The Journal of Open Source Software

ollamar: An R package for running large language models

Hause Lin [1] **and Tawab Safi** [1]

1 Massachusetts Institute of Technology, USA

DOI: 10.21105/joss.07211

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @KennethEnevoldsen
- @elenlefoll

Submitted: 24 August 2024
Published: 24 January 2025

Summary

Large language models (LLMs) have transformed natural language processing and AI applications across numerous domains. While cloud-based LLMs are common, locally deployed models offer distinct advantages in reproducibility, data privacy, security, and customization. `ollamar` is an R package that provides an interface to Ollama, enabling researchers and data scientists to integrate locally-hosted LLMs into their R workflows seamlessly. It implements a consistent API design that aligns with other programming languages and follows established LLM usage conventions. It further distinguishes itself by offering flexible output formats and easy management of conversation history. `ollamar` is maintained on GitHub and available through the Comprehensive R Archive Network (CRAN), where it regularly undergoes comprehensive continuous integration testing across multiple platforms.

State of the Field

The increasing importance of LLMs in various fields has created a demand for accessible tools that allow researchers and practitioners to leverage LLMs within their preferred programming environments. Locally deployed LLMs offer advantages in terms of data privacy, security, reproducibility, and customization, making them an attractive option for many users (Chan et al., 2024; Liu et al., 2024; Lytvyn, 2024; Shostack, 2024). Currently, Ollama (https://ollama.com/) is one of the most popular tools for running locally hosted LLMs, offering access to a range of models with different sizes and capabilities. Several R packages currently facilitate interaction with locally deployed LLMs through Ollama, each with distinct approaches, capabilities, and limitations.

The `rollama` (Gruber & Weber, 2024) and `tidyllm` (Brüll, 2024) libraries focus on text generation, conversations, and text embedding, but their core functions do not always or necessarily mirror the official Ollama API endpoints, which lead to inconsistencies and confusion for users familiar with the official API. Additionally, these libraries may not support all Ollama endpoints and features. Another popular R library is `tidychatmodels` (Albert, 2024), which allows users to chat with different LLMs, but it is not available on CRAN, and therefore is not subject to the same level of testing and quality assurance as CRAN packages.

All these libraries also adopt the tidyverse workflow, which some may find restrictive, opinionated, or unfamiliar. While the tidyverse is popular in the R community, it may not align with the programming style or workflow of all users, especially those coming from other programming languages or domains. This limitation can hinder the accessibility and usability of these libraries for a broader audience of R users. Thus, the R ecosystem lacks a simple and reliable library to interface with Ollama, even though R is a popular and crucial tool in statistics, data science, and various research domains (Hill et al., 2024).

Statement of Need

ollamar addresses the limitations of existing R libraries by providing a consistent API design that mirrors official Ollama endpoints, enabling seamless integration across programming environments. It also offers flexible output formats supporting dataframes, JSON lists, raw strings, and text vectors, allowing users to choose the format that best suits their needs. The package also includes independent conversation management tools that align with industry-standard chat formats, streamlining the integration of locally deployed LLMs into R workflows. These features make ollamar a versatile and user-friendly tool for researchers and data scientists working with LLMs in R. It fills a critical gap in the R ecosystem by providing a native interface to run locally deployed LLMs, and is already being used by researchers and practitioners (Turner, 2024).

Design

ollamar implements a modular, non-opinionated, and consistent approach that aligns with established software engineering principles, where breaking down complex systems into manageable components enhances maintainability, reusability, and overall performance. It also avoids feature bloat, ensuring the library remains focused on its core functionality. The key design philosophy of ollamar are described below.

Consistency and maintainability: It provides an interface to the Ollama server and all API endpoints, closely following the official API design, where each function corresponds to a specific endpoint. This implementation ensures the library is consistent with the official Ollama API and easy to maintain and extend as new features are added to Ollama. It also makes it easy for R users to understand how similar libraries (such as in Python and JavaScript) work while allowing users familiar with other programming languages to adapt to and use this library quickly. The consistent API structure across languages facilitates seamless transitions and knowledge transfer for developers working in multi-language environments.

