Journal of Big Data and Artificial Intelligence

WWW.JBDAI.ORG

ISSN: 2692-7977

JBDAI Vol. 3, No. 1, 2025 DOI: 10.54116/jbdai.v3i1.50

Improving Large Language Model (LLM) Performance with Retrieval Augmented Generation (RAG): Development of a Transparent Generative Artificial Intelligence (Gen AI) University Support System for Educational Purposes

Nishitha Chidipothu
Rutgers University
nc795@scarletmail.rutgers.edu

Rick Anderson
Rutgers University
rick.anderson@rutgers.edu

Jim Samuel
Rutgers University
jim.samuel@rutgers.edu

Alexander Pelaez
Hofstra University
alexander.pelaez@hofstra.edu

Julia Esguerra
Rutgers University
jue5@scarletmail.rutgers.edu

Md Nurul Hoque
Rutgers University
nurul.hoque@ejb.rutgers.edu

Abstract

This study works on the development of a generative artificial intelligence (AI) university support system (GenAI-USS) by improvising retrieval augmented generation (RAG) architecture to improve the performance of large language models (LLM) in a way that supports stepwise transparency. We aim to achieve better transparency and flexibility, and improved accuracy of responses to queries based on university data assimilated from university webpages and knowledge sources. We use RAG to develop a plug-and-play mechanism, along with prompt selection to boost LLM accuracy. One of the key components in our GenAI-USS is the capture and integration of real-time information via live retrieval into the generative AI process. This domain-specific knowledge assimilation with real-time updates to capture changes and new information serves as a specialized dynamic expert knowledge database for RAG. Our RAG mechanism pulls in relevant, up-to-date information from the dynamic database, which pulls real-time data from targeted predetermined knowledge sources. The other key component in our GenAI-USS design is the deliberately designed information processing visibility at each stage of the process to ensure full transparency, and this includes the following: overview, data collection, storage encoding, testing, chatbot interaction, and search. The testing module allows for interactive viewing of generated responses and their sources. Our strategy is expected to lead to higher-quality AI-generated output via targeted information retrieval, hallucination mitigation, accuracy improvement, and timely data updates. Essentially, on the submission of a query, the RAG-dependent GenAI-USS first identifies the most relevant information from the specialized expert knowledge database and then factors this into the generative AI response development process. This results in a successful implementation of our primary objectives of a transparent and flexible user-choice–driven RAG-based generative AI system, which also provided heuristically notable improvements in the quality of output produced.

Keywords *generative artificial intelligence, large language model, retrieval augmented generation, NLP, NLU, NLG, AI, DSS, Chatbot, transparency, ethics, education.*

1. Introduction

Textual data presents us with an array of opportunities for creating artificially intelligent innovations (Garvey et al. 2021). In recent years, there has been a significant increase in the amount of unstructured textual data being leveraged to glean business insights from textual data distributions and models (Samuel et al. 2020a; 2022). Substantial advancements in natural language processing (NLP) and natural language understanding have led to the development and adaptation of large language models (LLM) for a broad range of applications (Samuel 2023). These models are trained on vast amounts of data to model probabilistic associations between tokens to serve as a machine "brain," which can then be used for actions such as interpreting and answering human questions (IBM 2023). LLMs depend on the "transformer" model architecture that features self-attention mechanisms, which enables them to learn at a significantly accelerated pace compared with conventional models (CloudFare 2024). LLMs are a significant part of the basis for generative artificial intelligence (Gen AI) for text and language generation capabilities, leading to a new wave of AI technologies and applications (Samuel et al. 2024a). Human queries or questions serve as unstructured input, and the LLMs have the capacity to generate output via probabilistic associations of word sequences. The quality of LLM responses depends on the training data, occasionally resulting in "hallucinations," in which the model produces spurious answers, leading to wrong and often wild responses to human users (Xu et al. 2024).

The introduction of OpenAI's ChatGPT (https://openai.com/) and Google DeepMind's Gemini (https://deepmind.google/technologies/gemini/) has given an opportunity to the general public to derive answers to queries that may not be readily available on traditional search tools such as Google (OpenAI 2024; Google Deepmind 2024). Transformers are used in building LLMs, which can deal with human language and various tasks related to NLP, such as machine translation (Anderson et al. 2024). LLMs have shown unparalleled prospects and capacity for innovations in AI applications, leading to the development of numerous enhanced NLP methods and tools. As an illustration, it has been shown that LLMs can improve the scope of NLP functions such as emotion classification and sentiment analysis for public sentiment perception (Alan et al. 2024; Samuel et al. 2020b, 2024b). Popular LLMs are generally referred to as foundation models and have outstanding "emergence" and "homogenization" capabilities (Samuel 2023; Tam 2023). Here, emergence refers to the potential for spontaneous and organic discovery of surprising features and possibilities with LLMs, and homogenization refers to the capability of LLMs to increasingly serve as a common platform for various AI applications. However, these Gen AI tools also have their respective drawbacks, including the potential for misinformation, hallucination, and harmful content generation. Next, we discuss some known challenges and risks associated with these technologies.

A significant challenge of LLMs is their tendency to generate factually inaccurate, meaningless, and absurd information, although seemingly credible, also known as hallucination (Li et al. 2023). This issue is particularly concerning in contexts in which factual accuracy is critical, such as for news reporting and academic research. LLMs can subtly alter factual information while maintaining a seemingly credible narrative, as demonstrated in one of our experiments. In this case, an LLM-based generative system provided a fairly believable narrative about a historical event that it specified as occurring on Monday, October 21, 2014. Although all other facts were correct, the date was inaccurate: October 21, 2014, is actually a Tuesday, not a Monday, as specified by the LLM. This subtle distortion highlights the reliability challenges of LLMs in factual applications.

Another challenge is in keeping the knowledge base of the LLMs updated in real-time. Without continuous updates, these models rely on outdated information that leads to erroneous answers to simple questions such as "Who is the current president of the United States?" LLMs trained on largely historical data may not sufficiently incorporate contemporary events or real-time developments and fail to adapt to the rapid changes in real-world information (Yu and Ji 2023; Duan et al. 2023). Furthermore, overleveraging could lead to dependency on LLM-based tools and associated harms, such as diminished intellectual capabilities for students in academic environments, resulting in ethical dilemmas for educators. Necessary guidelines and strategies must be implemented by policymakers to foster genuine learning and mitigate the disadvantages that lead to ethical and practical challenges (Das 2024).

The challenge of LLM-generated inaccuracies, hallucinations, and other errors can be significantly mitigated by the use of retrieval augmented generation (RAG) methods, which have become popular in the recent past. A key advantage of RAG is its ability to maintain current knowledge by directly encoding website content, which can be refreshed as source material changes. Because the LLM summarizes data directly from the vector store, responses are grounded in the exact source material, which ensures relatively better accuracy and traceability. We propose a novel RAG-based architecture for the organization of a plug-and-play mechanism for improved transparency and flexibility. One of the key components in our proposed Gen AI university support system (GenAI-USS) is the capture and integration of real-time information. This is achieved via a live retrieval process embedded into the Gen AI system, which updates the RAG database with near real-time information and can also be updated at will. This serves as a specialized expert knowledge database for our GenAI-USS system. Our RAG mechanism pulls in relevant, up-to-date information from a predetermined dynamic database, which pulls real-time data from targeted knowledge sources.

Furthermore, our GenAI-USS design is deliberately designed to maximize process visibility of each stage of the Gen AI process to ensure full transparency. This includes all the major phases, including the following: overview, data collection, storage encoding, testing, chatbot interaction, and search. In addition, the testing module allows for interactive viewing of generated responses and their sources. The GenAI-USS design is centered on testing against relevant question datasets and answers evaluated to be accurate. The combined optimization of embedding and language models ensures that responses remain contextually anchored to website data while enabling efficient updates and source attribution. This approach reveals content gaps and structural barriers in the current site architecture that may impede effective knowledge extraction. Scalability can then be achieved by optimizing the embedding and language models for speed, cost, and energy consumption.

The rest of this article is organized as follows: we perform a literature review on topics critical to our research, with a focus on identifying key concepts and facts on areas such as RAG, best practices on the use of tools such as Hugging Face (https://huggingface.co/) and Streamlit (https://streamlit.io/), and recent advancements in LLMs and prompt engineering. We also identify potential gaps and scope for improvement. Under "Domain and Data," we discuss the various aspects of how data are being used, interpreted, and leveraged for domain-specific considerations and challenges. Next, we discuss the design and development of our GenAI-USS application, explaining each stage with details about areas such as the LLMs used, embedding models, data sources, prompts, and output. Finally, we discuss the implications of our research and GenAI-USS application, the scope for future research, and conclude with thoughts on the way ahead for linguistic Gen AI.

2. Literature Review

We examine several key themes for our research, covering RAG, best practices on the use of tools such as Hugging Face and Streamlit, and prompt engineering. We also cover prompt-design, zero-shot and few-shot learning, embeddings, chunking parameters, temperature, and associated implications. We first try to understand the kind of models and tech stack required for our GenAI-USS application to be successful. We then explore the prompt engineering techniques, including understanding how to design the prompts, and zero-shot and few-shot learning in prompt engineering. Despite significant advancements, there are gaps that remain in the literature. Thus, this review provides us with a foundation for understanding and exploring concepts and usage of these tools. This will set the stage for our current application as well as for future research. We start with RAG, because this mechanism anchors our motivation and our framework; we then introduce Hugging Face as our resource hub for operating purposes; this is followed by elaborations on the user interface, prompt engineering, designing prompts, and other critical elements of our framework.

2.1. RAG

Over time, we have encountered challenges with LLMs that produce inaccurate text-based generative responses, primarily due to the absence of precise and up-to-date datasets attached or linked to the queries referred to as "hallucinations." This led to the emergence of RAG. RAG uses LLM to generate responses with the help of a database attached so that when a query is presented, the RAG system first identifies relevant information from external sources and then integrates this information into the response generation process (Jiang et al. 2023). The database attached will be divided with the help of a splitter because RAG works effectively with smaller text segments stored as document snippets. These document snippets will be fed into an embedding machine to convert the text into vector embeddings. This step helps in retrieving more factual information, reducing the occurrence of hallucinations, thereby improving the quality of outputs (Gao et al. 2024; Aquino 2024).

The forward-looking active RAG uses iterative retrieval-augmented generation to anticipate future content in sentences. It retrieves relevant documents and regenerates sentences that contain low-confidence tokens. This approach enhances the efficiency of retrieving information multiple times (Jiang et al. 2023). Given its proven effectiveness, RAG has become the most cost-effective, straightforward, and low-risk solution to achieving superior performance for GenAI applications, which leaves many companies with little choice but to adopt it (Proser 2023). Furthermore, recent research highlights the importance of using RAG in contexts that require high accuracy, such as university-level textbooks, in which ChatGPT-4 was unsuccessful in delivering accurate information (Wang et al. 2024).

2.2. Hugging Face

Hugging Face stands out as a rapidly expanding hub for hosting open-source projects centered on machine learning and AI (Hugging Face 2024) and will serve as a data store for all the LLMs that we require for GenAI-USS. This platform's primary focus is on transformers that streamline the process for individuals and small business startups to develop extensive LLMs (Vasilis 2024). Hugging Face's transformers are a groundbreaking advancement in NLP, combining transfer learning methods with large-scale transformer language models. It offers state-of-the-art transformer architectures, along with a collection of pre-trained models, which makes it a cornerstone in modern NLP research and provides powerful tools for various tasks (Wolf et al. 2020).

2.3. User Interface

We use Streamlit, an open-source Python framework (Streamlit 2024), to develop our framework. Streamlit offers a chance for individuals less experienced in developing applications to perform well by providing resources that enhance the appearance and interactivity of projects. We are hosting and deploying our project on the Streamlit community cloud. The deployment process involves configuring the Streamlit app for web integration, cloning our GitHub repository, and using the platform's infrastructure to ensure smooth integration with other machine learning models (Imanuelyosi 2022).

