

# Enhancing the application of large language models with retrieval-augmented generation for a research community

Juan José García Mesa<sup>✉</sup>, Gil Speyer<sup>✉</sup>  
*Computational Research Accelerator*  
*Arizona State University*  
Tempe, Arizona, USA  
jgarc111@asu.edu, speyer@asu.edu

**Abstract**—The demand for efficient and innovative tools in research environments is ever-increasing in the rapidly evolving landscape of artificial intelligence (AI) and machine learning (ML). This paper explores the implementation of retrieval-augmented generation (RAG) to enhance the contextual accuracy and applicability of large language models (LLMs) to meet the diverse needs of researchers. By integrating RAG, we address various tasks such as synthesizing extensive questionnaire data, efficiently searching through document collections, and extracting detailed information from multiple sources. Our implementation leverages open-source libraries, a centralized repository of pre-trained models, and high-performance computing resources to provide researchers with robust, private, and scalable solutions.

**Index Terms**—Large Language Models, Retrieval-Augmented Generation

## I. INTRODUCTION

In the epoch of the artificial intelligence (AI) revolution, ever-growing demands for AI and Machine learning (ML) applications in research and academic environments have created a pressing need for continuous development of software engineering tools and methodologies. AI and ML are applied in numerous domains, including scientific computing, data analysis, and decision-making, driving the requirement for automation, efficiency, and innovation in academic research and education.

Previous work at our institution has focused on designing and implementing an efficient framework for leveraging locally operated large language models (LLMs), harnessing the benefits of generative artificial intelligence while providing control over data privacy and intellectual property, and managing the extensive size of the models [1]. LLMs, currently at the forefront of AI development, are tools capable of processing and generating human-like language, enabling various applications such as text classification, sentiment analysis, and language translation. Locally operating these models is possible thanks to open-source libraries such as HuggingFace [2] and Langchain, thus providing a myriad of resources while simultaneously addressing the complexities of intellectual property ownership and data security inherent to this technology.

## II. MOTIVATION

The landscape of LLMs is rapidly evolving, marked by the emergence of new architectures, training methods, and applications. This dynamic environment has captured the interest of researchers from various fields, eager to utilize LLMs to innovate and improve their methodologies. As LLMs continue to advance natural language understanding, generation, and processing, scientists are increasingly interested in integrating these technologies into their workflows. With the promise of enhanced productivity and innovation, the allure of LLMs has sparked a new wave of exploration and experimentation in the research community, driving the need for novel solutions and best practices to leverage these powerful models.

During the last few months alone, researchers at our institution have presented with a variety of use cases:

### 1) Data Synthesis and Analysis

- A researcher wants to synthesize extensive questionnaire data from a study that explores the perception of DNA and genetic inheritance in a selection of African tribes.
- A professor wants an LLM to classify the types of products for many companies according to a code list, facilitating decision-making in the manufacturing sector.

### 2) Information Retrieval and Search

- A professor aims to scrape the web of a collection of manufacturing companies that operate at the county level and use an LLM to extract the type of products each company processes.
- A researcher wants to efficiently search through a collection of previous science grant applications to retrieve technical information.
- A staff member intends to extract detailed information such as required format and specific deadlines from documents describing guidelines for national science funding opportunities.

Most of the applications highlighted require retrieving and incorporating relevant information from existing sources in addition to generation text, emphasizing the

need for LLMs to move beyond standalone generation and integrate retrieval capabilities. This is a perfect scenario for retrieval-augmented generation (RAG), a tool that can effectively combine the strengths of both generation and retrieval models to produce accurate and contextually relevant outputs by grounding generation in specific, reliable data sources. By targeting a specific corpus and providing citations, RAG can mitigate hallucinations, a common problem with LLMs. Hallucinations occur when LLMs generate plausible albeit incorrect or non-sensical information. Sourcing the content generation in a reliable and curated dataset RAG reduces these errors, producing more accurate and reliable outputs.

A special scenario is the use case where a researcher wants to combine web scraping features, the ability of LLMs to process a multitude of different inputs, and using RAG to perform classification using a detailed and comprehensive list of codes spanning a wide range of manufacturing sectors. By leveraging RAG and embracing the interdisciplinary nature of these applications, we can develop innovative solutions that bridge the gap between language processing, information retrieval, and domain-specific expertise, ultimately enhancing the productivity, efficiency, and accuracy of researchers and practitioners across various fields. In addition, hosting these models locally provides a secure environment for researchers to use unpublished and private data without the risk of uploading to third-party cloud services.

### III. IMPLEMENTATION

RAG represents a powerful paradigm in natural language processing, combining the strengths of LLMs with efficient retrieval-based techniques. RAG enables targeted questioning within a specific dataset, leveraging the capabilities of LLMs to extract contextually relevant responses. Our implementation of RAG exploits a centralized repository of open-source pre-trained models downloaded from HuggingFace, strategically stored in a scratch directory with high data transfer bandwidths facilitated by a 200GiB/s InfiniBand infrastructure. This ensures rapid model loading times while avoiding the redundancy of multiple identical models stored in the system.