Consistent and flexible output formats: All functions that call API endpoints return `httr2::httr2_response` objects by default, which provides a consistent interface for flexible downstream processing. If preferred, users have the option to specify different output formats when calling the endpoints, such as dataframes (`"df"`), lists (of JSON objects) (`"jsonlist"`), raw strings (`"raw"`), text vectors (`"text"`), or request objects (`"req"`). Alternatively, use the `resp_process()` function to convert and process the response object as needed. This flexibility allows users to choose the format that best suits their needs, such as when working with different data structures, integrating the output with other R packages, or allowing parallelization via the popular `httr2` library. It also allows users to easily build applications or pipelines on top of the library, without being constrained by a specific output format.

Easy management of LLM conversation history: LLM APIs often expect conversation/chat history data as input, often nested lists or JSON objects. Note that this data format is standard for chat-based applications and APIs (not limited to Ollama), such as those provided by OpenAI and Anthropic. ollamar simplifies preparing and processing conversational data for input to different LLMs, focusing on streamlining the workflow for the most popular chat-based applications.

Automated regular testing: ollamar is hosted on GitHub and available through CRAN, where it undergoes comprehensive continuous integration testing across multiple platforms to ensure reliability. Daily automated quality checks maintain long-term stability, and scheduled tests verify version compatibility.

Conclusion

`ollamar` bridges a crucial gap in the R ecosystem by providing seamless access to large language models through Ollama. Its focus on consistency, flexibility, and maintainability makes it a versatile and user-friendly tool for researchers and data scientists working with LLMs in R. These design choices ensure `ollamar` is a user-friendly and reliable tool for integrating locally deployed LLMs into R workflows, accelerating research and development in fields relying on R for data analysis and machine learning.

Acknowledgements

This project was partially supported by the Canadian Social Sciences & Humanities Research Council Tri-Agency Funding (funding reference: 192324).

References

Albert, R. (2024). *Tidychatmodels: Chat with all kinds of AI models through a common interface*. https://github.com/AlbertRapp/tidychatmodels

Brüll, E. (2024). Tidyllm: Tidy integration of large language models. *CRAN: Contributed Packages*. https://doi.org/10.32614/cran.package.tidyllm

Chan, R. S.-Y., Nanni, F., Brown, E., Chapman, E., Williams, A. R., Bright, J., & Gabasova, E. (2024). Prompto: An open source library for asynchronous querying of LLM endpoints. *arXiv*. https://doi.org/10.48550/arXiv.2408.11847

Gruber, J. B., & Weber, M. (2024). rollama: An R package for using generative large language models through Ollama. *arXiv*. https://doi.org/10.48550/arXiv.2404.07654

Hill, C., Du, L., Johnson, M., & McCullough, B. D. (2024). Comparing programming languages for data analytics: Accuracy of estimation in Python and R. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *14*(3), e1531. https://doi.org/10.1002/widm.1531

Liu, F., Kang, Z., & Han, X. (2024). Optimizing RAG techniques for automotive industry PDF chatbots: A case study with locally deployed Ollama models. *arXiv*. https://doi.org/10.48550/arXiv.2408.05933

Lytvyn, O. (2024). Enhancing propaganda detection with open source language models: A comparative study. *Proceedings of the MEi:CogSci Conference*, *18*(1). https://journals.phl.univie.ac.at/meicogsci/article/view/822

Shostack, A. (2024). The boy who survived: Removing Harry Potter from an LLM is harder than reported. *arXiv*. https://doi.org/10.48550/arXiv.2403.12082

Turner, S. D. (2024). biorecap: an R package for summarizing bioRxiv preprints with a local LLM. *arXiv*. https://doi.org/10.48550/arXiv.2408.11707