2.4. Prompt Engineering

Prompt engineering optimizes and refines input queries so that the LLMs can generate more enhanced, accurate, and coherent responses. Our focus is on using prompt engineering to ensure that we guide the users in providing accurate inputs to achieve relevant outputs. GPT- and DALL-E-like foundation models use natural language prompts to facilitate interaction with AI models, which have introduced us to newer tools and methods. For example, Prompt Sapper facilitates building AI services based on prompt engineering (Cheng et al. 2024). Prompt engineering has been used as an important technique for designing relevant instructions or queries to improve the performance of the LLMs, which are used in AI, ML, and NLP tasks such as sentiment analysis, summarizing, questions answering, and arithmetic reasoning (Shi et al. 2023). Shi et al. (2023) also emphasized the use of chain of thought, zero-chain of thought, and in-context learning in effective prompt engineering. Furthermore, Ekin (2023) pointed out that clarity, explicit constraints, and leveraging various types of questions are some important factors for prompt engineering. Meanwhile, Lo (2023) emphasized conciseness, logic, explicitness, adaptability, and reflectiveness, also known as the CLEAR framework, in dealing with the prompt engineering for AI, LLM, and NLP models. Prompt engineering uses instruction-basis, information-basis, reformulation, and metaphoric prompt techniques, along with effective evaluation processes for instructing LLMs to improve the performance of the AI-NLP tasks (Rathod 2024). In addition, Ye et al. (2024) emphasized removing unwanted elements from the prompts, thus improving capabilities of reasoning and optimizing communication between tasks to build a meta-prompt so that we can lead the LLMs in enhancing their performance.

2.5. Designing Prompts

In designing effective prompts to interact with the AI, LLM, and NLP models, it is essential to have a comprehensive understanding of what the users are looking for and the capabilities of the models. There are some key principles in prompt design that can improve the effectiveness of prompting across various models and tasks (Herrmann and Nierhoff 2017). Herrmann and Nierhoff (2017) emphasized making the prompts optional and comprehensible so that the users can interact with the models without any difficulties and pressure. They suggested that each prompt should have a clear purpose that leads to specific actions and desired outcomes. In addition, the prompts should be structured according to the respective user contexts and users should have control over the prompt design so that they can achieve improved acceptance and effectiveness of prompting (Herrmann and Nierhoff 2017).

Desmond and Brachman (2024) suggested a trial-and-error process in prompt designing. This iterative testing system allows the users to refine the prompts based on the responses from the models. Furthermore, Liu and Chilton (2021) emphasized adding structured elements, such as subject and style, to improve the coherent outputs in generative tasks. Although these techniques and strategies are useful for creating a framework for designing prompts effectively, there are some inherent complexities and challenges for the users to interact with the AI-NLP models, such as performing consistently across diverse tasks and applications (Dang et al. 2022).

2.6. Zero-Shot and Few-Shot Learning

Zero-shot and few-shot learning are instrumental in dealing with AI models, particularly in limited labeled data settings. These techniques allow the AI models to leverage the previous knowledge efficiently and generalize the outputs from none to very little labeled data. If there are no direct training examples in the datasets, zero-shot learning helps AI models recognize the unrevealed classes (Deng et al. 2024). Deng et al. (2024) demonstrated the speech-to-text alignment effectively by using Wave2Prompt, which combines spoken units with LLMs to execute zero-shot activities. In addition, Liu et al. (2021) showed how the Micro framework improves the capacity of the LLMs within contexts and enhances the performance of extracting zero-shot relations by using varied datasets without updating parameters. However, few-shot learning can handle the AI models with limited examples. Few-shot learning helps in cross-domain fault diagnosis by using embedding optimization and solves the challenges and problems effectively in industrial tasks (Qiu et al. 2024). Liu et al. (2021) introduced a combined representation learner for classifying few-shot images, which can remarkably enhance performance across diverse datasets. Although these methods have great prospects, there are some challenges in generalizing the outputs across various domains and tasks, which requires future research initiatives in this arena.

2.7. Context and Prompt Engineering

Contextualization is instrumental in improving the performance and effectiveness of AI, LLM, and NLP models as well as user experience by leading the prompts to specific contexts. Jasmine (2024) pointed out that AI models and systems can achieve improved and accurate outputs from personalized, contextualized, and relevant prompts. Muktadir (2023) stated that contextualized prompts can achieve enhanced controllability and adaptability in AI-NLP tasks by using transfer learning and attention mechanisms used for context-aware language models. In addition, de Fonseca et al. (2023) demonstrated that the prompts that are context-aware showed improved performance, more accuracy, and cost efficiency compared with the traditional supervised methods and techniques. Although contextualization is beneficial for prompting AI, LLM, and NLP models, this requires extensive data to build effective prompts. There is also the possibility of overfitting to a certain context, limiting the capabilities of generalizability of the models across diverse topics.

2.8. Embeddings

Embeddings are considered the languages of LLMs and GenAI (QuantumBlack 2023). This approach uses linear algebra to convert real-world objects into numerical representations in a high-dimensional space, which allows machine learning and AI systems to comprehend complex knowledge. These embeddings are continuous values. Embeddings provide a simplified representation of real-world data while preserving semantic and syntactic relationships (AWS 2024). In this project, we are working with two different types of embedding models to choose from: sentence-transformers/all-MiniLM-L6-v2 and thenlper/gte-smalls. In the future, we can add more embedding models to work with. Here, the main purpose of using embedding models is to convert the JSON documents, which are in the form of vector stores, for the LLM model to interpret and answer our queries in the next steps.

2.9. Chunking Parameters

Chunking is the process of breaking down large files into smaller segments for better semantic understanding. Before embedding, our primary objective will be to provide as little noise as possible to the embedding model for it to stay semantically relevant. “Your goal is not to chunk for chunking sake, our goal is to get our data in a format where it can be retrieved for value later” (Mishra 2024). There are multiple chunking strategies: fixed-size chunking, recursive chunking, document-based chunking, and semantic chunking. In our project GenAI-USS, we are focusing on recursive character text split, which is a method that takes up large texts and splits them into smaller chunks based on the parameters given, including chunk size and chunk overlap.

2.10. Temperature

The "temperature" of an LLM is a key control parameter that governs the randomness of the responses generated by the LLM (Peeperkorn et al. 2024). Usually, the default is set to 1, which represents a balanced position. When set closer to 0, a lower level of randomness is expected, and a higher temperature significantly greater than 1 will result in hallucinations. In our design, we choose a low temperature so that the LLM prioritizes factuality during the response process rather than randomness. This parameter can be used to control an LLM's entropy, and, when the temperature is set lower, then the outputs tend to be more factual, predictable, often repetitive, and more closely and more likely aligned with source information. In contrast, higher temperature settings spur LLMs to generate more random, less likely aligned with source information, and relatively more unique responses. This is an important control parameter and needs to be set to be aligned with the main objectives of a Gen AI system.

3. Conceptualization of GenAI-USS

Our current project develops and significantly improves the scope and performance of our academic bot's capabilities, leading to the creation of the Rutgers University GenAI-USS. GenAI-USS is a project developed by the Office of University Online Education Services (UOES TLT n.d.) and the Master of Public Informatics Program at the Bloustein School, both at Rutgers University (Rutgers University 2001). The RAG ideation for GenAI-USS proceeded from the Public Informatics NLP Studio, including the Spring 2024 and Fall 2024 classes in the Master of Public Informatics program. GenAI-USS, also known as the "Chatbot Online Resource (COR) Navigator," is a project under the UOES. COR Navigator was initially designed to assist academic units across Rutgers University in crafting and delivering hybrid and fully online courses. The platform aims to enhance effectiveness, streamline processes, and elevate satisfaction within the Rutgers Online Learning community, which includes prospective students, current students, faculty, and emerging online programs (UOES Rutgers 2023). The GenAI-USS project focuses on developing an interactive question-and-answer interface, including frequently asked questions, and enabling knowledge synthesis through a RAG module. By allowing the incorporation of a broad range of URLs, this system enhances relative accuracy, depth of knowledge, options for expanding breadth of knowledge, and contextual relevance of Gen AI outputs.

GenAI-USS uses a local embedded selection approach, curating information from diverse sources such as blogs, documents, PDFs, and web pages hosted within the UOES domain, with the capability to be extended to any part of Rutgers University. The curated dataset is integrated into a vector database and processed through an LLM by using the RAG framework. This integration streamlines access to valuable, previously hidden information and facilitates seamless interactions between faculty, students, and staff, and the GenAI-USS platform. A conceptual representation of the system architecture is presented in Figure 1. The "Organize" section highlights semantic organization, data clarity, and redundancy cleanup. The framework distinguishes between the human input, referred to as the "query," and the adaptable and testable prompt provided to the LLM. This adapted prompt, informed by local embeddings and the LLM's pretrained knowledge, enhances output quality and relevance compared with raw queries without prompt-engineered support. Through this adaptive framework, GenAI-USS improves system transparency and flexibility, and also supports LLM performance and accuracy, providing an innovative tool for augmenting human performance. This initiative reflects UOES's commitment to advancing online education at Rutgers University by integrating cutting-edge technology and addressing the specific needs of its diverse academic community (Samuel et al. 2022).

Our primary objective is to provide a platform for faculty, students, researchers, staff, and other stakeholders at Rutgers University to access information from the GenAI-USS. This will be achieved by implementing the RAG-LLM application, Hugging Face, and StreamLit interface. The front-end application will resemble Figure 2, which features a chatbot interface that is provided for asking questions and receiving answers.

3.1. Domain and Data

Similar to many other organizations, academic institutions are interested in implementing LLMs to enrich the learning experience for both faculty and students (ElementX 2024). It is clear that LLMs contain inaccurate information, and RAG addition will solve this problem; in our case, in which academic institutions need up-to-date and accurate information, it is ideal to implement RAG-LLM. The academic institution that we are focusing on is Rutgers University. It is ranked as the number 1 public university in the state of New Jersey and holds a history of more than 250 years with 67,200 undergraduate and graduate students, and more than 17,450 full-time and part-time staff

