowledge graph as pre-training corpus for structural reasoning via multi-hop linearization, *IEEE Access* 13 (2025) 7273–7283.
- [2] W. Kim, H. Jung, B. Kim, S. Kwon, W. Kim, Graph discrete prompt optimization for knowledge graph question answering, in: *Proceedings of the ACM Web Conference 2026*, 2026, pp. 8353–8356.
- [3] W. Kim, W. Kim, Addressing information bottlenecks in graph augmented large language models via graph neural summarization, *Information Fusion* 127 (2026) 103784.
- [4] S. Wu, S. Zhao, M. Yasunaga, K. Huang, K. Cao, Q. Huang, V. N. Ioannidis, K. Subbian, J. Zou, J. Leskovec, Stark: Benchmarking llm retrieval on textual and relational knowledge bases, *Advances in Neural Information Processing Systems* 37 (2024) 127129–127153.
- [5] Y. Lei, H. Han, R. A. Rossi, F. Deroncourt, N. Lipka, M. M. Halappanavar, J. Tang, Y. Wang, Mixture of structural-and-textual retrieval over text-rich graph knowledge bases, *arXiv preprint arXiv:2502.20317* (2025).
- [6] M.-C. Lee, Q. Zhu, C. Mavromatis, Z. Han, S. Adeshina, V. N. Ioannidis, H. Rangwala, C. Faloutsos, Hybgrag: Hybrid retrieval-augmented generation on textual and relational knowledge bases, *arXiv preprint arXiv:2412.16311* (2024).
- [7] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, J. Leskovec, Qa-gnn: Reasoning with language models and knowledge graphs for question answering, *arXiv preprint arXiv:2104.06378* (2021).
- [8] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, L. M. Ni, H.-Y. Shum, J. Guo, Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph, *arXiv preprint arXiv:2307.07697* (2023).
- [9] E. Karpas, O. Abend, Y. Belinkov, B. Lenz, O. Lieber, N. Ratner, Y. Shoham, H. Bata, Y. Levine, K. Leyton-Brown, et al., Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning, *arXiv preprint arXiv:2205.00445* (2022).
- [10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafraan, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: *International Conference on Learning Representations (ICLR)*, 2023.
- [11] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, *Advances in Neural Information Processing Systems* 36 (2023) 68539–68551.
- [12] S. G. Patil, T. Zhang, X. Wang, J. E. Gonzalez, Gorilla: Large language model connected with massive apis, 2023, URL <https://arxiv.org/abs/2305.15334> (2023).
- [13] S. Wu, S. Zhao, Q. Huang, K. Huang, M. Yasunaga, K. Cao, V. N. Ioannidis, K. Subbian, J. Leskovec, J. Zou, Avatar: Optimizing llm agents for tool usage via contrastive reasoning, in: *NeurIPS*, 2024.
- [14] S. Robertson, H. Zaragoza, et al., The probabilistic relevance framework: Bm25 and beyond, *Foundations and Trends® in Information Retrieval* 3 (2009) 333–389.
- [15] V. Karpukhin, B. Oguz, S. Min, P. S. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering., in: *EMNLP* (1), 2020, pp. 6769–6781.
- [16] N. Muennighoff, S. Hongjin, L. Wang, N. Yang, F. Wei, T. Yu, A. Singh, D. Kiela, Generative representational instruction tuning, in: *The Thirteenth International Conference on Learning Representations*, 2024.
- [17] P. Behnamghader, V. Adlakha, M. Mosbach, D. Bahdanau, N. Chapados, S. Reddy, Llm2vec: Large language models are secretly powerful text encoders, *arXiv preprint arXiv:2404.05961* (2024).
- [18] K. Santhanam, O. Khattab, J. Saad-Falcon, C. Potts, M. Zaharia, Colbertv2: Effective and efficient retrieval via lightweight late interaction, *arXiv preprint arXiv:2112.01488* (2021).