The Journal of Open Source Software

Nexarag: Democratizing Reproducible Knowledge Graph Contexts for LLM Research

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

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

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Wentao Ye

Reviewers:

- @pebabion
- @aslan-ng
- @gsquared94
- @r0hin

Submitted: 01 December 2025
Published: unpublished

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 teams can explore 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

Traditional retrieval-augmented generation (RAG) systems rely primarily on embedding-based similarity (Guu et al., 2020; Lewis et al., 2020; Reimers & Gurevych, 2019), which often misses long-range semantic structure and relationships across documents, especially in long-context, multi-document settings (Gao et al., 2023; Wang et al., 2024). This weakens controllability, auditability, and reproducibility for complex research tasks. Knowledge graphs provide a structured and interpretable alternative by modeling entities and relations explicitly (Reinanda et al., 2020). However, existing KG-powered tooling is either proprietary and expensive or technically demanding to deploy and maintain. Nexarag fills this gap with a researcher-friendly platform that automates KG creation from academic inputs, supports semantic exploration, and standardizes LLM access via MCP (Anthropic, 2024), enabling reproducible literature synthesis across local and cloud settings.

RAG improves access to external knowledge and has become a standard strategy for knowledge-intensive NLP (Gao et al., 2023; Guu et al., 2020; Lewis et al., 2020), yet it struggles with long contexts and multi-step reasoning when similarity search is the only primitive (Wang et al., 2024). KGs address this by enabling path-based queries and explicit relation reasoning while preserving transparency and updatability (Reinanda et al., 2020; Sahlab et al., 2022; Xu et al., 2024). Nexarag's contribution is to operationalize these advantages in a package that researchers can run locally, share with collaborators, and connect to a wide range of LLM hosts through MCP (Anthropic, 2024).

State of the field

The open-source ecosystem around retrieval-augmented generation (RAG) has matured quickly, with widely used orchestration frameworks such as LangChain, LlamaIndex, and Haystack providing reusable components for ingesting documents, building indexes, and implementing retrieve-then-generate pipelines (deepset Haystack, n.d.; LangChain, n.d.-a; LlamaIndex, n.d.-a). These frameworks are well suited for building bespoke RAG applications, but they are primarily developer libraries: users typically assemble pipelines in code, and the resulting "context construction" logic is often tightly coupled to a specific project's embeddings, chunking choices, runtime configuration, and LLM host.

Graph-augmented retrieval approaches have emerged to address limitations of similarity-only retrieval for multi-document reasoning. For example, LlamaIndex provides a KnowledgeGraphIndex that supports automated knowledge graph construction from unstructured text and entity-based querying (LlamaIndex, n.d.-b). LangChain offers graph question-answering utilities, including GraphCypherQAChain for translating natural-language questions into Cypher queries over Neo4j (LangChain, n.d.-b). Dedicated GraphRAG toolkits such as Microsoft GraphRAG and the Neo4j GraphRAG package for Python provide pipelines for extracting structured representations from text and using graph-aware retrieval strategies during LLM inference (Microsoft, n.d.; Neo4j, n.d.). These projects demonstrate that knowledge graphs can improve retrieval structure and interpretability, but they are generally positioned as building blocks for developers rather than as reproducible, end-to-end research applications for literature-centric workflows.

In parallel, literature discovery tools such as Connected Papers, ResearchRabbit, and Litmaps offer citation-network visualizations and recommendations to help researchers explore related work and track research areas over time (Connected Papers, n.d.; Litmaps, n.d.; ResearchRabbit, n.d.). While valuable for interactive discovery, these tools are not designed to serve as a researcher-owned, queryable knowledge graph substrate that can be versioned, shared, and integrated as a controllable context source for LLM experiments across local and secure environments.