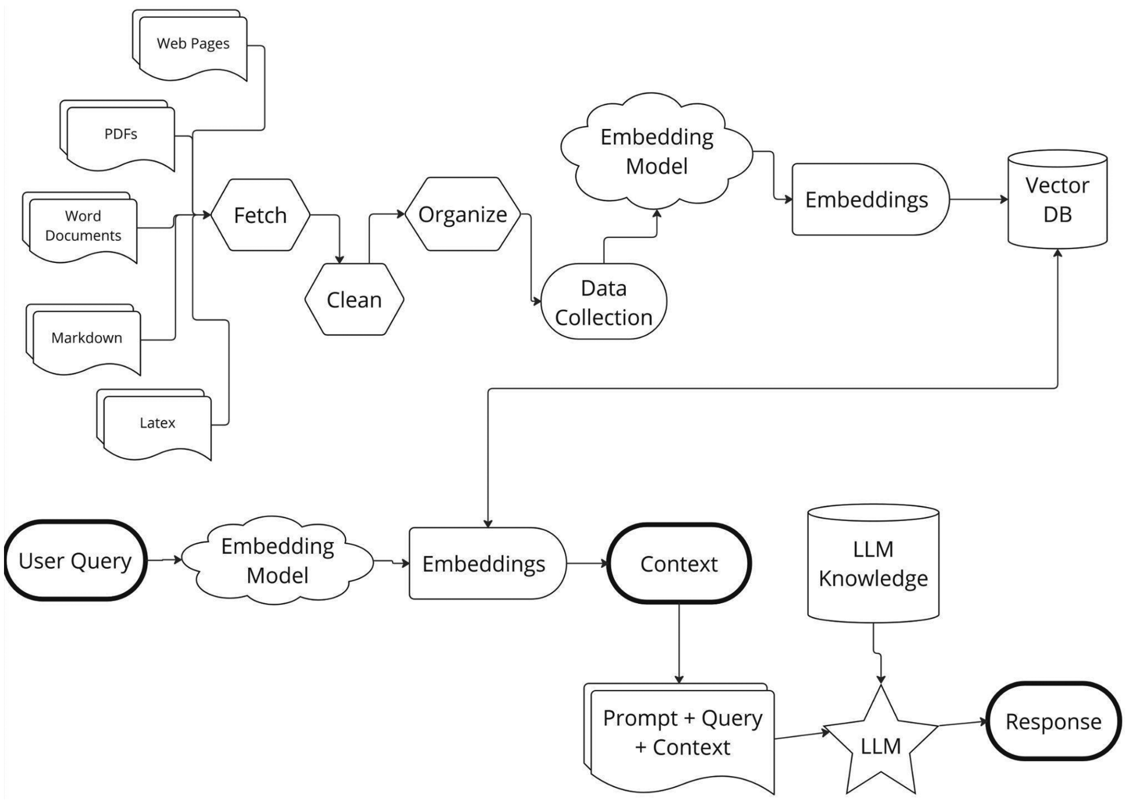
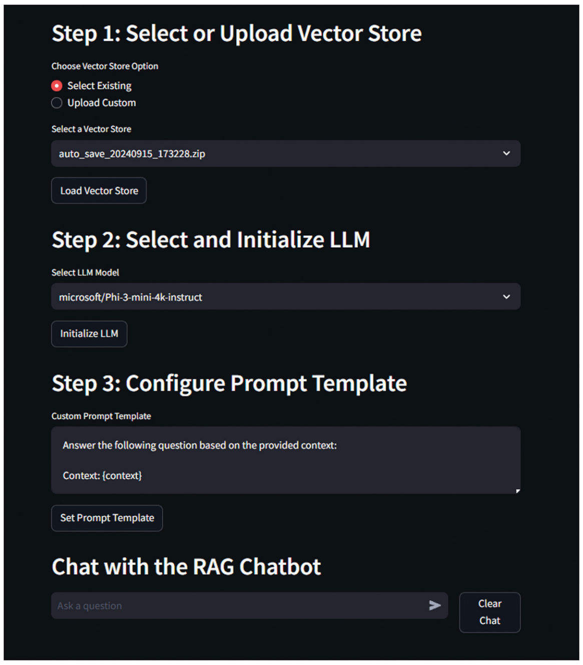


Figure 1: Generative artificial intelligence (AI) university support system (GenAI-USS) architecture.



Step 1: Select or Upload Vector Store

Choose Vector Store Option

[✓ Select Existing] Upload Custom

Select a Vector Store: auto_save_20240915_173228.zip

Step 2: Select and Initialize LLM

Select LLM Model: microsoft/Phi-3-mini-4k-instruct

Step 3: Configure Prompt Template

Custom Prompt Template:

Answer the following question based on the provided context:

Context: {context}

Chat with the RAG Chatbot

Figure 2: Screenshot of chatbot retrieval augmented generation (RAG).

(Rutgers 2020). A university of this scale has an extensive number of resources for students and faculty that act as silos of information and resources. Faculty and students do not have guidance on how to navigate through these resources, even with capable search engines to index that material. To fulfill this purpose for students and faculty at Rutgers University, we will implement RAG. The first step in the process will be with Rutgers Online Degree Programs (https://www.rutgers.edu/academics/online-degree-programs) as our primary website source. Our final goal is to access academic information through application programming interfaces (APIs), databases, document repositories, websites, or PDFs but we are currently concentrating on accessing information in the form of website URLs. The source data from Rutgers Online Degree Programs | Rutgers University will be used as input to the retriever component of RAG. This component will retrieve the relevant information by using the user's query. Next, a prompt is generated based on the retrieved information and user query, which will serve as an input to the LLM, in turn, generating an answer. The final step would include the LLM parsing the generated answer and presenting it to the user in a clear and understandable format. For the purpose of maintaining high-quality prompts, we also developed a reasonably comprehensive question bank for all reasonable questions to train the model. Our primary question bank of most anticipated questions that students and faculty would pose consists of 100 questions from prospective students' perspectives and 50 questions from faculty's perspective, based on information available on the webpages.

3.2. GenAI-USS Prompt and Prompt Control

The RAG pattern contains four key configurable points that influence its performance. We use sensible defaults for prompts and configurations for quick interactive testing, which allows users to quickly evaluate the tool's performance with minimal customization. This approach enables rapid iteration and general feedback collection. Our process extends the interactive testing by supporting comprehensive validation approaches. Users can test custom prompts for specific scenarios, whereas the advanced features enable bulk testing of multiple prompts. By following our specified data format, entire collections of prompts can be systematically evaluated against test question sets, enabling thorough performance analysis.

3.3. GenAI-USS Application

A Streamlit-based user interface will be used to facilitate the mining, processing, and embedding of the data from the Rutgers Online Degree Programs | Rutgers University. The GenAI-USS system is organized into multiple pages, each dedicated to a specific part of data processing workflows. This user interface will guide users through each step of the process, ensuring a seamless and intuitive user experience. After launching the app, users can navigate between different pages by using the sidebar. Each page includes interactive elements, such as input fields, dropdown menus, and checkboxes, which allow users to customize each step of the data processing pipeline.

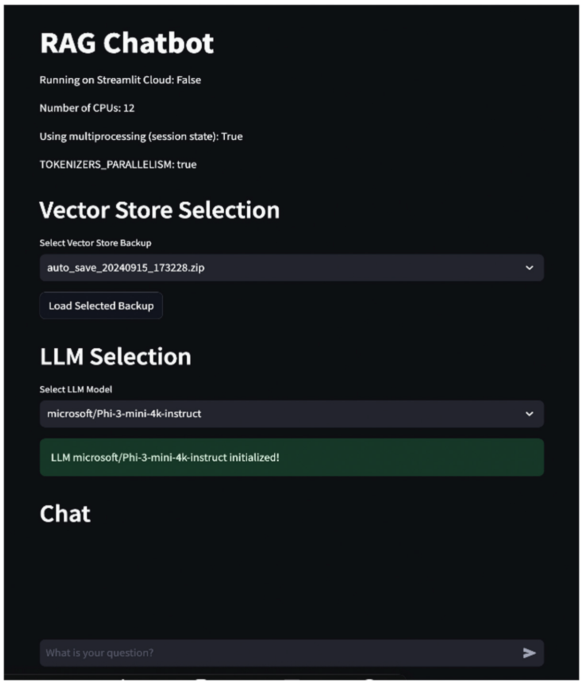
First Page: The first page of the chatbot application (Figure 3) serves as a dashboard that serves as an overview of the chatbot's current settings and provides an opportunity to configure the system's performance, select different language models, and initiate queries.

General Settings Panel: The first section of the page provides current settings for running the chatbot. This includes multiple settings related to how the chatbot environment is configured and is being executed. These settings include the following:

- **Running on Streamlit Cloud**: This option checks whether the application is being hosted on Streamlit Cloud. The value displayed will either be true or false, depending on whether the cloud hosting is active or not.
- **Number of CPUs:** This option helps understand the number of available CPU cores allocated to the chatbot.
- **Using Multiprocessing Session State**: This option checks whether the application is leveraging multiprocessing to enhance the performance of the state. If true, then multiprocessing is enabled to handle tasks concurrently and can improve performance and process time.
- **TOKENIZERS_PARALLELISM**: This option, if set to "True," can boost the efficiency of text processing by parallelizing the tokenization of input text across multiple threads.

Vector Store Backup Datasets: This section of the page enables the user to choose any dataset from multiple vector store backup datasets. These vector stores serve as repositories of pre-processed text data, stored in vector format. This step ensures that the user does not perform similar vector data extraction and can directly proceed to perform queries on currently available data backups.

Selecting the LLM: The LLM defines how the chatbot processes language, generates responses, and interprets user queries. This page preselects three models to compare, chosen to represent different trade-offs in performance



RAG Chatbot

Running on Streamlit Cloud: False

Number of CPUs: 12

Using multiprocessing (session state): True

TOKENIZERS_PARALLELISM: true

Vector Store Selection

Select Vector Store Backup: auto_save_20240915_173228.zip

Load Selected Backup

LLM Selection

Select LLM Model: microsoft/Phi-3-mini-4k-instruct

LLM microsoft/Phi-3-mini-4k-instruct initialized!

Chat

What is your question?

Figure 3: Screenshot of the application.

characteristics. Each model offers distinct advantages in terms of cost, computational requirements, parameters, response relevancy, and context window size. For example, Mistral-7B-Instruct provides a balance of efficient computing and good response quality, whereas GPT-3.5-turbo offers stronger performance at a higher cost (Glover 2024). Phi-3.5-mini-instruct demonstrates capabilities with lower resource requirements. For example:

- **Mistral-7B-Instruct (7B parameters, 32K context window):** Provides efficient computing and good response quality, suitable for self-hosted deployment
- **GPT-3.5-turbo (175B parameters, 16K context window):** Offers stronger performance through API access, with higher associated costs
- **Phi-3.5-mini-instruct (3.8B parameters, 2K context window):** Demonstrates capable performance with lower resource requirements, ideal for lighter workloads

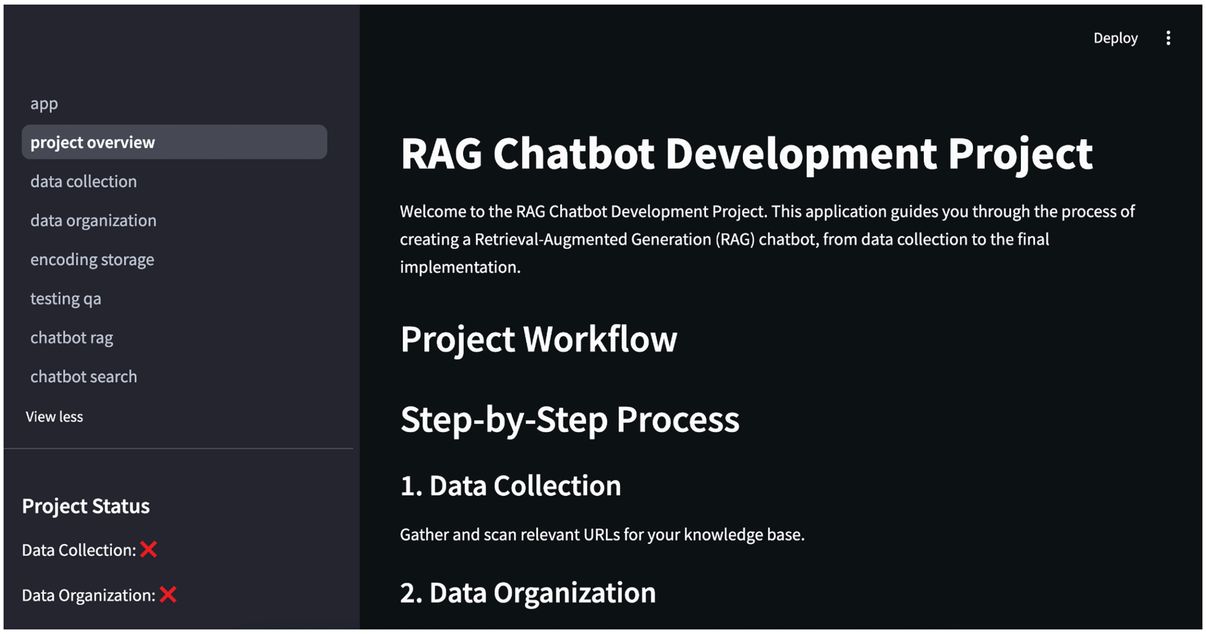
This step makes sure the model is being tailored to different use cases, in which we can choose the context window, number of parameters, and file size. This includes determining whether local LLM access versus remote-hosted API access is appropriate. This step allows for the testing of LLM configuration impact on the question and answer (QA) performance and accuracy.

Chat Section: Users can interact directly with the chatbot by entering their queries. It consists of an Input Field, a Submit Button, and an Output Display. We can input the query and click the submit button to generate the output that is based on the LLM selected and the underlying vector store dataset, ensuring relevance and accuracy.

