RoostAI: Building a University-Centric Chatbot with Retrieval Augmented Generation

Vansh Nagpal
Computer Science and Engineering
vnagpal@email.sc.edu

Nitin Gupta
Computer Science and Engineering
niting@email.sc.edu

***Abstract*—In the dynamic academic landscape, efficient access to university-specific information is paramount for students, staff, and visitors. At the University of South Carolina (USC), information resources are often scattered across multiple platforms, posing challenges for users seeking accurate and timely data. This paper introduces *RoostAI*, a Retrieval-Augmented Generation (RAG) based chatbot designed to serve the USC community by providing precise, context-aware responses to natural language queries. *RoostAI* leverages a USC-specific knowledge base, complemented by advanced web-scraping, natural language processing, and machine learning techniques, to retrieve and generate relevant information efficiently. Our evaluation demonstrates the system's superior performance in terms of semantic and factual accuracy compared to standalone large language models (LLMs), highlighting its potential to enhance the USC campus experience through a user-friendly conversational interface.**

***Index Terms*—Large Language Models (LLMs), Retrieval Augmented Generation (RAG), Chatbots**

I. Introduction

In the modern academic environment, digital tools such as Google Search play a crucial role in facilitating access to essential university information. Recently, Large Language Models(LLMs) have begun to serve as a proxy to Google Search [9], often being the first tool many use to inform themselves. However, LLMs avoid providing specific answers to questions not readily available in their training data, such as details specific to a university. As a result, at many universities, like the University of South Carolina (USC), students, faculty, and staff navigate fragmented resources across the web with Google Search, leading to inefficiencies in information retrieval. This fragmentation is particularly challenging for new students and visitors, who may struggle to quickly obtain accurate answers to their queries.

To address these challenges of fragmentation while providing the conversationality of LLMs, we propose *RoostAI*, a Retrieval-Augmented Generation (RAG) based chatbot specifically designed for the USC community. Unlike traditional LLMs, which rely solely on pre-trained knowledge and may struggle with domain-specific queries, the RAG framework enhances response accuracy by combining retrieval mechanisms with generative capabilities. This approach allows *RoostAI* to draw on a comprehensive, up-to-date USC-specific knowledge base, ensuring that users receive precise and contextually relevant answers.

The choice of RAG over simply fine-tuning existing LLMs is driven by the need for scalable and continuous integration of new information. Fine-tuning requires substantial computational resources and periodic retraining to incorporate fresh data, while RAG systems can dynamically retrieve and integrate information without necessitating model updates at a much lower computational cost. By leveraging advanced web-scraping techniques and state-of-the-art natural language processing (NLP) algorithms, *RoostAI* efficiently collects, indexes, and retrieves information from diverse USC web domains.

RoostAI's architecture is built around two main components: a data pipeline that processes and stores information in a vector database for quick retrieval, and a query pipeline that handles user interactions, retrieves relevant data, and generates responses using an LLM, as illustrated in Figure 1. The system is designed for high accessibility and scalability through cloud hosting, making it readily available to users both on and off campus.

In this paper, we detail the *RoostAI* system design and evaluate its performance against leading LLMs using a dataset curated from USC's frequently asked questions (FAQs). Our results indicate that *RoostAI* outperforms its counterparts in terms of semantic accuracy and response relevance, while responsibly declining to answer queries when confidence is low. These findings underscore the potential of RAG systems in addressing the unique informational needs of university environments and their broader applicability in similar domains. In addition, we evaluate 4 different chunking strategies with known RAG metrics and report our findings about which chunking strategy is the most optimal for our use case.

