StarGraph: Text-Preserving Meta-Path Indexes for RAG, Agents, and Tool-Grounded LLM Systems

Yuxin Yang[1], Bizhao Shi[1], Yuxuan Zhou[2], Jieran Zhang[1], Haoyang Wu[1], Tao Wang[1], Jia Yang[1], Hao Ma[1] and Guojie Luo[1]

[1]Peking University, No. 5 Yiheyuan Road, Haidian District, Beijing 100871, China
[2]Tsinghua University, No. 30 Shuangqing Road, Haidian District, Beijing 100084, China

Abstract

Knowledge-graph-centric LLM systems need a representation that is neither a plain vector store nor a conventional graph database. A vector store keeps readable text but weakens relational control, while a graph database keeps structure but often removes the provenance needed for generation, agent planning, and tool use. This short paper describes StarGraph, a representation layer in which text segments, entities, typed relations, tool/action records, and meta-path indexes coexist in one graph. The central idea is to make provenance and path structure first-class operational objects: text nodes keep source evidence available, and node-level meta-path indexes summarize reusable relation patterns before queries or agent actions arrive. We present the data model, construction procedure, update behavior, and affordances of StarGraph for RAG, agentic workflows, tool grounding, and memory maintenance. Across four LLM backbones, StarGraph improves over LightRAG by an average of 14.68 points on MultiHop and 10.06 on RGB; at meta-path length five, pre-built indexes reduce average node visits from 899 to 88.

Keywords

Knowledge graph, retrieval-augmented generation, agent systems, tool grounding, meta-path index

1. Introduction

Retrieval-augmented generation (RAG) systems are often built on chunk indexes: documents are split into passages, embedded, and searched by sparse or dense similarity [1, 2, 3]. This keeps evidence readable, but treats relations among entities as implicit text. Knowledge graphs (KGs) provide the opposite strength: they expose entities and relations, which is useful when a question depends on a chain of facts [4, 5, 6]. Yet a pure triple graph is also insufficient for generation, because triples usually omit source phrasing, context, and provenance.

The RAGE-KG setting therefore needs a representation with three properties: textual provenance for grounding, typed graph structure for relation-aware retrieval, and precomputed path access for low-latency interaction. Meta-path research shows that typed path patterns carry semantics distinct from untyped connectivity [7, 8]. StarGraph is designed around these requirements. Rather than treating chunks, triples, and path indexes as separate artifacts, it places them inside one graph layer.

This paper focuses on that representation layer. Although RAG is the immediate use case, the same representation can support agent planning, tool selection, execution trace memory, and post-hoc auditing. The contributions are:

- We define StarGraph, a KG substrate that jointly represents source text, entities, typed relations, tool/action records, and meta-path availability.
- We describe a construction and maintenance workflow that preserves provenance while supporting local refresh of path indexes.
- We analyze how the same representation supports RAG, agent planning, tool grounding, trace memory, and auditing.

3rd International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs (RAGE-KG 2026)
yxyang@pku.edu.cn (Y. Yang); bshi@pku.edu.cn (B. Shi); zhou-yx21@mails.tsinghua.edu.cn (Y. Zhou); jrzhang@pku.edu.cn (J. Zhang); why@stu.pku.edu.cn (H. Wu); wangtao@pku.edu.cn (T. Wang); yangj@pku.edu.cn (J. Yang); mah@pku.edu.cn (H. Ma); gluo@pku.edu.cn (G. Luo)
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-04

CEUR-WS.org/Vol-4312/short9.pdf

Table 1
Representation requirements for KG-centric LLM workflows.

Requirement	Vector index	Triple KG	StarGraph
Readable evidence	**yes**	**no**	**yes**
Entity relations	**no**	**yes**	**yes**
Source provenance	**yes**	**partial**	**yes**
Multi-hop routing	**no**	**online**	**indexed**
Tool/action schema	**no**	**partial**	**yes**
Trace memory	**partial**	**no**	**yes**

- We report component ablations, four-backbone transfer results, and RGB path-length sensitivity, showing complementary quality and efficiency gains.