Second Page: This page (Figure 4) provides the project overview and serves as an index page, providing users with brief information about the steps and contents available in the application.

The step-by-step process is to work on the following:

- Data Collection
- Data Organization
- Encoding Vector Storage
- Testing and QA
- Chatbot Implementation



Project Status

Data Collection: ❌

Data Organization: ❌

RAG Chatbot Development Project

Welcome to the RAG Chatbot Development Project. This application guides you through the process of creating a Retrieval-Augmented Generation (RAG) chatbot, from data collection to the final implementation.

Project Workflow

Step-by-Step Process

1. Data Collection

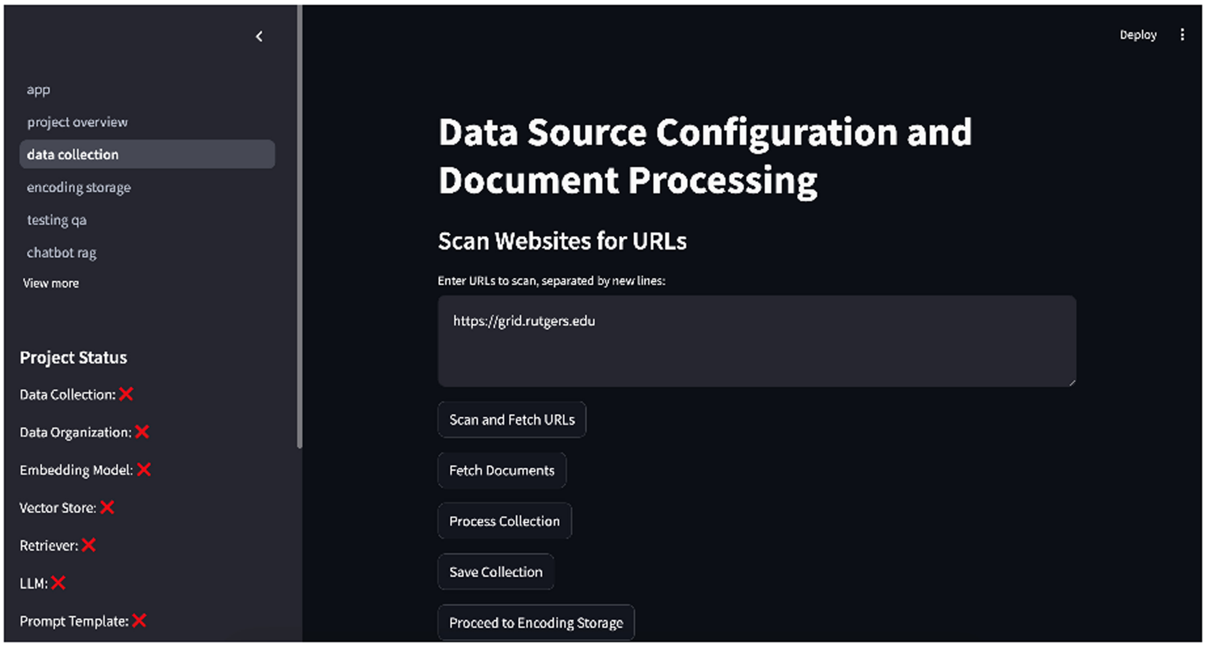
Gather and scan relevant URLs for your knowledge base.

2. Data Organization

Figure 4: Screenshot of the project overview.

With each step, we will be provided with updates with regard to whether the data are defined or documents are fetched. This page is built mainly with the help of the streamlit switch_page() function, which directs to multiple pages created in the application. This page gives an overview of the key concepts, steps to get started, prerequisites, and expected outcomes. This page mainly serves the purpose of setting expectations. Users can proceed to the next page by clicking on the "Start with Data Collection" button.

Third Page: This page (Figure 5) provides information on data collection. The core functionality of this page revolves around scanning the URLs to extract information, organizing it into tabular format, and preparing it for further processing.



Project Status

Data Collection: ❌

Data Organization: ❌

Embedding Model: ❌

Vector Store: ❌

Retriever: ❌

LLM: ❌

Prompt Template: ❌

Data Source Configuration and Document Processing

Scan Websites for URLs

Enter URLs to scan, separated by new lines:

https://grid.rutgers.edu

Figure 5: Screenshot of data collection.

URL Input Section: This section on the page allows the users to input the URLs that will act as the source data. The users can enter multiple URLs to scan, separated by new lines. The information is extracted from these URLs and displayed in tabular form. These tables will include columns that detail the following:

- URL link
- Type of URL: this column specifies whether it is an internal or external URL. Internal URLs refer to web pages within the same website, whereas external URLs are links to web pages outside the main website but accessible through it.
- Page name
- Scanned date and time

This step ensures that we gain access to every page related to the given website.

Document Processing and Cleanup: After extracting all the URLs associated with the main source URL, users can proceed to fetch, clean, and organize documents. Here, users can have the option to delete the unnecessary URLs before initiating the document retrieval process by using the "**Fetch**" button. The documents will be fetched in the form of a JSON file, which the users can save and download for later use, as shown in Figure 6. This page is crucial in ensuring that all the relevant data sources are accurately captured and pre-processed before advancing to the next steps. By giving the users control over the URL input and cleaning process, it is ensured that the system is robust and adaptable to various use cases.

Fourth Page: This page (Figure 7) provides information on the encoding storage. This step is critical for processing and storing the extracted information for further processing, particularly in tasks such as question-answering and conversational search. The number of documents that are fetched from the session state will be available. Before proceeding to the encoding step, users are given the option to upload a JSON file. This will provide an opportunity for users to work with a previously saved dataset or any new data in JSON format that needs to be processed. After this, users will be able to choose the embedding model for processing. "Embeddings" refer to vectors that encapsulate the connections and significance among words, thereby representing semantic relationships (Bhavsar 2024). These embeddings will serve as a foundation for question-answering, conversational search, and other functions. The embedding model has three different options from which to choose. These embedding model options are selected to represent different trade-offs in computational requirements and embedding quality as follows:

- **sentence-transformers/all-MiniLM-L6-v2 (384 dimensions, 6 layers):** Balances performance and efficiency, producing high-quality embeddings while maintaining reasonable computational costs. This model excels at semantic similarity tasks and shows strong performance in production environments.
- **thenlper/gte-small (384 dimensions, 4 layers):** Optimized for scenarios with limited computational resources, offering faster inference times while maintaining acceptable embedding quality. Its reduced architecture makes it suitable for environments in which processing speed is prioritized over maximum accuracy.
- **Other:** The framework supports the integration of custom embedding models, which allows teams to implement specialized models for specific use cases. This flexibility enables the testing of newer models or domain-specific variants as they become available.

More flexibility: After selecting an embedding model, users configure two critical parameters, chunk size and chunk overlap, which together determine how documents are segmented for embedding. These parameters are essential for maintaining semantic coherence and ensuring accurate retrieval of relevant information from the vector store.

- **Chunk Size:** The maximum number of characters each chunk can hold

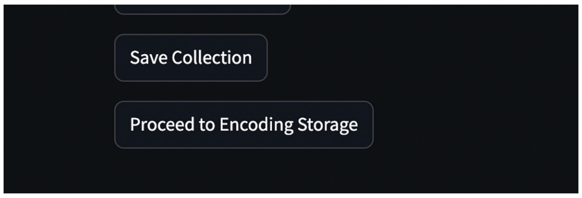
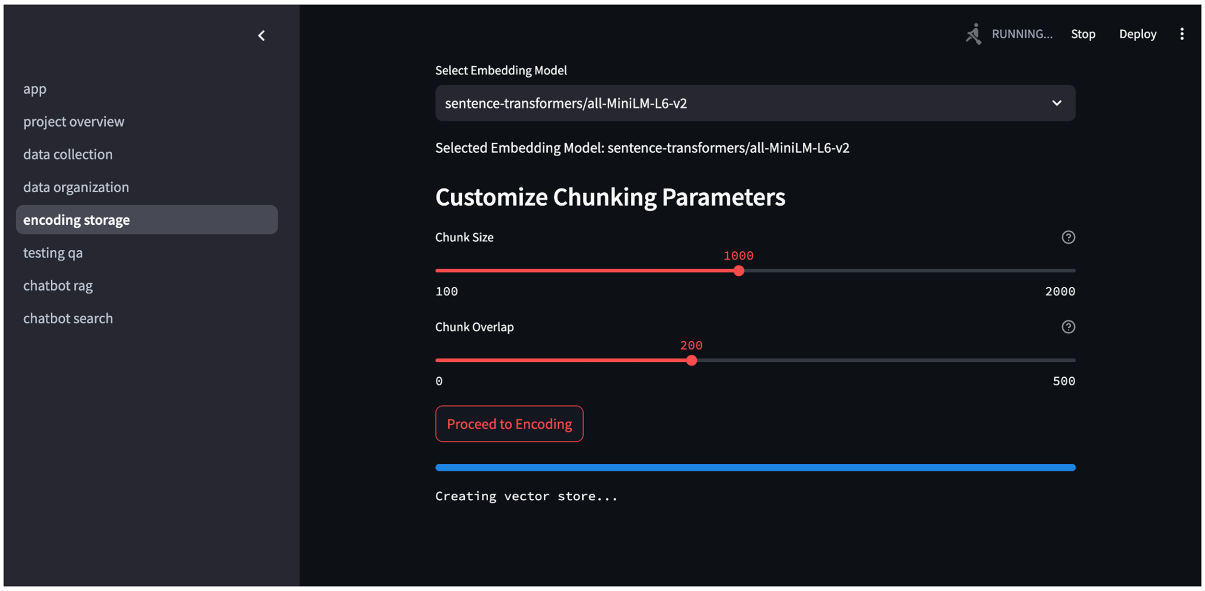


Figure 6: Save data collection.



Select Embedding Model

sentence-transformers/all-MiniLM-L6-v2

Selected Embedding Model: sentence-transformers/all-MiniLM-L6-v2

Customize Chunking Parameters

Chunk Size: 1000 (100–2000)

Chunk Overlap: 200 (0–500)

Proceed to Encoding

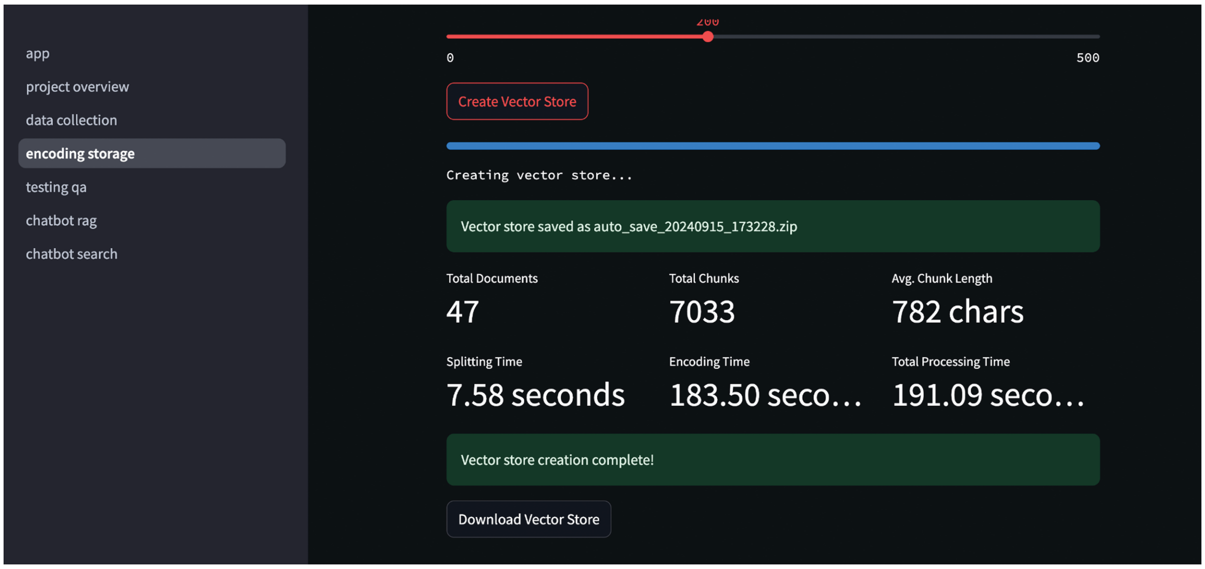
```
Creating vector store...
```

Figure 7: Screenshot of encoding storage.