II. Related Works

Recent advances in Retrieval-Augmented Generation (RAG), introduced by Patrick et al. [16], have demonstrated its effectiveness in enhancing LLM responses with external knowledge. Huang et al. [11] provides a comprehensive survey of RAG frameworks, detailing various retrieval and generation techniques that inform our approach. The integration of vector databases for efficient information retrieval has been explored by Lewis et al. [14] in their seminal work on dense passage retrieval for open-domain question answering. Xie et al. [27] further advance this field with WeKnow-RAG, demonstrating

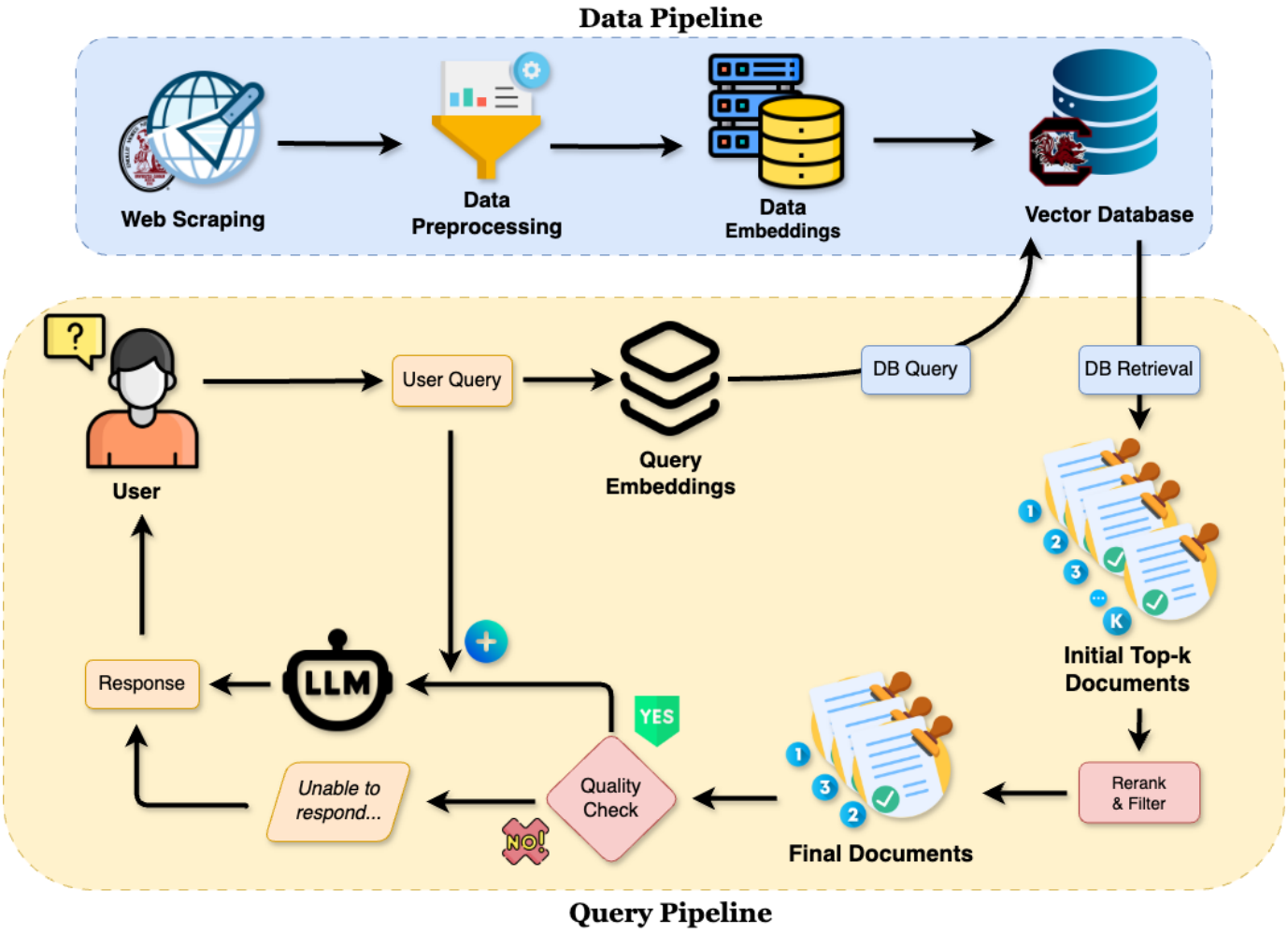

Fig. 1. The *RoostAI* workflow illustrating the dual-pipeline architecture consisting of the Data Pipeline and Query Pipeline. The Data Pipeline encompasses web scraping, preprocessing, embedding generation, and vector database storage. The Query Pipeline involves user query processing, database retrieval, reranking with quality checks, and response generation via LLM, ultimately delivering precise information to users about USC.

the potential of hybrid approaches combining web search capabilities with knowledge graph integration.