2. Related Work

Retrieval and graph grounding. RAG connects language models with external evidence at inference time [1]. Beyond fixed retrieve-then-generate pipelines, FLARE retrieves during generation and Self-RAG learns when to retrieve and critique evidence [9, 10]. Graph-based systems instead organize retrieval structure: GraphRAG aggregates entity communities, LightRAG combines local and global graph retrieval, and HippoRAG couples a KG with graph-based memory access [11, 12, 13]. Retrieval quality remains decisive [14]; moreover, systematic comparisons find that text RAG and GraphRAG have task-dependent strengths rather than a universal ordering [15]. StarGraph therefore focuses on a shared representation that keeps source text, typed relations, and path indexes together.

Agents and tool-using LLMs. Recent LLM agents combine reasoning with actions, tools, and memory. ReAct interleaves reasoning and actions [16]; Toolformer studies tool-use behavior [17]; Gorilla targets API selection [18]; Voyager maintains reusable skills [19]; Reflexion stores verbal feedback [20]; and Generative Agents use memory streams for long-horizon behavior [21]. These systems motivate a representation that stores evidence, actions, tool schemas, and traces in one inspectable substrate.

3. Representation Design

StarGraph is a directed, typed graph $G = (V, E, P, M)$. The node set contains entity nodes and text nodes, $V = V_e \cup V_t$. Edges E include ordinary typed relations between entities and provenance links between entities and text segments. The set P stores meta-path signatures, and M maps nodes to the signatures that can be instantiated from them:

$$
\begin{aligned}
V &= V_e \cup V_t, \\
E &= E_r \cup E_x, \\
P &= \{\pi = (\mathcal{T}_0, \mathcal{R}_0, \ldots, \mathcal{T}_k) \mid k \le l\}, \\
M &: V_e \to 2^P.
\end{aligned}
$$

Here $\mathcal{T}_i$ and $\mathcal{R}_i$ denote node and relation types; E_r and E_x denote entity-relation and provenance edges, respectively. A text node stores source, position, text, and embedding; an entity node stores type, aliases, and normalization. An edge $(v_e, v_t, \textit{Extract}) \in E_x$ states that the entity was observed in the text segment and is retained after extraction rather than discarded as construction metadata.

Meta-paths provide a second layer of organization. A signature describes a sequence of node and relation types, while an instance is a concrete walk matching that signature [7, 8]. StarGraph stores signatures globally in P and records node-level availability in M, so a retriever can inspect possible structural patterns before path instantiation. The index acts as a compact routing table over the graph.

Table 1 summarizes the design position. StarGraph does not replace vector indexes or symbolic KGs; it records what LLM workflows need from both. The key difference from a post-hoc hybrid system is that provenance, path availability, tool constraints, and trace records are represented in the graph itself.