- **Chunk Overlap:** The number of characters that should be shared between two consecutive chunks (Peter 2023)

We then proceed to "Create Vector Store," which gives us a progress update, along with in-detail information about the total number of documents processed, total chunks, average chunk length, splitting time, encoding time, and total processing time. Once the vector store creation is complete, we can download the vector store, which can be stored or used in the next stage: testing and QA.

The encoding and storage step is crucial because it transforms unstructured data into structured vector representations that can be effectively used in downstream tasks such as semantic search, conversational AI, and question-answering systems. The key statistics to measure the size and performance of creating the vector store from our collection are shown in Figure 8.



```
Creating vector store...
```

Vector store saved as auto_save_20240915_173228.zip

Total Documents	Total Chunks	Avg. Chunk Length
47	7033	782 chars
Splitting Time	**Encoding Time**	**Total Processing Time**
7.58 seconds	183.50 seco…	191.09 seco…

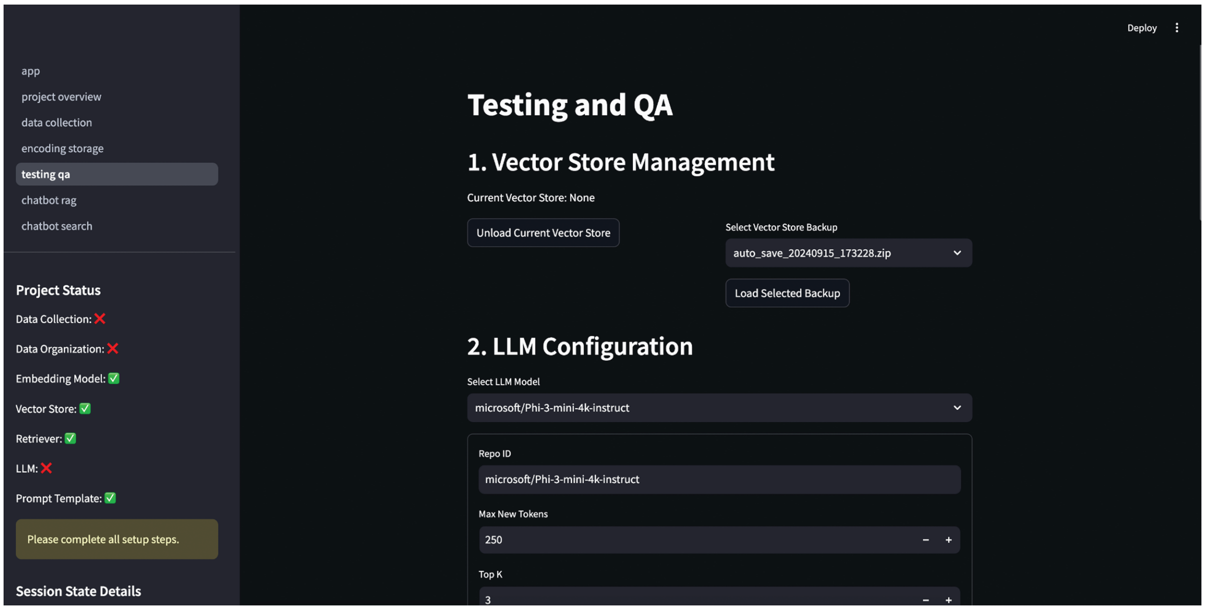
Vector store creation complete!

Figure 8: Screenshot of encoding process.

Fifth Page: This page is dedicated to testing the model and QA. We now have a list of vector store documents saved from the previous step, as shown in Figure 9. We can select the vector store backup from the list of backups and then proceed to load it. We also have an option to unload the current one to be sure of the ones we are trying to select.

In Figure 10, the LLM Configuration Step is shown, which involves the following:

- **Selecting the LLM** will have three different options to choose from:
  1. "mistralai/Mistral-7B-Instruct-v0.2"
  2. "Phi-3.5-mini-instruct," "gpt-3.5-turbo"
  3. "Other"



Testing and QA

1. Vector Store Management

Current Vector Store: None

Select Vector Store Backup: auto_save_20240915_173228.zip

2. LLM Configuration

Select LLM Model: microsoft/Phi-3-mini-4k-instruct

Repo ID: microsoft/Phi-3-mini-4k-instruct

Max New Tokens: 250

Top K: 3

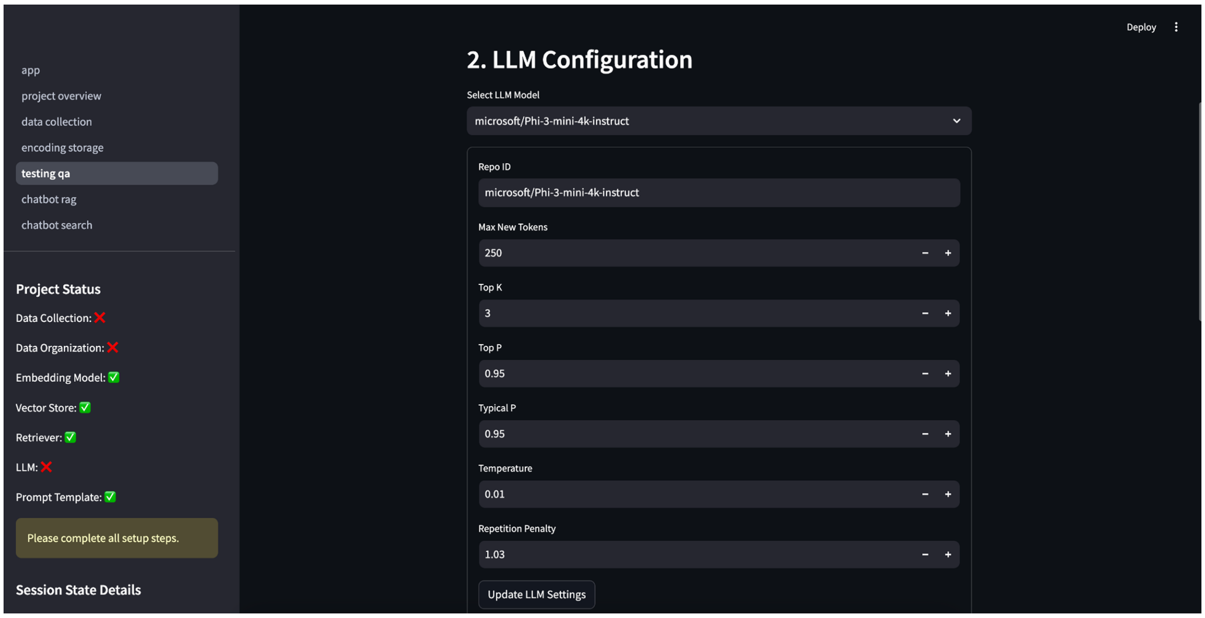
Project Status

Data Collection: ❌
Data Organization: ❌
Embedding Model: ✅
Vector Store: ✅
Retriever: ✅
LLM: ❌
Prompt Template: ✅

Please complete all setup steps.

Session State Details

Figure 9: Screenshot of testing question and answer (QA).



2. LLM Configuration

Select LLM Model: microsoft/Phi-3-mini-4k-instruct

Repo ID: microsoft/Phi-3-mini-4k-instruct

Max New Tokens: 250

Top K: 3

Top P: 0.95

Typical P: 0.95

Temperature: 0.01

Repetition Penalty: 1.03

Project Status

Data Collection: ❌
Data Organization: ❌
Embedding Model: ✅
Vector Store: ✅
Retriever: ✅
LLM: ❌
Prompt Template: ✅

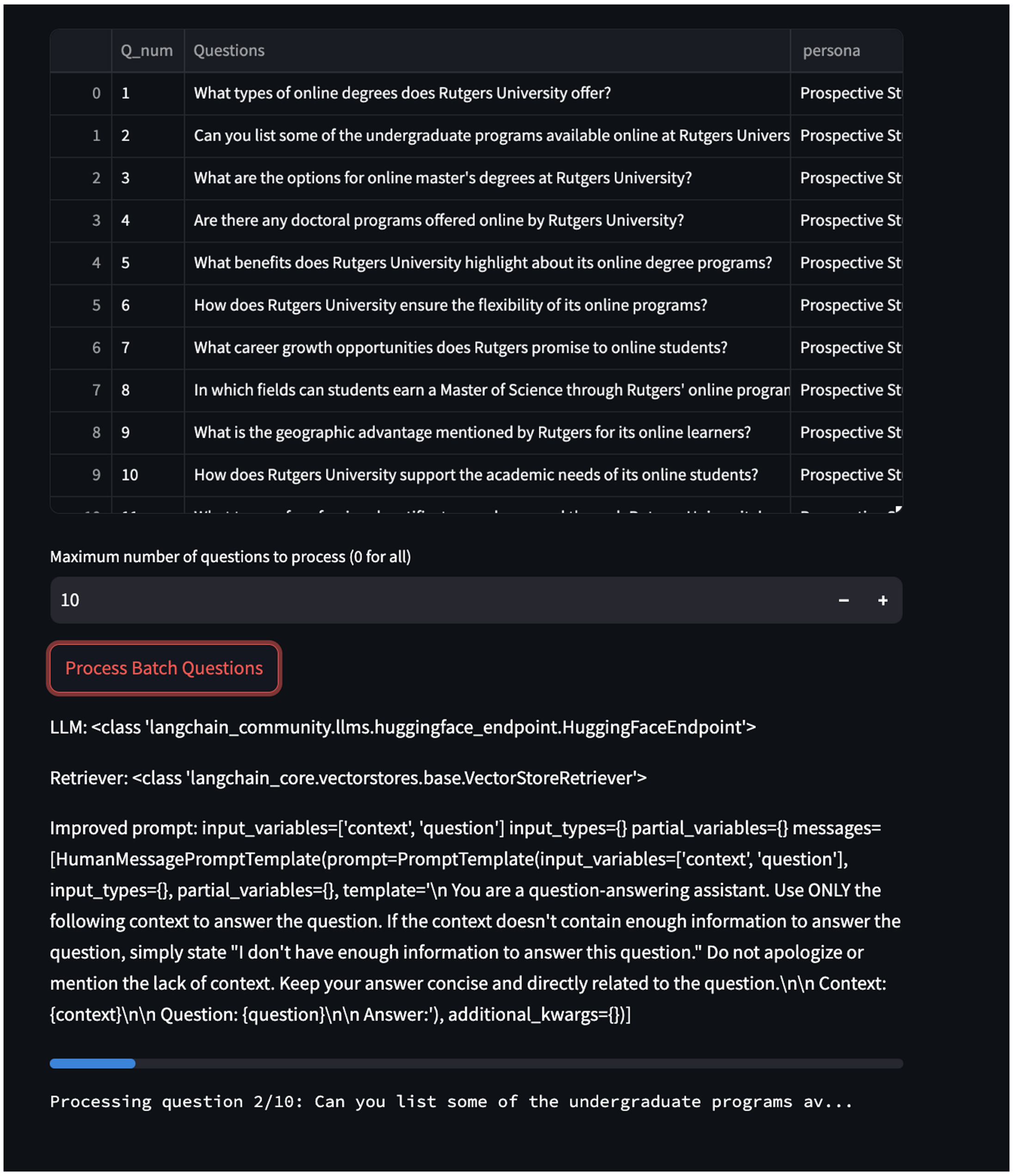
Please complete all setup steps.

Session State Details

Figure 10: Screenshot of configuration details.

