The Journal of Open Source Software

Nexarag: Democratizing Reproducible Knowledge Graph Contexts for LLM Research

Thomas J. Kerby[1], **Benjamin N. Fuller**[2], **and Kevin R. Moon**[3]

1 Brigham Young University, Provo, UT **2** Independent Researcher **3** Utah State University, Logan, UT

DOI: 10.21105/joss.09910

Software

- Review
- Repository
- Archive

Editor: Wentao Ye

Reviewers:

- @pebabion
- @gsquared94
- @r0hin

Submitted: 01 December 2025
Published: 25 April 2026

Summary

Large language models (LLMs) are widely used in research workflows but struggle with hallucinations, short context windows, and weak reproducibility in literature reviews (Huang et al., 2025; Ji et al., 2023). Nexarag is a modular, open-source platform that lets researchers curate, visualize, and share custom knowledge graphs (KGs) from academic sources stored in Neo4j (Neo4j Inc., 2024). Through native support for the Model Context Protocol (MCP), any MCP-compatible LLM can access these curated KGs for controllable, reproducible context injection (Anthropic, 2024; Model Context Protocol Contributors, 2024)—including fully private, air-gapped deployments via containers (Boettiger, 2015)—so research teams can explore relevant literature more deeply and transparently. Nexarag provides interactive graph/semantic visualizations using Cytoscape.js and D3 (Bostock et al., 2011; Franz et al., 2023).

Statement of need

Retrieval-augmented generation (RAG) has become a standard approach for knowledge-intensive NLP (Gao et al., 2023; Guu et al., 2020; Lewis et al., 2020), but systems built primarily on embedding-based similarity (Guu et al., 2020; Lewis et al., 2020; Reimers & Gurevych, 2019) can miss long-range semantic structure and cross-document relationships, especially in long-context and multi-document settings (Gao et al., 2023; Wang et al., 2024). For literature synthesis and related research workflows, this limits transparency, controllability, and reproducibility.

Knowledge graphs address part of this problem by representing entities and relations explicitly, enabling path-based queries and more interpretable reasoning over document collections (Reinanda et al., 2020; Sahlab et al., 2022; Xu et al., 2024). However, existing KG-based tooling is often either proprietary or too technically demanding for routine research use. Nexarag addresses this gap with a researcher-friendly, self-hostable platform for constructing and curating literature knowledge graphs and exposing them to language models through MCP (Anthropic, 2024).

State of the field

Open-source RAG frameworks such as LangChain, LlamaIndex, and Haystack provide reusable components for ingestion, indexing, and retrieve-then-generate pipelines (deepset Haystack, 2026; LangChain, 2026a; LlamaIndex, 2026a). They are effective developer libraries, but they are mainly designed for assembling application-specific pipelines in code.

Graph-augmented retrieval tools extend this ecosystem. LlamaIndex includes a KnowledgeGraphIndex (LlamaIndex, 2026b), LangChain provides graph QA utilities such as GraphCypherQAChain (LangChain, 2026b), and packages such as Microsoft GraphRAG

and Neo4j GraphRAG support structured extraction and graph-aware retrieval (Microsoft, 2026; Neo4j, 2026). These projects show the value of graph-based retrieval, but they are generally aimed at developers rather than at reproducible, literature-centered research workflows.

Literature discovery platforms such as Connected Papers, ResearchRabbit, and Litmaps support citation exploration and related-work discovery (Connected Papers, 2026; Litmaps, 2026; ResearchRabbit, 2026), but they do not provide a researcher-owned, versionable graph substrate for controlled LLM experiments. Nexarag sits between these tool families by packaging persistent Neo4j-based graph construction, interactive curation, and standardized model access through MCP into a self-hostable research application (Anthropic, 2024; Model Context Protocol Contributors, 2024). It complements existing RAG and GraphRAG libraries by turning graph-based context construction into a reusable and shareable research workflow with user-friendly UI tools, visualization, and pluggable MCP integration.

Software design

Nexarag was designed around four principles: ease of use, flexibility, modularity, and privacy/security. The system uses a containerized, microservices design orchestrated with Docker Compose (Docker, Inc., 2024; Merkel, 2014). Primary services include: a FastAPI service for HTTP coordination (FastAPI Contributors, 2024), a Neo4j database for graph storage (Neo4j Inc., 2024), and a Knowledge Graph service for document processing/embeddings/AI tasks. Services communicate asynchronously via RabbitMQ, enabling horizontal scaling (VMware, Inc., 2024).