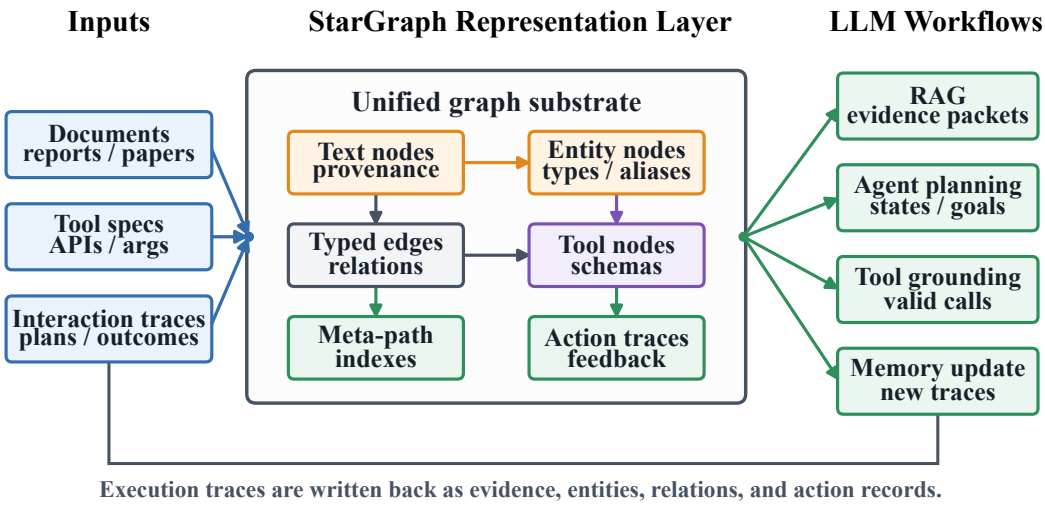


Figure 1: Overview of StarGraph as a shared substrate for RAG, agent planning, tool grounding, and memory updates.

Query-time access. Given a query q, a text retriever first identifies semantic anchors $A_t(q) \subseteq V_t$, while entity linking produces structural anchors $A_e(q) \subseteq V_e$. Provenance edges connect the two views, so either an entity mention or a semantically matched passage can initiate retrieval. For each entity anchor v, the retriever reads $M(v)$, ranks the available signatures against q, and instantiates only selected signatures rather than enumerating arbitrary walks. The returned context is

$$I(q) = \mathrm{TopK}\left(A_t(q) \cup \bigcup_{v \in A_e(q)} \mathrm{Inst}(v, \mathrm{Select}(q, M(v))) \right),$$

where path instances carry their endpoint entities, typed edges, and linked source segments. This interface is deliberately retrieval-model agnostic: exact matching, dense similarity, or an LLM selector may rank anchors and signatures, but all of them consume the same inspectable graph objects. It also distinguishes StarGraph from methods that serialize a large neighborhood and ask the generator to discover useful relations inside the prompt.

4. Construction and Maintenance

Algorithm 1 outlines construction. Documents are segmented into stable text units and inserted as text nodes before extraction. Entity and relation extraction then produces typed entity nodes and relation edges. The extractor can use rules, neural taggers, or an LLM-assisted parser; recent LLM-based construction work likewise separates open extraction from schema definition and canonicalization [22]. Each entity mention is connected back to its source segment through an *Extract* edge.

The workflow separates evidence capture from structural indexing. Text nodes and provenance edges are stable under later extractor changes. Relation edges can be corrected or retyped without deleting the source segment, and meta-path indexes can be refreshed locally when affected neighborhoods change. In particular, inserting or deleting an edge only invalidates signatures rooted in the bounded l-hop neighborhood of its endpoints. An implementation can therefore mark those roots dirty and rebuild their entries in M, while leaving unrelated text, entities, and path indexes untouched. This locality matters for continuously updated corpora because full graph reconstruction would otherwise erase the latency advantage of offline indexing.

The materialized records cover text and entity nodes, relation and provenance edges, tool schemas, action traces, meta-path signatures, and concrete path instances. The schema does not prescribe a backend: a graph database, property graph, or adjacency-backed document store is sufficient if runtime code can move from entities to evidence, typed neighbors, valid tools, and prior traces.

Algorithm 1 StarGraph construction

Input: document collection $\mathcal{D}$, maximum meta-path length l
initialize V_e, V_t, E_r, E_x, P, M
for each document $d \in \mathcal{D}$ **do**
    create text nodes from segmented units of d
    extract typed entities and candidate relations
    add or merge entity nodes in V_e
    add relation edges to E_r
    add provenance edges to E_x
for each entity node $v \in V_e$ **do**
    enumerate bounded path instances from v
    abstract instances into meta-path signatures
    update P and node index $M(v)$
return $G = (V, E, P, M)$

5. Affordances Beyond RAG

StarGraph separates semantic and symbolic signals: text embeddings find anchors and source passages, typed edges preserve relational precision, and meta-path indexes select structural routes before instantiation, complementing joint graph reasoning and explicit KG path search [23, 24, 25]. Retrieved paths remain interpretable because their nodes and edges link back to source text; failures can be traced to missing extraction edges, incorrect relation types, or absent signatures.