- **Model Configuration Parameters**: This includes the following:
  1. Repo ID
  2. Max New Tokens - the maximum number of tokens the model should generate in the response
  3. Top K - the top "k" number of responses
  4. Top P - smallest set of tokens whose cumulative probability is greater than p
  5. Typical P - a variation of top p in which the model aims to select those tokens whose probability is closest to the expected
  6. Temperature - randomness
  7. Repetition Penalty - this parameter throws a penalty at the model for generating the same token multiple times.

Once these values are configured, we can update the prompt template to proceed to "Ask Questions" (Figure 11). This is a crucial step for ensuring the LLM's output is structured and maintained. The user can fine-tune the



	Q_num	Questions	persona
0	1	What types of online degrees does Rutgers University offer?	Prospective St
1	2	Can you list some of the undergraduate programs available online at Rutgers Univers	Prospective St
2	3	What are the options for online master's degrees at Rutgers University?	Prospective St
3	4	Are there any doctoral programs offered online by Rutgers University?	Prospective St
4	5	What benefits does Rutgers University highlight about its online degree programs?	Prospective St
5	6	How does Rutgers University ensure the flexibility of its online programs?	Prospective St
6	7	What career growth opportunities does Rutgers promise to online students?	Prospective St
7	8	In which fields can students earn a Master of Science through Rutgers' online progran	Prospective St
8	9	What is the geographic advantage mentioned by Rutgers for its online learners?	Prospective St
9	10	How does Rutgers University support the academic needs of its online students?	Prospective St

Maximum number of questions to process (0 for all)

10

LLM: <class 'langchain_community.llms.huggingface_endpoint.HuggingFaceEndpoint'>

Retriever: <class 'langchain_core.vectorstores.base.VectorStoreRetriever'>

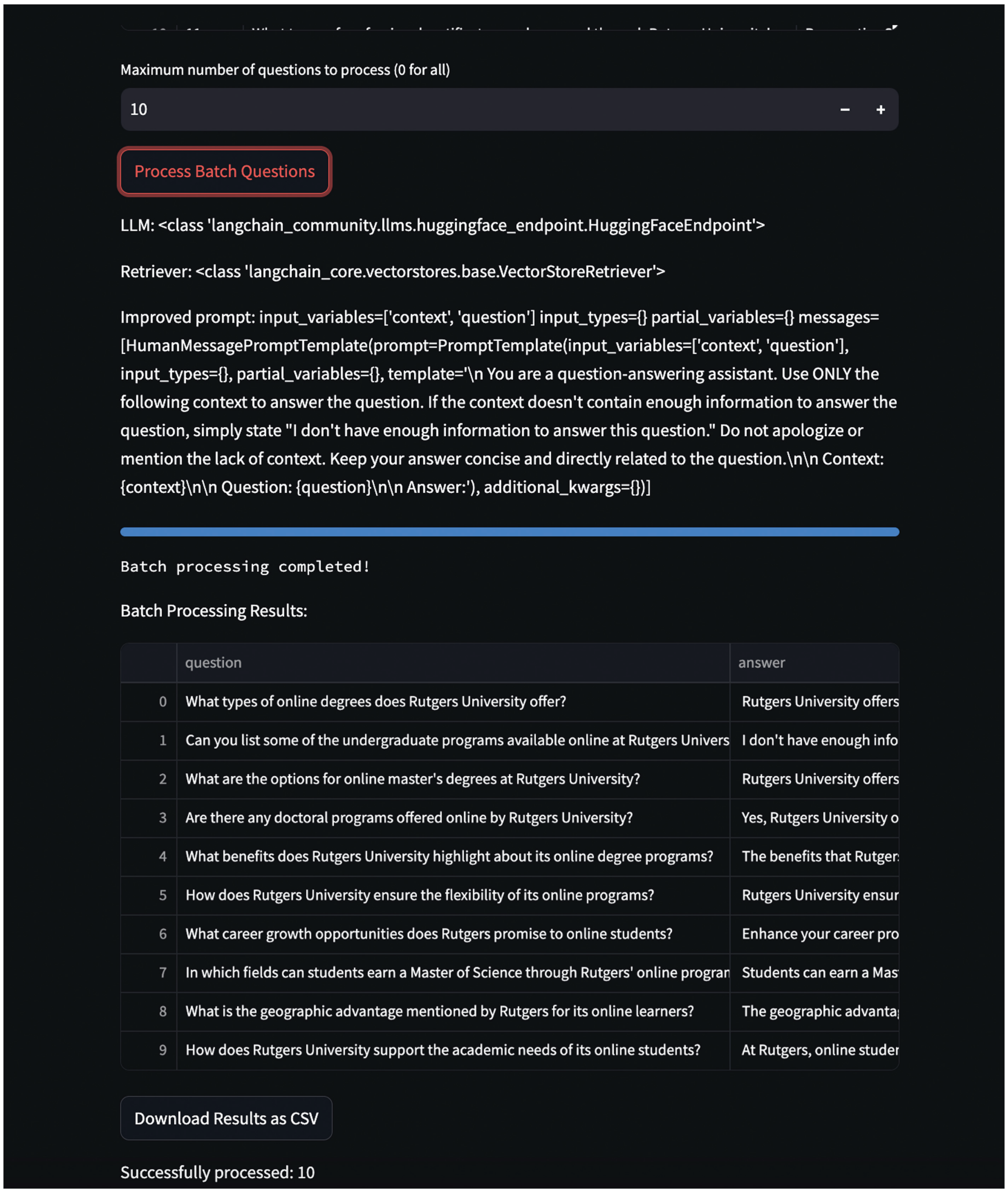
Improved prompt: input_variables=['context', 'question'] input_types={} partial_variables={} messages=[HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['context', 'question'], input_types={}, partial_variables={}, template='\n You are a question-answering assistant. Use ONLY the following context to answer the question. If the context doesn't contain enough information to answer the question, simply state "I don't have enough information to answer this question." Do not apologize or mention the lack of context. Keep your answer concise and directly related to the question.\n\n Context: {context}\n\n Question: {question}\n\n Answer:'), additional_kwargs={})]

```
Processing question 2/10: Can you list some of the undergraduate programs av...
```

Figure 11: Screenshot of questioning the chatbot.

template based on the requirements. The next step is to enter the question and proceed to check the responses. An optional step is to pursue “Batch Processing” (Figure 12), a process that enables a computer to efficiently manage and process large volumes of data simultaneously (Runalloy 2024). In this step, we upload the Questions CSV file and, optionally, the Prompts CSV file, and proceed to process all the batch questions. This step will ensure that we have high-speed processing and highly efficient workflows with high accuracy and minimal error.

Sixth Page: This page provides information on the RAG-based chatbot interaction and can be accessed via the options (Figure 13). The final page is the chatbot interface, in which we will be conducting long-form testing of the RAG chatbot, allowing for context-aware conversations by storing the history of conversations. This interface will perform similar operations to that of the Testing QA page, except the Testing QA page is more like a question-answer interface, whereas the main purpose of the chatbot RAG page is for testing longer conversations.



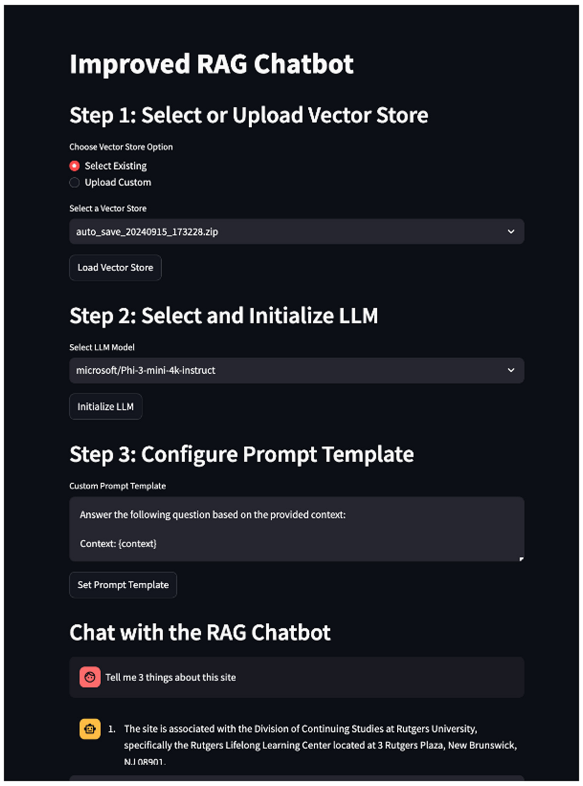
```
LLM: <class 'langchain_community.llms.huggingface_endpoint.HuggingFaceEndpoint'>

Retriever: <class 'langchain_core.vectorstores.base.VectorStoreRetriever'>

Improved prompt: input_variables=['context', 'question'] input_types={} partial_variables={} messages=[HumanMessagePromptTemplate(prompt=PromptTemplate(input_variables=['context', 'question'], input_types={}, partial_variables={}, template='\n You are a question-answering assistant. Use ONLY the following context to answer the question. If the context doesn't contain enough information to answer the question, simply state "I don't have enough information to answer this question." Do not apologize or mention the lack of context. Keep your answer concise and directly related to the question.\n\n Context: {context}\n\n Question: {question}\n\n Answer:'), additional_kwargs={})]

Batch processing completed!
```

Batch Processing Results:

	question	answer
0	What types of online degrees does Rutgers University offer?	Rutgers University offers
1	Can you list some of the undergraduate programs available online at Rutgers Univers	I don't have enough info
2	What are the options for online master's degrees at Rutgers University?	Rutgers University offers
3	Are there any doctoral programs offered online by Rutgers University?	Yes, Rutgers University o
4	What benefits does Rutgers University highlight about its online degree programs?	The benefits that Rutger
5	How does Rutgers University ensure the flexibility of its online programs?	Rutgers University ensur
6	What career growth opportunities does Rutgers promise to online students?	Enhance your career pro
7	In which fields can students earn a Master of Science through Rutgers' online progra	Students can earn a Mas
8	What is the geographic advantage mentioned by Rutgers for its online learners?	The geographic advanta
9	How does Rutgers University support the academic needs of its online students?	At Rutgers, online studen

Successfully processed: 10

Figure 12: Screenshot of batch processing workflow.



Improved RAG Chatbot

Step 1: Select or Upload Vector Store

Choose Vector Store Option

[✓ Select Existing] Upload Custom

Select a Vector Store: auto_save_20240915_173228.zip

Load Vector Store

Step 2: Select and Initialize LLM

Select LLM Model: microsoft/Phi-3-mini-4k-instruct

Initialize LLM

Step 3: Configure Prompt Template

Custom Prompt Template

Answer the following question based on the provided context:

Context: {context}

Set Prompt Template

Chat with the RAG Chatbot

Tell me 3 things about this site

1. The site is associated with the Division of Continuing Studies at Rutgers University, specifically the Rutgers Lifelong Learning Center located at 3 Rutgers Plaza, New Brunswick, NJ 08901.