Design choices in Nexarag	Tradeoffs
Ease of use: Nexarag uses familiar frontend technologies, including Angular, D3.js, Cytoscape.js, and PrimeNg, to provide an intuitive interface for building knowledge graphs, finding papers, interacting with LLMs, and visualizing results. Nexarag is fully containerized and can be deployed with a single command.	Multiple component libraries increases frontend maintenance overhead and onboarding cost for new contributors. Container deployments add complexity and additional software dependencies (Docker).
Flexibility: Nexarag integrates with Ollama and supports any embedding model or LLM that the user's hardware can run, making it easy to switch models across tasks or adopt new ones as they become available. Users can also connect their preferred LLM or coding agent through the built-in MCP server.	Users have more choices but also more responsibility for hardware configuration, model selection, and staying up-to-date on relevant tools and architectures.
Modularity: The system is organized as distinct services for the REST API, Neo4j knowledge graph, MCP server, and frontend application, connected through a RabbitMQ messaging backbone. This supports horizontal scaling and reduces the blast radius of changes made within any single service.	Service decomposition improves scalability and isolates changes, but increases deployment complexity, inter-service coordination, and operational overhead.
Privacy and security: Nexarag supports on-premises, air-gapped deployment, providing a level of privacy and security that cloud-based applications typically cannot offer.	Air-gapped deployments can offer heightened security and privacy, but place more burden on the user for hardware configuration, deployment, and maintenance. Local compute resources may also be limited compared to cloud services.

Nexarag brings these design choices together in a research platform that offers:

- Automated KG construction from BibTeX, paper lists, search queries, and citation expansion (Semantic Scholar integration) (Kinney et al., 2023; Semantic Scholar, 2024).
- Interactive graph and semantic visualizations (Cytoscape.js and D3.js) (Bostock et al., 2011; Franz et al., 2023).
- An AI "Talk To Your Data" interface that supports both simple retrieve-and-generate and ReAct-style agentic workflows (Yao et al., 2022).
- An MCP-compatible server that exposes graph querying, semantic search over embedded content, and external search via Semantic Scholar to any MCP-enabled LLM (local via Ollama or remote via hosted providers) (Anthropic, 2024; Model Context Protocol Contributors, 2024; Ollama Team, 2024; OpenAI, 2026).
- Neo4j-backed storage and Cypher querying (Neo4j Inc., 2024).
- Tools to share/export/import graphs across teams to support longitudinal projects.
- Entirely offline (air-gapped) deployment with local LLMs for sensitive domains (e.g., healthcare, legal, proprietary research) (Boettiger, 2015).

Research impact statement

Nexarag addresses a growing need in LLM-assisted research for transparent, reproducible, and inspectable context construction beyond embedding-only retrieval by operationalizing knowledge-graph–based context building in a locally deployable, visually inspectable form that can be shared across projects while lowering the technical barrier to constructing, inspecting, and reusing reproducible graph-based contexts. By combining Neo4j-backed knowledge graphs with standardized model access through the Model Context Protocol (MCP), it creates a reproducible bridge between structured scholarly knowledge and LLM-driven analysis while remaining model-agnostic, allowing researchers to interchange local or API-hosted LLMs without changing the underlying knowledge graph or retrieval logic. This makes Nexarag well suited as shared research infrastructure for retrieval-augmented generation, knowledge-graph–augmented reasoning, and AI-assisted literature review, enabling direct comparison of model behavior under identical graph-derived contexts and facilitating methodological work on controllability, hallucination reduction, and long-context reasoning.

AI usage disclosure

Generative AI tools were used in the development of the software, supporting code reviews, providing minor features in the frontend, and identifying and fixing bugs. Generative AI tools were also used to generate some of the documentation, and assisted with paper authoring. We primarily used:

- ChatGPT with the GPT-4o model for writing tasks.
- Claude Code with the Sonnet 4 model for coding tasks.

All AI-generated material was explicitly reviewed by at least one author, and all major design decisions were formalized by multiple authors.

Acknowledgements

We acknowledge the open-source ecosystems behind Neo4j, Cytoscape.js, D3.js, RabbitMQ, FastAPI, Ollama, and the Model Context Protocol, as well as contributors and users who provided feedback during development. This research was supported in part by the NSF under Grant 2212325.

References

Anthropic. (2024). *Introducing the model context protocol*. https://www.anthropic.com/news/model-context-protocol

Boettiger, C. (2015). An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, *49*(1), 71–79. https://doi.org/10.1145/2723872.2723882

Bostock, M., Ogievetsky, V., & Heer, J. (2011). D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, *17*(12), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

Connected Papers. (2026). *Connected papers*. https://www.connectedpapers.com/

deepset Haystack. (2026). *Get started (Haystack documentation)*. https://docs.haystack.deepset.ai/docs/get-started

Docker, Inc. (2024). *Docker: Accelerated container application development*. https://www.docker.com/

FastAPI Contributors. (2024). *FastAPI: Modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints*. https://fastapi.tiangolo.com/

Franz, M., Lopes, C. T., Fong, D., Kucera, M., Cheung, M., Siper, M. C., Huck, G., Dong, Y., Sumer, O., & Bader, G. D. (2023). Cytoscape.js 2023 update: A graph theory library for visualization and analysis. *Bioinformatics*, *39*(1), btad031. https://doi.org/10.1093/bioinformatics/btad031

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv Preprint arXiv:2312.10997*, *2*.