The same objects extend beyond RAG. Meta-path indexes encode valid planning transitions, tool nodes store argument constraints, action traces provide reusable memory, provenance edges support auditing, and typed relations expose drift. Thus an agent can recover state, select valid tools, and write execution outcomes back into the graph without a separate memory representation.

6. Experimental Study

6.1. Setup

We use two complementary RAG benchmarks. MultiHop-RAG requires retrieving and integrating multiple supporting passages, whereas RGB tests noise robustness, negative rejection, information integration, and counterfactual robustness [26, 27]. Benchmark text is integrated into one corpus, with non-target entries retained as distractors; scores follow the benchmark-provided 0–100 scale.

We compare four matched retrieval conditions: Base has no retrieval, ES uses Elasticsearch vector search [28], LightRAG is the graph baseline [12], and StarGraph adds text-preserving typed structure and pre-built meta-path indexes. Backbones are GPT-OSS-20b [29], LLaMA3-70b [30], Qwen3-plus [31], and DeepSeek-V3.2-Exp [32]. Only retrieval/representation changes within a panel; stated means macro-average the four backbones. We also ablate text and indexing and sweep path length, reporting average nodes visited per query. We do not claim a new agent/tool benchmark result.

The comparison is paired at the backbone–benchmark level: each panel uses the same model and task data, so differences isolate the retrieval condition rather than model scale. We report the benchmark scores directly and summarize them with arithmetic macro-averages; no benchmark is reweighted by its number of examples. Because the available runs yield one aggregate score per configuration, our analysis reports point differences and consistency across the eight paired settings, not confidence intervals or statistical significance. This distinction is important for a short-paper result: the experiments support comparative claims about the evaluated configurations, while broader claims about all GraphRAG systems require repeated runs and additional corpora.

Overall retrieval. Figure 2 shows a consistent aggregate ordering. Macro-average MultiHop/RGB scores are $61.23/46.66$ for Base, $68.42/65.47$ for ES, $73.41/70.40$ for LightRAG, and $88.09/80.45$ for StarGraph. The full representation gains $26.87/33.80$ points over Base and $19.68/14.99$ over ES, so

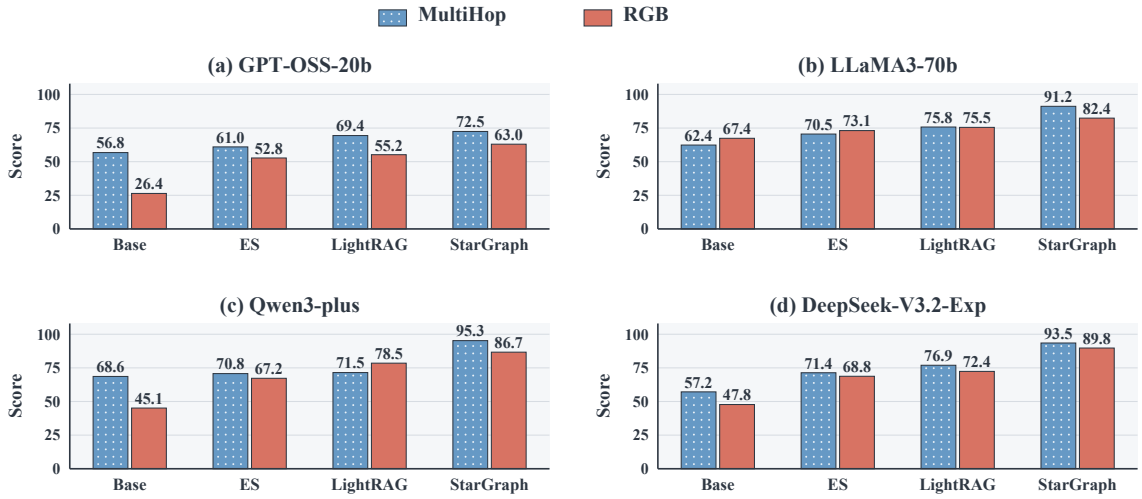


Figure 2: RAG-specific scores across four backbones. Each panel compares Base, ES, LightRAG, and StarGraph on MultiHop (blue) and RGB (orange).