The initial step involves creating a vector database from the corpus of text we want to query. This begins by parsing the documents, mainly PDFs, CSV files, and plain text documents, although expandable to comprise a list of formats supported by the LangChain document loaders, including a user-defined custom class. These documents can be read from a directory, as a zip compressed file, or as a URL pointing to a zip file. The pre-processing of the documents entails splitting the text and creating chunks of a selected size and overlap. We employ a pre-trained transformer-based language model to convert the split text data into dense vector embeddings, which capture the semantic meaning of the text enabling efficient similarity searches. Specifically, we selected to store the

vector embeddings in a chroma database for its speed, simplicity, and ample documentation for developers. The final step is to create a retriever to search the chroma database utilizing similarity search techniques, where the query is compared against the document embeddings. We specify two adjustable parameters, the number of sources and the amount of top-k most similar documents retrieved per answer. Notably, the user creates the database and persists in memory, only having to be recreated to include new documents. Alternatively, research computing staff can facilitate this step.

Central to our system’s functionality is a question-answering chain, powered by a local LLM. We load the LLM, which has been pre-trained and fine-tuned for text generation, and a tokenizer, laying the groundwork for the system’s generative capabilities. By default, given the performance quality and relatively manageable size, the selected model is the Llama 2 13B [3]. Loaded quantized, the model takes 13 GiB and fits in one of our 20 GiB NVIDIA A100 multi-instance GPU (MIG) slices, facilitating an efficient utilization of resources across our supercomputer. The use of an LLM allows us to take the retrieved documents as context and generate coherent and contextually relevant responses to the input queries. Moreover, our system goes beyond simple question answering by incorporating a memory mechanism.

```
#!/bin/bash
#SBATCH -p general
#SBATCH -q public
#SBATCH --time=0-00:30:00
#SBATCH --gres=gpu:a100:1

# set up the environment
module load mamba/latest
source activate genai

# create the database
python scripts/create_db.py <data_dir> <db_dir>
# RAG
python scripts/query.py <db_dir> <query> <output>
```

Fig. 1. Template bash script to create a vector database and perform retrieval-augmented generation in a Linux-based supercomputer.

A novelty characteristic of our RAG pipeline is the ability to retain a summary of previous questions and answers in memory to aid in future queries during the question and answer session. This is implemented by building a conversation summary memory that uses an LLM to create a summary of the conversation over time. Synthesizing the conversation and inserting it into the LLM prompt, this memory reduces the number of tokens required to retain the essence of previous information, preventing exceeding the context length and resulting in answers with better

context.

The usage of our RAG implementation spans two primary modes: interactive and batch queries. Interactive usage allows users to engage in real-time dialogue, posing questions and receiving immediate responses (Fig. 2). To facilitate user interaction, we’ve integrated a user-friendly Gradio interface [4], providing users with a seamless platform to interact with our system. Through this interface, users can input questions effortlessly, while advanced options such as temperature control and maximum token generation offer additional customization. On the other hand, batch queries facilitate large-scale data processing tasks, enabling efficient processing of extensive datasets with minimal user intervention (Fig. 1). This dual approach ensures versatility and adaptability, catering to diverse user needs and application scenarios. Note that our RAG implementation is easily adaptable to any centralized system that uses Slurm and supports Jupyter Notebooks.