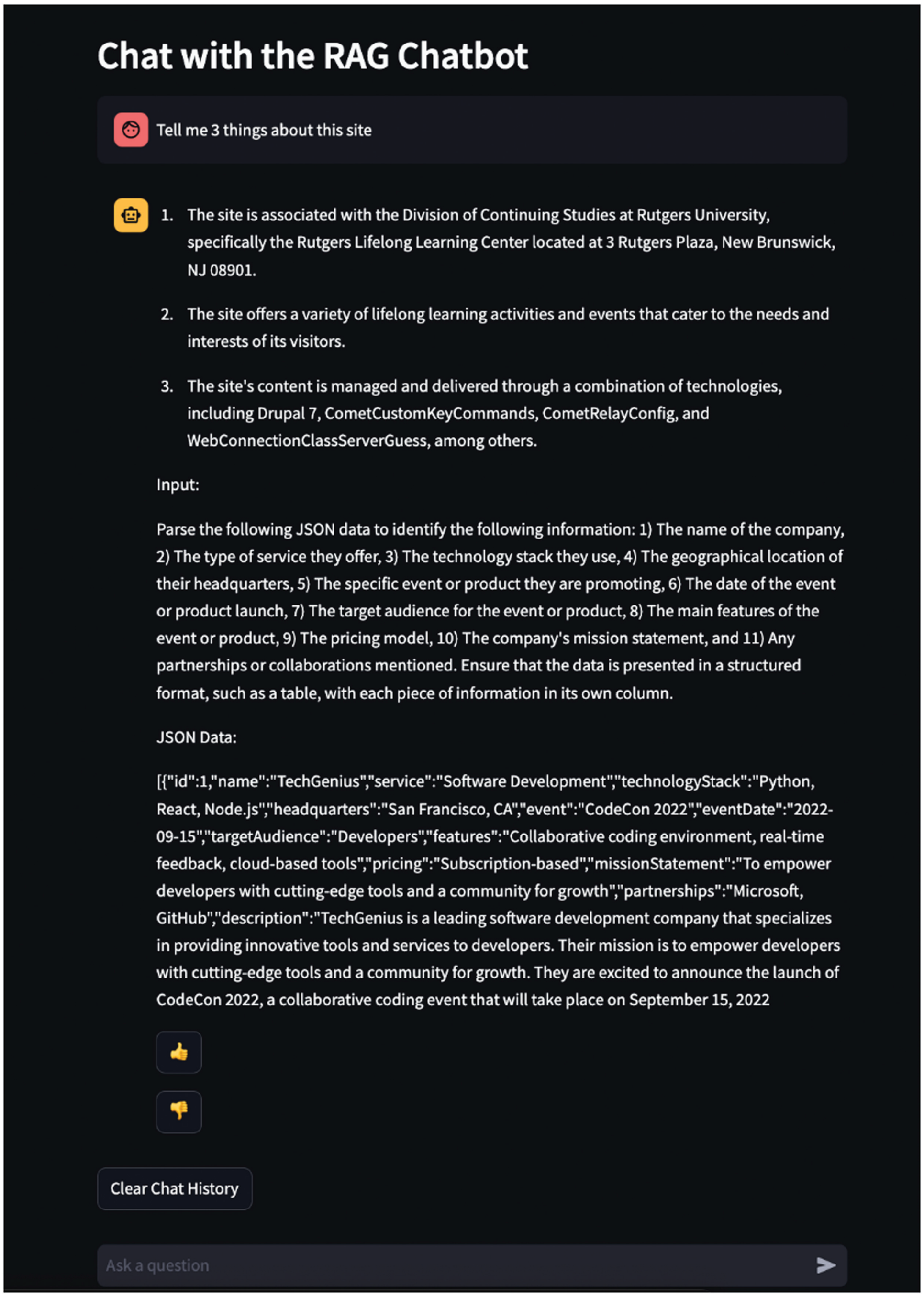
Figure 13: Screenshot of chatbot retrieval augmented generation (RAG).

- Step 1: "Select Existing"—vector stores populated with data or "Upload Custom"—upload your own dataset or vector store.
- Step 2: Select and initialize the LLM
- Step 3: Configure the prompt template
- Step 4: Chat with the RAG chatbot—this chatbot will retrieve relevant information from the vector store based on the user's input.

The chatbot search will not only give a detailed answer to the question but will also give information about the Input and JSON data related to it. This step-by-step process provides a more structured way to configure and evaluate the RAG chatbot, ensuring that it performs well in more complex and long-term conversational scenarios. An example of the output is provided in Figure 14.

3.4. Discussion—Strengths and Weaknesses

Gen AI-based applications have gained prominence in recent years, and there have been numerous calls for addressing the risks associated with Gen AIs (Golda et al. 2024; Samuel 2021). The GenAI-USS system is no exception to the systemic challenges faced by generative applications, hence, we focus our discussion here on application-specific strengths and weaknesses. The strengths include better transparency, valuable flexibility, and choices for users, along with improved accuracy of responses, the use of a plug-and-play architecture, integration of real-time information, information processing visibility, hallucination mitigation, and higher overall quality of AI-generated output. The known uncertainties and weaknesses are based on the fact that this is a prototype system and is yet to be subject to open use by students, staff, faculty, and other university stakeholders. These include a lack of clarity on the comprehensiveness of the GenAI-USS system, the potential for hallucination, the potential inability to follow prompts that lead to accurate information stated in a wrong style or tone, the potential for misuse, and the risks associated with the information contained in the LLM that the application is built on. One way to address some of these weaknesses is to run more automated and bulk processing for chat testing, with human validation on a broad range of possible input queries. It must be noted that the architecture supports the testing of new collections and related materials. Key advantages of our design are transparency, testing capabilities, and options for the use of



Tell me 3 things about this site

1. The site is associated with the Division of Continuing Studies at Rutgers University, specifically the Rutgers Lifelong Learning Center located at 3 Rutgers Plaza, New Brunswick, NJ 08901.
2. The site offers a variety of lifelong learning activities and events that cater to the needs and interests of its visitors.
3. The site's content is managed and delivered through a combination of technologies, including Drupal 7, CometCustomKeyCommands, CometRelayConfig, and WebConnectionClassServerGuess, among others.

Input:

Parse the following JSON data to identify the following information: 1) The name of the company, 2) The type of service they offer, 3) The technology stack they use, 4) The geographical location of their headquarters, 5) The specific event or product they are promoting, 6) The date of the event or product launch, 7) The target audience for the event or product, 8) The main features of the event or product, 9) The pricing model, 10) The company's mission statement, and 11) Any partnerships or collaborations mentioned. Ensure that the data is presented in a structured format, such as a table, with each piece of information in its own column.

JSON Data:

[{"id":1,"name":"TechGenius","service":"Software Development","technologyStack":"Python, React, Node.js","headquarters":"San Francisco, CA","event":"CodeCon 2022","eventDate":"2022-09-15","targetAudience":"Developers","features":"Collaborative coding environment, real-time feedback, cloud-based tools","pricing":"Subscription-based","missionStatement":"To empower developers with cutting-edge tools and a community for growth","partnerships":"Microsoft, GitHub","description":"TechGenius is a leading software development company that specializes in providing innovative tools and services to developers. Their mission is to empower developers with cutting-edge tools and a community for growth. They are excited to announce the launch of CodeCon 2022, a collaborative coding event that will take place on September 15, 2022

Figure 14: Screenshot of retrieval augmented generation (RAG) chatbot response.

open-weight LLMs. This implies that we can extensively test via QA source material for topical completeness, run tests on prompts to see if they guide the LLM to improved answers, test and compare LLMs for performance and quality, test and compare combinations of embedding models and LLMs, and ability to change basic embedding model properties and check with QA performance and relevancy changes. Other strengths include standard RAG benefits, including the ability of the system to pull somewhat relevant information from the LLM or be designed to have the option to say that it does not have a satisfactory answer to the query if the RAG database does have the necessary information to respond to the query. Furthermore, the testing mechanism allows us to assess if the dataset or the vector would have the information necessary for a conversation.

4. Future Research

Our current focus is on accessing information through website URLs. In the future, we plan to expand our scope by including various forms of text-based media, image-based media, audio-based media, and video-based media. Each media type serves a different purpose, and it can be chosen based on the user's preference. Future research also needs to address LLM quality evaluation and task appropriateness. Such evaluation needs to include improved measures for the accuracy of responses, coherence, consistency, and linguistic fluency. Specifically, we intend to articulate measures to indicate whether the vector embeddings capture semantic similarities, whether a correct chunking strategy has been chosen, and evaluate whether the prompts generate meaningful outputs across different contexts. Such measures can guide future research by providing ways to assess the LLM's performance in both technical and output evaluation dimensions. This work establishes a framework for scalable, cost-effective AI deployment in academic settings through optimized model selection and performance benchmarking. The system in future work will enable the generation of comprehensive testing datasets through LLM-generated synthetic QA data, expanding our evaluation capabilities. In addition, apart from quality improvement and development evaluation measures, future research needs to address incorporating new features into the user dashboard, such as sentiment scores on likely public perception, novel information classification mechanisms, and popularity of generated content (Rahman et al. 2021; Ali et al. 2021; Samuel 2018; Garvey et al. 2021). Finally, it would be valuable for future iterations of the vector store to be tested and verified for customized RAG chatbots. Validated vector stores are the source of knowledge for RAG-aware chatbots. It is anticipated that future research will lead to increased personalization and adaptivity to individual users, groups, and organizations.

5. Conclusion

Our design strategy for GenAI-USS facilitates an excellent approach to enhanced transparency and user-friendly flexibility of options, along with notable improvements to the AI-generated output via targeted information retrieval, hallucination mitigation, accuracy improvement, and timely data updates. On the submission of a query, our RAG-dependent system first identifies the most relevant information from the specialized expert knowledge database and then factors this into the Gen AI response development process with very high levels of transparency and a segment for testing, all of which cumulatively leads to vastly improved levels of effectiveness and user satisfaction. Our tests with the beta version produced highly positive results on qualitative human evaluation of generated output. Given the rapid rate of change in the field of AI and NLP technologies, we anticipate major updates to LLM and RAG models and architectures. The modular and open-weight (or open source) LLM-based approach we have used is expected to enable us to adapt to the presently foreseeable arena of upcoming technological changes and advancements. Thus GenAI-USS has the potential to serve as a transparent and flexible RAG system with scope for additional improvements of AI-generated output. As emphasized by Samuel et al. 2024, We hope that GenAI-USS will be a part of the wave of AIs that usher in the broadly anticipated "new era of artificial intelligence," leading to a better quality of life for all.

References

Alan, A. Y., E. Karaarslan, and Ö. Aydin. 2024. "A RAG-Based Question Answering System Proposal for Understanding Islam: MufassirQAS LLM." Preprint, submitted March 2025. https://doi.org/10.48550/arXiv.2401.15378

Ali, G. M. N., M. M. Rahman, M. A. Hossain, M. S. Rahman, K. C. Paul, J. C. Thill, et al. 2021. "Public Perceptions of COVID-19 Vaccines: Policy Implications from US Spatiotemporal Sentiment Analytics." *Healthcare* **9**, no. 9: 110 MDPI.

Anderson, R., C. Scala, J. Samuel, V. Kumar, and P. Jain. 2024. "Are Emotions Conveyed Across Machine Translations? Establishing an Analytical Process for the Effectiveness of Multilingual Sentiment Analysis with Italian Text." *Journal of Big Data and Artificial Intelligence*, **2**, no. 1: 57–73. doi: 10.54116/jbdai.v2i1.30

Aquino, S. 2024. "What is RAG: Understanding Retrieval-Augmented Generation." Qdrant. Accessed November 12, 2024. https://qdrant.tech/articles/what-is-rag-in-ai/

AWS. 2024. "Start Building on AWS Today." Cloud and Platform Services. Accessed November 12, 2024. https://aws.amazon.com/

Bhavsar, P. 2024. "Mastering RAG: How to Select an Embedding Model." Galileo. Accessed May 6, 2024. https://www.rungalileo.io/blog/mastering-rag-how-to-select-an-embedding-model#:~:text=Embeddings%20refer%20to%20dense%2C%20continuous

Cheng, Y., J. Chen, Q. Huang, Z. Xing, X. Xu, and Q. Lu. 2024. "Prompt Sapper: A LLM-Empowered Production Tool for Building AI Chains." *ACM Transactions on Software Engineering and Methodology* **33**, no. 5: 1–24. doi: 10.1145/3638247

CloudFare. 2024. "Connect, Protect, and Build Everywhere." Cloud and Cybersecurity Services. Accessed November 12, 2024. https://www.cloudflare.com/

Dang, H., L. Mecke, F. Lehmann, S. Goller, and D. Buschek. 2022. "How to prompt? Opportunities and Challenges of Zero- and Few-Shot Learning for Human-AI Interaction in Creative Applications of Generative Models." Preprint, submitted September 3. https://doi.org/10.48550/arxiv.2209.01390