Table 2
Two-factor representation ablation on Qwen3-plus. Δ is measured from the plain graph. The interaction is full $-$ index-only $-$ text-only $+$ plain.

Variant	Text	Index	MultiHop	Δ	RGB	Δ
Plain graph	no	no	70.12	–	41.85	–
Index only	no	yes	82.84	+12.72	66.92	+25.07
Text only	yes	no	75.12	+5.00	57.24	+15.39
Full StarGraph	yes	yes	**95.26**	**+25.14**	**86.68**	**+44.83**
Interaction	–	–	–	+7.42	–	+4.37

retrieval alone does not explain the result. StarGraph wins all eight backbone–task pairs; over LightRAG its mean gains are $14.68/10.06$, peaking at 23.74 on Qwen3-plus MultiHop and 17.37 on DeepSeek RGB. These are descriptive paired differences, not a universal GraphRAG ranking [15].

ES improves over Base by $7.19/18.81$ MultiHop/RGB points, LightRAG adds $4.99/4.93$ over ES, and StarGraph adds $14.68/10.06$ over LightRAG. Gains remain positive across all four backbones, so the effect is not confined to one model family.

Component analysis. Table 2 isolates text and indexing on Qwen3-plus. From the plain graph, indexes add $12.72/25.07$ MultiHop/RGB points and text adds $5.00/15.39$; the full representation adds $25.14/44.83$. Its positive interaction ($+7.42/+4.37$) supports complementarity: meta-paths constrain higher-order routes while retained text supplies readable evidence. The especially large index-only RGB gain is consistent with filtering distractors. Importantly, the interaction is computed from all four measured cells rather than inferred from the overall comparison. It quantifies the portion of the full gain not explained by simply adding the two isolated main effects, although it should not be interpreted as a causal estimate beyond this backbone and setup.

Backbone transfer. Relative to LightRAG, MultiHop/RGB gains are $+3.03/+7.80$ for GPT-OSS, $+15.39/+6.88$ for LLaMA3, $+23.74/+8.18$ for Qwen3-plus, and $+16.55/+17.37$ for DeepSeek. All are positive but vary by task and model; the largest gains occur on Qwen3-plus MultiHop and DeepSeek RGB, rather than only on the weakest Base configuration.

Path-length sensitivity. Figure 3 macro-averages 68.02, 76.94, 78.94, 79.68, and 80.45 RGB points for $l = 1 \ldots 5$. The first extension adds 8.92 points, but the two after $l = 3$ add only 1.51 total. Full $l = 1$ to 5 gains are 26.62 for GPT-OSS, 9.88 for LLaMA3, 7.47 for Qwen3-plus, and 5.75 for DeepSeek, so stronger backbones saturate earlier. Cost diverges after $l = 2$: index/online visits are $15/79$ at $l = 3$, $40/243$ at $l = 4$, and $88/899$ at $l = 5$, reductions of $5.27\times$, $6.08\times$, and $10.22\times$. Hence $l = 3$ is the