Guu, K., Lee, K., Tung, Z., Pasupat, P., & Chang, M.-W. (2020). REALM: Retrieval-augmented language model pre-training. *Proceedings of the 37th International Conference on Machine Learning*.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., & Liu, T. (2025). A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, *43*(2). https://doi.org/10.1145/3703155

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Comput. Surv.*, *55*(12). https://doi.org/10.1145/3571730

Kinney, R. M., Anastasiades, C., Authur, R., Beltagy, I., Bragg, J., Buraczynski, A., Cachola, I., Candra, S., Chandrasekhar, Y., Cohan, A., Crawford, M., Downey, D., Dunkelberger, J., Etzioni, O., Evans, R., Feldman, S., Gorney, J., Graham, D. W., Hu, F. Q., … Weld, D. S. (2023). The semantic scholar open data platform. *arXiv*, *2301.10140*. https://doi.org/10.48550/arXiv.2301.10140

LangChain. (2026a). *Build a RAG agent with LangChain*. https://docs.langchain.com/oss/python/langchain/rag

LangChain. (2026b). *Neo4j (LangChain integration documentation)*. https://docs.langchain.com/oss/python/integrations/providers/neo4j

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Proceedings of the 34th International Conference on Neural Information Processing Systems*.

Litmaps. (2026). *Litmaps*. https://www.litmaps.com/

LlamaIndex. (2026a). *Introduction to RAG (retrieval-augmented generation)*. https://developers.llamaindex.ai/python/framework/understanding/rag/

LlamaIndex. (2026b). *Knowledge graph index (LlamaIndex python documentation)*. https://developers.llamaindex.ai/python/examples/index_structs/knowledge_graph/knowledgegraphdemo/

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Microsoft. (2026). *Microsoft/graphrag: A modular graph-based retrieval-augmented generation (RAG) system*. https://github.com/microsoft/graphrag

Model Context Protocol Contributors. (2024). *Model context protocol: A protocol for seamless integration between LLM applications and external data sources*. https://github.com/modelcontextprotocol

Neo4j. (2026). *neo4j/neo4j-graphrag-python: Neo4j GraphRAG package for Python*. https://github.com/neo4j/neo4j-graphrag-python

Neo4j Inc. (2024). *Neo4j graph database platform*. https://neo4j.com/product/neo4j-graph-database/

Ollama Team. (2024). *Ollama: Get up and running with Llama 3.2, Mistral, Gemma 2, and other large language models*. https://ollama.com/

OpenAI. (2026). *OpenAI API documentation*. https://developers.openai.com/api/docs

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 3973–3983. https://doi.org/10.18653/v1/D19-1410

Reinanda, R., Meij, E., & Rijke, M. de. (2020). Knowledge graphs: An information retrieval perspective. *Foundations and Trends in Information Retrieval*, *14*(2-3), 289–444. https://doi.org/10.1561/1500000063

ResearchRabbit. (2026). *ResearchRabbit*. https://www.researchrabbit.ai/

Sahlab, N., Kahoul, H., Jazdi, N., & Weyrich, M. (2022). A knowledge graph-based method for automating systematic literature reviews. *arXiv Preprint arXiv:2208.02334*. https://doi.org/10.48550/arXiv.2208.02334

Semantic Scholar. (2024). *Semantic scholar academic graph API*. https://www.semanticscholar.org/product/api

VMware, Inc. (2024). *RabbitMQ: The most widely deployed open source message broker*. https://www.rabbitmq.com/

Wang, M., Chen, L., Cheng, F., Liao, S., Zhang, X., Wu, B., Yu, H., Xu, N., Zhang, L., Luo, R., Li, Y., Yang, M., Huang, F., & Li, Y. (2024). Leave no document behind: Benchmarking long-context LLMs with extended multi-doc QA. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 5627–5646. https://doi.org/10.18653/v1/2024.emnlp-main.322

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., & Li, Z. (2024). Retrieval-augmented generation with knowledge graphs for customer service question answering. *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2905–2909. https://doi.org/10.1145/3626772.3661370

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. *arXiv Preprint arXiv:2210.03629*.

https://doi.org/10.48550/arXiv.2210.03629