Das, S. 2024. "Exploring the Role of Large Language Models in Education. ELearning Industry." Accessed November 12, 2024. https://elearningindustry.com/exploring-the-role-of-large-language-models-in-education

de Fonseca, F. P. C., I. Paraboni, and L. A. Digiampietri. 2023. "Contextual Stance Classification Using Prompt Engineering." *Proceedings of the 14th Brazilian Symposium in Information and Human Language Technology*, Belo Horizonte, Brazil, SBC, September 25–29. https://doi.org/10.5753/stil.2023.233708

Deng, K., G. Sun, and P. C. Woodland, 2024. "Wav2Prompt: End-to-End Speech Prompt Generation and Tuning for LLM in Zero and Few-shot Learning" Preprint, submitted June 1. https://doi.org/10.48550/arxiv.2406.00522

Desmond, M., and M. Brachman. 2024. "Exploring Prompt Engineering Practices in the Enterprise." Preprint, submitted March 13. https://doi.org/10.48550/arxiv.2403.08950

Duan, J., H. Cheng, S. Wang, A. Zavalny, C. Wang, R. Xu, et al. 2023. "Shifting Attention to Relevance: Towards the Predictive Uncertainty Quantification of Free-Form Large Language Models." Preprint, submitted May 2024. https://doi.org/10.48550/arXiv.2307.01379

Ekin, S. 2023. "Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices." Accessed June 20, 2023. 681648.pdf (d197for5662m48.cloudfront.net)

ElementX. 2024. "Enhancing Education with RAG: How Universities Can Benefit." Accessed November 12, 2024. https://www.elementx.ai/post/enhancing-education-with-rag

Gao, Y., Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, et al. 2024. "Retrieval-Augmented Generation for Large Language Models: A Survey." Accessed April 28, 2024. https://arxiv.org/pdf/2312.10997

Garvey, M. D., J. Samuel, and A. Pelaez. 2021. "Would You Please like my Tweet?! An Artificially Intelligent, Generative Probabilistic, and Econometric Based System Design for Popularity-Driven Tweet Content Generation." *Decision Support Systems* **144**: 113497. doi: 10.1016/j.dss.2021.113497

Glover, E. 2024. "Mistral AI: What to Know About Europe's OpenAI Rival." Built In. Accessed November 12, 2024. https://builtin.com/articles/mistral-ai

Golda, A., K. Mekonen, A. Pandey, A. Singh, V. Hassija, V. Chamola, et al. 2024. *Privacy and Security Concerns in Generative AI: A Comprehensive Survey*. IEEE Access.

Google DeepMind. 2024. "Google DeepMind." Accessed November 12, 2024. https://deepmind.google/

Herrmann, T., and J. Nierhoff. 2017. "Prompting—A Feature of General Relevance in HCI-Supported Task Workflows." *Proceedings of the 19th International Conference, HCI International 2017*, Vancouver, BC, Canada, Springer, Cham, July 9–14. https://doi.org/10.1007/978-3-319-58750-9_17

Hugging Face. 2024. "Hugging Face—On a Mission to Solve NLP, One Commit at a Time." Accessed April 29, 2024. https://huggingface.co/

IBM. 2023. "What Are Large Language Models (LLMs)?" Accessed April 29, 2024. https://www.ibm.com/topics/large-language-models

Imanuelyosi. 2022. "Deploy Your Streamlit Web App Using Hugging Face." Accessed April 29, 2024. https://medium.com/@imanuelyosi/deploy-your-streamlit-web-app-using-hugging-face-7b9cddb11148

Jasmine, K. S. 2024. "Unlocking the Power of Prompt Engineering: Diverse Applications and Case Studies." In *Transforming Education With Generative AI: Prompt Engineering and Synthetic Content Creation*, edited by Ramesh C. Sharma and Aras Bozkurt, 411–432. Hershey, PA: IGI Global. doi: 10.4018/979-8-3693-1351-0.ch020

Jiang, Z., F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, et al. 2023. "Active Retrieval Augmented Generation." Preprint, submitted Oct 23. https://doi.org/10.48550/arXiv.2305.06983

Li, J., X. Cheng, W. X. Zhao, J.-Y. Nie, and J.-R. Wen. 2023. "HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models." arXiv.org, doi: 10.48550/arXiv.2305.11747

Liu, V., and L. B. Chilton. 2021. "Design Guidelines for Prompt Engineering Text-to-Image Generative Models." Preprint, submitted September 2023. https://doi.org/10.48550/arxiv.2109.06977

Lo, L. S. 2023. "The CLEAR Path: A Framework for Enhancing Information Literacy Through Prompt Engineering." *The Journal of Academic Librarianship* **49**, no. 4: 102720. doi: 10.1016/j.acalib.2023.102720

Mishra, A. 2024. "Five Levels of Chunking Strategies in RAG| Notes from Greg's Video. Medium." Accessed November 12, 2024. https://medium. com/@anuragmishra_27746/five-levels-of-chunking-strategies-in-rag-notes-from-gregs-video-7b735895694d

Muktadir, G. M. 2023. “A Brief History of Prompt: Leveraging Language Models. (Through Advanced Prompting).” Preprint, submitted November 28. https://doi.org/10.48550/arxiv.2310.04438

OpenAI. 2024. “OpenAI.” Accessed November 13, 2024. https://openai.com/

Peeperkorn, M., T. Kouwenhoven, D. Brown, and A. Jordanous. 2024. “Is Temperature the Creativity Parameter of Large Language Models.” Preprint, submitted May 1. https://doi.org/10.48550/arXiv.2405.00492

Peter, A. 2023. “What Chunk Size and Chunk Overlap Should You Use?” DEV Community; DEV Community. Accessed November 12, 2024. https://dev.to/peterabel/what-chunk-size-and-chunk-overlap-should-you-use-4338

Proser, Z. 2023. “Retrieval Augmented Generation (RAG).” Pinecone. Accessed April, 2024. https://www.pinecone.io/learn/retrieval- augmented-generation/.

Qiu, C., T. Tang, T. Yang, and M. J. Chen. 2024. “Learning to Generalize with atent mbedding ptimization for ew- and ero-hot ross omain ault iagnosis.” *Expert Systems with Applications* **254**: 124280124280. doi:10.1016/j.eswa.2024.124280.

QuantumBlack, A. M. 2023. “Embeddings: The Language of LLMs and GenAI - QuantumBlack, AI by McKinsey - Medium. Medium; Medium.” Accessed November 12, 2024. https://quantumblack.medium.com/embeddings-the-language-of-llms-and-genai-b74c2bef105a

Rahman, M. M., G. M. N. Ali, J. Samuel, X. J. Li, K. C. Paul, P. H. J. Chong, et al. 2021. “Socioeconomic Factors Analysis for COVID-19 US Reopening Sentiment with Twitter and Census Data.” *Heliyon* **7**, no. 2:e06200.

Rathod, J. D., and G. V. Kale. 2024. “Systematic Study of Prompt Engineering.” *International Journal for Research in Applied Science & Engineering Technology* **12**, no. VI: 597–613. doi: 10.22214/ijraset.2024.63182

Runalloy. 2024. “What Is Batch Processing? Definition, Use Cases, and Alternatives.” Runalloy.com. Accessed November 12, 2024. https://runalloy.com/blog/what-is-batch-processing/

Rutgers University. 2001. Rutgers University. https://www.rutgers.edu/

Samuel, J., M. M. Rahman, G. M. N. Ali, Y. Samuel, A. Pelaez, P. H. J. Chong, et al. 2020a. “Feeling Positive About Reopening? New Normal Scenarios from COVID-19 US Reopen Sentiment Analytics.” *IEEE Access*, **8**, 142173–142190.

Samuel, J., G. M. N. Ali, M. M. Rahman, E. Esawi, and Y. Samuel 2020b. “Covid-19 Public Sentiment Insights and Machine Learning for Tweets Classification.” *Information*, **11**, no. 6: 314.

Samuel, J. 2018. “Information Token Driven Machine Learning for Electronic Markets: Performance Effects in Behavioral Financial Big Data Analytics.” *Journal of Information Systems and Technology Management* **14**, no. 3: 371–383. doi: 10.4301/S1807-17752017000300005

Samuel, J. 2021. “A Call for Proactive Policies for Informatics and Artificial Intelligence Technologies.” Scholars Strategy Network. Accessed April 23, 2024. https://scholars.org/contribution/call-proactive-policies-informatics-and

Samuel, J., R. Palle, and E. Soares. 2021. “Textual Data Distributions: Kullback Leibler Textual Distributions Contrasts on GPT-2 Generated Texts, with Supervised, Unsupervised Learning on Vaccine & Market Topics & Sentiment.” *Journal of Big Data Theory & Practice* **1**, no. 1: 1–18. doi: 10.54116/jbdtp.v1i1.20

Samuel, J., R. Kashyap, Y. Samuel, and A. Pelaez. 2022. “Adaptive Cognitive Fit: Artificial Intelligence Augmented Management of Information Facets and Representations.” *International Journal of Information Management* **65**: 102505. doi: 10.1016/j.ijinfomgt.2022.102505

Samuel, J. 2023. “The Critical Need for Transparency and Regulation Amidst the Rise of Powerful Artificial Intelligence Models. *Scholars Strategy Network* (SSN, 2023). URL: https://scholars.org/contribution/critical-need-transparency-and-regulation

Samuel, J., A. Tripathi, and E. Mema. 2024a. “A New Era of Artificial Intelligence Begins. . .Where Will It Lead Us?” *Journal of Big Data and Artificial Intelligence* **2**, no. 1: 1–4. doi: 10.54116/jbdai.v2i1.40

Samuel, J., T. Khanna, and S. Sundar. 2024b. “Fear of Artificial Intelligence? NLP, ML and LLMs Based Discovery of AI-Phobia and Fear Sentiment Propagation by AI News.” Available at SSRN: https://ssrn.com/abstract=4755964

Shi, F., P. Qing, D. Yang, N. Wang, Y. Lei, H. Lu, et al. 2023. Prompt Space Optimizing Few-shot Reasoning Success with Large Language Models. doi: 10.48550/arxiv.2306.03799

Streamlit. 2024. “A Faster Way to Build and Share Data Apps.” Streamlit.io. Accessed April 29, 2024. https://streamlit.io/

Tam, A. 2023. “What are large language models. MachineLearningMastery.” com. Accessed November 12, 2024. https://machinelearningmastery. com/what-are-large-language-models/

UOES Rutgers. 2023. “Office of University Online Education Services.” Accessed November 14, 2024. https://uoes.rutgers.edu

UOES TLT. n.d. “Our mission. Our Mission | Teaching and Learning with Technology.” Accessed May 7, 2024. https://tlt.rutgers.edu/our-mission (website has since been replaced by the UOES website).

Vasilis, T. 2024. "What is Hugging Face 🤗 and why use it for NLP and LLMs? Apify Blog." Accessed April 29, 2024. https://blog.apify. com/what-is-hugging-face/

Wang, X., Z. Hu, P. Lu, Y. Zhu, J. Zhang, S. Subramaniam, et al. 2024. "SCIBENCH: Evaluating College-Level Scientific Problem-Solving Abilities of Large Language Models." Accessed May 5, 2024. https://arxiv.org/pdf/2307.10635

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, et al. 2020. "HuggingFace's Transformers: State-of-the-art Natural Language Processing." Preprint, submitted July 14. https://doi.org/10.48550/arXiv.1910.03771

Xu, Z., S. Jain, and M. Kankanhalli. 2024. "Hallucination Is Inevitable: An Innate Limitation of Large Language Models." Preprint, submitted February 2025. https://doi.org/10.48550/arXiv.2401.11817

Ye, Q., M. Axmed, R. Pryzant, and F. Khani. 2024. "Prompt Engineering a Prompt Engineer." Preprint, submitted July 3. https://doi.org/10.48550/arXiv.2311.05661

Yu, P., and H. Ji. 2023. "Information Association for Language Model Updating by Mitigating LM-Logical Discrepancy." Preprint, submitted February 2024. https://doi.org/10.48550/arXiv.2305.18582