Nexarag fills the gap between these two tool families by packaging knowledge-graph-based context construction as a reproducible, shareable research artifact and workflow. Instead of treating graph retrieval as a code-level feature, Nexarag provides a self-hostable platform centered on a persistent Neo4j knowledge graph, interactive graph curation and visualization, and standardized LLM access through the Model Context Protocol (MCP) (Anthropic, 2024; Model Context Protocol Contributors, 2024). This design enables researchers to hold the underlying graph context and retrieval tools fixed while varying models, prompts, and deployment settings, supporting more transparent and reproducible LLM-assisted literature synthesis.

We chose to build Nexarag rather than contribute directly to an existing RAG or GraphRAG library because our core contribution requires application-level infrastructure that is outside the scope of most libraries' design constraints: a literature-oriented ingestion workflow (BibTeX and paper lists, citation expansion, and persistent graph storage), a collaborative and visual curation interface, containerized offline deployment, and an MCP-first tool surface for model-agnostic experimentation. Nexarag therefore complements, rather than replaces, existing RAG frameworks by providing a researcher-facing system for producing and reusing reproducible knowledge graph contexts in LLM research.

Software Design

In designing Nexarag we focused on four principles: (1) ease-of-use (in deployment and practical application), (2) flexibility (in model selection and configuration), (3) modularity (to promote scale and independent contribution), and (4) privacy and security. For ease-of-use, we chose popular frontend technologies such as Angular, D3.js, and Cytoscape.js and leveraged existing component libraries to provide a simple and familiar frontend user interface with intuitive tools for building knowledge graphs, searching for relevant papers, conversing with LLMs, and visualizing data. Nexarag is also fully containerized, with release artifacts produced by automated build pipelines readily available for local deployment through Docker. For flexibility, we integrate with Ollama and support any embedding model and LLM that is also supported by the user's hardware, making it simple to switch between models for different tasks and as new models are released. Users can also plug in their preferred LLM or coding agent of choice using the built-in MCP server. Additionally, Nexarag features a highly modular design, with distinct services for the REST API, the neo4j knowledge graph, the MCP server, and the frontend application, all bound together with a RabbitMQ messaging backbone. This allows components to scale horizontally, and to minimize the blast radius of contributions in a single service. Finally, Nexarag supports on-premises, air-gapped deployments, offering privacy and security that is not available in cloud-based applications.

Software overview

Core capabilities. Nexarag provides: (i) automated KG construction from BibTeX, paper lists, search queries, and citation expansion (Semantic Scholar integration) (Kinney et al., 2023; Semantic Scholar, 2024); (ii) Neo4j-backed storage and Cypher querying (Neo4j Inc., 2024); (iii) interactive graph and semantic visualizations (Cytoscape.js and D3.js) (Bostock et al., 2011; Franz et al., 2023); and (iv) an AI "Talk To Your Data" interface that supports both simple retrieve-and-generate and ReAct-style agentic workflows (Yao et al., 2022).

Architecture. 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).

MCP integration. Nexarag ships 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, 2023). This standardizes context delivery and promotes reproducible prompt-driven research workflows.

Install & minimal run. (see repository docs for full instructions)

```
# CPU example
docker compose -f docker-compose.cpu.yml up -d
# or on macOS
docker compose -f docker-compose.macos.yml up -d
```

Optionally pull local models for embedding/LLM integration with Ollama (Ollama Team, 2024); for example, a long-context text embedder like Nomic Embed (Nussbaum et al., 2025):

```
# inside the Ollama container or on macOS host
ollama pull nomic-embed-text:v1.5
ollama pull gemma3:1b
```

Repository: https://github.com/REPLACE-WITH-YOUR-ORG/nexarag

License: GNU General Public License v3.0.

Use cases

- **Reproducible literature reviews.** Build a KG from a seed set (e.g., via BibTeX), expand by citations, and generate a structured review through the MCP interface (Sahlab et al., 2022).
- **Private research contexts.** Run entirely offline (air-gapped) with local LLMs for sensitive domains (e.g., healthcare, legal, proprietary research) (Boettiger, 2015).

- **Collaborative curation.** Share/export/import graphs across teams to support longitudinal projects.