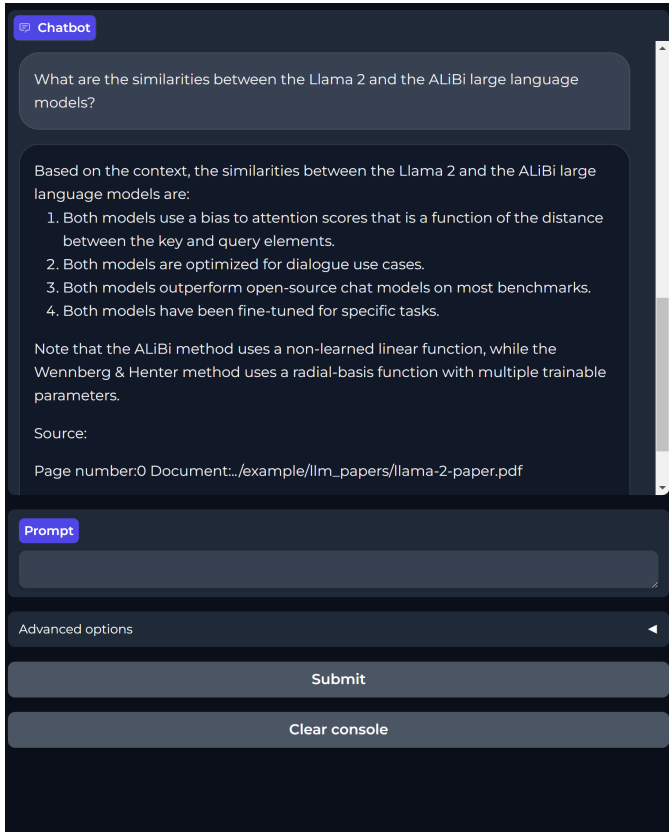


Fig. 2. Example Gradio interaction for an implemented RAG. Specifically, the Llama 2 model with 13 billion parameters was initialized in a background Jupyter notebook with Gradio enabled, launching an independent, temporary and publicly accessible Gradio powered chat-like web application.

### A. Models

The integration of LLMs remains a highly relevant and sought-after area in the field, with users frequently requesting and showing great interest in utilizing the latest

models. At our institution, in addition to supporting open-source models including Llama 2 (7b, 13b, and 30b) chat and Falcon 180B chat versions, we have incorporated the newest Llama 3 model from Meta (8b and 70b instruct) [5] and the colossal Grok-1 from xAI. Llama 3 offers substantial advancements, including a large context window, and can be easily integrated with previous pipelines after finessing its prompt syntax. On the other hand, Grok-1 has only been run as a proof-of-concept test, since loaded in its original size the model requires 520 GiB of GPU memory, which only one node at our institution can provide.

The decision of which LLMs we support and recommend is a carefully considered process, driven by user requests. To ensure a model is suitable for support, it must be open-source, have no limitations on intellectual property, and be feasible to run within our computational infrastructure. For instance, we do not recommend closed-source models as they limit accessibility, transparency, and may have an accompanying cost. In addition, we discourage heavy usage of models that require a large amount of resources, as these may result in large waiting times for the allocation of resources and limit the effective computing time in our system.

Moving forward, we plan to support newer models using the current framework that leverages HuggingFace and evaluate their performance via user feedback. Furthermore, after engaging with the community of facilitators at other universities that support LLMs, we are exploring alternative methods to provide LLM access that utilize ollama.

## IV. CONCLUSION

In this paper, we have explored the local implementation of retrieval-augmented generation (RAG) to enhance the contextual accuracy of large language models within our computational infrastructure. RAG significantly reduces hallucinations, leading to more accurate and reliable outputs. In addition, locally hosting the models creates a secure framework, allowing researchers to analyze their data unpublished data. Therefore, by leveraging this tool, researchers at our institution are exploring innovative approaches to their work. Most of the user requests mentioned previously have already started using this technology.

As the field of generative artificial intelligence continues to evolve, our mission is to stay at the forefront to assist and facilitate research and innovation. Specifically, future work in this area involves general support and guidelines for fine-tuning, expanding the number of models available, and improving the adaptability of this technology in multidisciplinary projects.

## V. CODE AVAILABILITY

The source code for the project, along with documentation and example data, is freely available on the GitHub repository: <https://github.com/jgarciamesa/US-RSE24-RAG> (doi: 10.5281/zenodo.13363087).

## ACKNOWLEDGMENT

The authors acknowledge Research Computing at Arizona State University for providing HPC and storage resources [6].

## REFERENCES

- [1] D. Shah, G. Speyer, and J. Yalim, “Centralized provisioning of large language models for a research community,” in *Proceedings of the SC’23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, 2023, pp. 704–707.
- [2] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, “Huggingface’s transformers: State-of-the-art natural language processing,” *arXiv preprint arXiv:1910.03771*, 2019.
- [3] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, “Llama 2: Open foundation and fine-tuned chat models,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.09288>
- [4] A. Abid, A. Abdalla, A. Abid, D. Khan, A. Alfozan, and J. Y. Zou, “Gradio: Hassle-free sharing and testing of ML models in the wild,” *CoRR*, vol. abs/1906.02569, 2019. [Online]. Available: <http://arxiv.org/abs/1906.02569>
- [5] Meta, 2024-. [Online]. Available: <https://ai.meta.com/blog/meta-llama-3/>
- [6] D. M. Jennewein, J. Lee, C. Kurtz, W. Dizon, I. Shaeffer, A. Chapman, A. Chiquete, J. Burks, A. Carlson, N. Mason, A. Kobwala, T. Jagadeesan, P. Barghav, T. Battelle, R. Belshe, D. McCaffrey, M. Brazil, C. Inumella, K. Kuznia, J. Buzinski, S. Dudley, D. Shah, G. Speyer, and J. Yalim, “The sol supercomputer at arizona state university,” in *Practice and Experience in Advanced Research Computing*, ser. PEARC ’23. New York, NY, USA: Association for Computing Machinery, 2023.