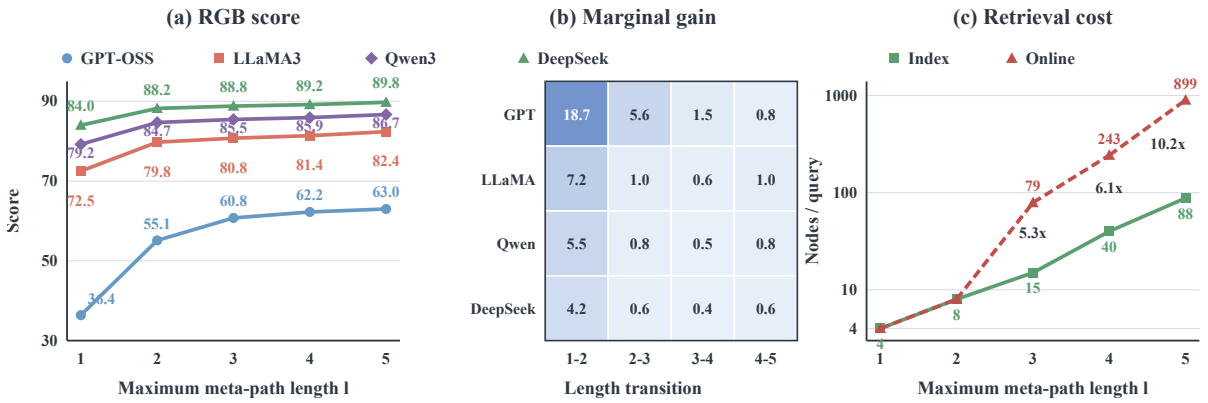


Figure 3: Meta-path sensitivity: (a) measured RGB scores across four backbones, (b) stepwise gains, and (c) retrieval cost.

empirical quality–cost knee, matching the rationale for materializing short typed paths [7].

Marginal gains diminish after $l = 3$: the final two extensions add only 1.51 points on average, while indexed access exposes a tunable quality–cost trade-off. Longer paths may still benefit weaker models or denser corpora.

7. Limitations and Design Trade-Offs

StarGraph shifts work from runtime to indexing, which suits reused corpora, tool sets, and task environments. Extra storage for text, tool/action records, and node indexes avoids unconstrained online search, while new records refresh only affected neighborhoods. The approach is less attractive for one-off corpora or graphs whose edges change faster than their indexes can be refreshed. Likewise, increasing l expands both the signature vocabulary and the number of path instances; the measured node-visit reduction captures online traversal work but is not a complete accounting of construction time, storage, selector latency, or prompt tokens.

Representation errors can also propagate. Missed entities prevent useful provenance links, entity-resolution errors merge unrelated evidence, and incorrect relation types admit invalid meta-path instances. Preserving source segments makes these failures inspectable, but does not automatically correct them. The current evaluation isolates retrieval and representation effects on two RAG benchmarks and four backbones; it does not yet measure end-to-end agent success, tool-call validity, update throughput, or behavior under temporal graph drift. Future evaluation should include multiple seeds where decoding is stochastic, retrieval recall and evidence attribution, index build/storage measurements, and controlled corruption of entities and edges.

8. Conclusion

StarGraph combines readable evidence, typed relations, tool/action constraints, and precomputed higher-order access. Across four backbones it improves MultiHop and RGB over LightRAG by 14.68 and 10.06 points on average and cuts length-five traversal from 899 to 88 nodes per query.

Acknowledgments

This work was supported by the Special Project for Discipline Construction of Peking University.

Declaration on Generative AI

During the preparation of this work, the authors used OpenAI Codex in order to: Peer review simulation and Formatting assistance. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] P. Lewis, E. Perez, A. Piktus, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, Advances in Neural Information Processing Systems (NeurIPS) 33 (2020) 9459–9474.

[2] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2020, pp. 6769–6781. doi:10.18653/v1/2020.emnlp-main.550.

[3] H. Li, Y. Su, D. Cai, et al., A survey on retrieval-augmented text generation, arXiv:2202.01110, 2022.

[4] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37.

[5] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert systems with applications 141 (2020) 112948.

[6] S. Pan, L. Luo, Y. Wang, et al., Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering (TKDE) 36 (2024) 3580–3599. doi:10.1109/TKDE.2024.3352100.

[7] Y. Sun, J. Han, X. Yan, P. S. Yu, T. Wu, Pathsim: Meta path-based top-k similarity search in heterogeneous information networks, Proceedings of the VLDB Endowment 4 (2011) 992–1003.