Quality control

Nexarag emphasizes verifiable operation through containerized deployment and a guided quick start (Boettiger, 2015). Reviewers can launch the full stack with Docker Compose, query/persist KGs in Neo4j, and exercise end-to-end flows (semantic search, citation expansion, MCP tools). A worked MCP chat transcript and an automatically generated literature review illustrate that the system's graph building, retrieval, and reporting features execute as described. The repository includes example datasets/notebooks and scripts for running tests where applicable, supporting broader reproducibility goals in research practice (Rothacher et al., 2023).

Research impact statement

Nexarag addresses a growing need in LLM-assisted research for transparent, reproducible, and inspectable context construction beyond embedding-only retrieval. While many RAG systems remain opaque and difficult to reproduce, Nexarag operationalizes knowledge-graph–based context building in a form that researchers can deploy locally, inspect visually, and share across projects. By combining Neo4j-backed knowledge graphs with standardized access through the Model Context Protocol (MCP), the software provides a reproducible bridge between structured scholarly knowledge and LLM-driven analysis.

Although Nexarag is a relatively new project and has not yet accumulated extensive downstream citations, it demonstrates credible near-term research impact through its design, documentation, and reproducible reference materials. The repository includes end-to-end examples that reproduce literature expansion, graph construction, semantic querying, and LLM-mediated synthesis from fixed inputs, allowing independent researchers to verify behavior and compare results across models and deployment environments. Containerized deployment and air-gapped operation further support use in domains where reproducibility, auditability, or data sensitivity are critical.

Nexarag is positioned to serve as shared research infrastructure for studies on retrieval-augmented generation, knowledge-graph–augmented reasoning, and AI-assisted literature review workflows. Its model-agnostic design, enabled by MCP, allows researchers to interchange local or API-hosted LLMs while holding the underlying knowledge graph and retrieval logic fixed. This supports a direct comparison of LLM behavior under identical, graph-derived contexts, facilitating methodological research on controllability, hallucination reduction, and long-context reasoning. By lowering the technical barrier to building, inspecting, and sharing reproducible knowledge graph contexts, Nexarag enables researchers to move beyond ad hoc, model-coupled RAG pipelines toward more transparent and portable AI-assisted research practices.

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 221235.

References

Anthropic. (2024). *Introducing the model context protocol*. https://www.anthropic.com/news/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. (n.d.). *Connected papers*. https://www.connectedpapers.com/.

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

Docker, Inc. (2024). *Docker: Accelerated container application development*. https://www.docker.com/. 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/. 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*, *abs/2301.10140*. https://api.semanticscholar.org/CorpusID:256194545

LangChain. (n.d.-a). *Build a RAG agent with LangChain*. https://docs.langchain.com/oss/

python/langchain/rag.

LangChain. (n.d.-b). *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. (n.d.). *Litmaps*. https://www.litmaps.com/.

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

LlamaIndex. (n.d.-b). *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. (n.d.). *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. https://github.com/modelcontextprotocol

Neo4j. (n.d.). *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/. https://neo4j.com/product/neo4j-graph-database/

Nussbaum, Z., Morris, J. X., Mulyar, A., & Duderstadt, B. (2025). Nomic embed: Training a reproducible long context text embedder. *Transactions on Machine Learning Research*. https://openreview.net/forum?id=IPmzyQSiQE

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

OpenAI. (2023). *OpenAI API*.

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. http://arxiv.org/abs/1908.10084

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. (n.d.). *ResearchRabbit*. https://www.researchrabbit.ai/.

Rothacher, L., Engel, C., Garbeva, P., Grüning, B., Goble, C., Gundersen, S., Ioannidis, J. P., Marwick, B., Parsons, S., Pronk, T. E., & others. (2023). Eleven strategies for making reproducible research and open science training the norm at research institutions. *eLife*, *12*, RP89736. https://doi.org/10.7554/eLife.89736

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.

The Journal of Open Source Software

semanticscholar.org/product/api. https://www.semanticscholar.org/product/api

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