[8] Y. Sun, J. Han, Mining heterogeneous information networks: principles and methodologies, Morgan & Claypool Publishers, 2012.

[9] Z. Jiang, F. Xu, L. Gao, et al., Active retrieval augmented generation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), 2023, pp. 7969–7992.

[10] A. Asai, Z. Wu, Y. Wang, A. Sil, H. Hajishirzi, Self-RAG: Learning to retrieve, generate, and critique through self-reflection, in: International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=hSyW5go0v8.

[11] D. Edge, H. Trinh, N. Cheng, et al., From local to global: A graph RAG approach to query-focused summarization, arXiv:2404.16130, 2024.

[12] Z. Guo, L. Xia, Y. Yu, et al., LightRAG: Simple and fast retrieval-augmented generation, arXiv:2410.05779, 2024.

[13] B. Jimenez Gutierrez, Y. Shu, Y. Gu, M. Yasunaga, Y. Su, Hipporag: Neurobiologically inspired long-term memory for large language models, Advances in Neural Information Processing Systems 37 (2024) 59532–59569.

[14] A. Salemi, H. Zamani, Evaluating retrieval quality in retrieval-augmented generation, in: International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2024, pp. 2395–2400.

[15] H. Han, H. Shomer, Y. Wang, Y. Lei, K. Guo, Z. Hua, B. Long, H. Liu, J. Tang, RAG vs. GraphRAG: A systematic evaluation and key insights, 2025. arXiv:2502.11371.

[16] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023.

[17] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, T. Scialom, Toolformer: Language models can teach themselves to use tools, in: Advances in Neural Information Processing Systems (NeurIPS), 2023.

[18] S. G. Patil, T. Zhang, X. Wang, J. E. Gonzalez, Gorilla: Large language model connected with massive APIs, arXiv:2305.15334, 2023.

[19] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, A. Anandkumar, Voyager: An open-ended embodied agent with large language models, arXiv:2305.16291, 2023.

[20] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, S. Yao, Reflexion: Language agents with verbal reinforcement learning, in: Advances in Neural Information Processing Systems (NeurIPS), 2023.

[21] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, M. S. Bernstein, Generative agents: Interactive simulacra of human behavior, in: ACM Symposium on User Interface Software and Technology (UIST), 2023.

[22] B. Zhang, H. Soh, Extract, define, canonicalize: An LLM-based framework for knowledge graph construction, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Miami, Florida, USA, 2024, pp. 9820–9836.

[23] M. Yasunaga, H. Ren, A. Bosselut, et al., QA-GNN: Reasoning with language models and knowledge graphs for question answering, in: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), 2021, pp. 535–546.

[24] 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).

[25] X. Tan, X. Wang, Q. Liu, X. Xu, X. Yuan, W. Zhang, Paths-over-graph: Knowledge graph empowered large language model reasoning, in: Proceedings of the ACM on Web Conference 2025, 2025, pp. 3505–3522.

[26] Y. Tang, Y. Yang, MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries, arXiv:2401.15391, 2024.

[27] J. Chen, H. Lin, X. Han, L. Sun, Benchmarking large language models in retrieval-augmented generation, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024) 17754–17762.

[28] Elastic, Elasticsearch reference, Software documentation, 2026. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/, accessed 2026-07-22.

[29] OpenAI, gpt-oss-120b & gpt-oss-20b model card, 2025. arXiv:2508.10925.

[30] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, et al., The Llama 3 herd of models, 2024. arXiv:2407.21783.

[31] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, et al., Qwen3 technical report, 2025. arXiv:2505.09388.

[32] DeepSeek-AI, DeepSeek-V3.2-Exp: Boosting long-context efficiency with DeepSeek Sparse Attention, Official model release and technical report, 2025. URL: https://github.com/deepseek-ai/DeepSeek-V3.2-Exp.