The implementation of chatbots in university settings has evolved significantly, moving from simple rule-based systems to more sophisticated RAG-based solutions. Winkler & Söllner [25] provide a foundational analysis of chatbots in education, emphasizing the need for context-aware systems that can handle complex student queries. Recent implementations have shown promising results: Neupane et al. [19] developed BARKPLUG V.2 at Mississippi State University, demonstrating high performance in generating accurate responses about campus resources. Similarly, Thway et al. [24] implemented Professor Leodar at Nanyang Technological University, showing how RAG-based chatbots can provide personalized guidance and enhance student engagement.

Educational chatbots are being developed to address specific user needs within university communities. Burgan et al. [5] developed RamChat at Shepherd University to help students navigate the student handbook, while Saha & Saha [21] created a specialized system for international graduate students. These implementations demonstrate the versatility of RAG-based approaches in addressing diverse educational needs. Yuhao et al. [6], through their work on EduChat, established important benchmarks for maintaining accuracy and relevance in educational contexts.

The development of effective RAG-based university chatbots faces several technical challenges. Maryamah et al. [17] highlight the importance of optimizing retrieval methods and improving response generation, particularly in evaluating different embedding models and search methods. These challenges inform our approach to system design and implementation.

Our contributions, with *RoostAI*, advance the state-of-the-art in several ways:

1) We implement a dynamic RAG framework specifically optimized for university content, with continuous knowledge updates through automated web scraping of USC

domains.

2) Our system introduces a novel preprocessing pipeline designed for university content, incorporating domain-specific metadata tagging and relevance scoring.
3) We provide a comprehensive evaluation framework that combines traditional metrics with university-specific benchmarks to evaluate the quality of generated respones as well as retrieval methods
4) Our cloud-based deployment strategy offers insights into scalability and accessibility considerations for university-scale information systems.

These contributions address key limitations in existing solutions while providing a blueprint for implementing similar systems at other educational institutions.

III. System Design

Our university chatbot's architecture is divided into two primary pipelines: the Data Pipeline and the Query Pipeline, as illustrated in Figure 1. This design leverages state-of-the-art techniques in information retrieval and large language models (LLM) to provide precise and context-aware responses to user queries about the University of South Carolina (USC).[1]

A. Data Pipeline

The Data Pipeline is responsible for acquiring, processing, and storing information relevant to USC. Initially, web scraping techniques are employed to extract data from official university sources (ending in *sc.edu*), ensuring a comprehensive collection of documents encompassing various domains such as academics, campus life, and current events as well as various colleges such as the *Molinaroli College of Engineering and Computing* or the *Darla Moore School of Business*. Subsequently, the extracted data undergoes pre-processing, which involves cleaning, and structuring the extracted text. Following this, we **chunk** the text into documents which will be retrieved by the RAG pipeline for generating an informed response.

We explore a few different chunking strategies with the LlamaIndex Framework [23], including **(1)** Fixed-token chunking, **(2)** Sentence-splitting chunking [28], and **(3, 4)** two versions of Semantic chunking [3]. Fixed-token chunking entails splitting the text corpus into chunks of length 1024 tokens, sentence-splitting chunking entails considering each sentence in the corpus as a separate chunk, and semantic chunking considers variable-size chunks of semantically related information. This similarity is based on BERTScore similarity [29] and a dissimilarity percentile threshold which determines the strictness of the similarity requirement (a higher threshold results in a laxer requirement and larger chunks, while a lower threshold results in a stricter requirement and smaller chunks). We consider two thresholds: 50% and 95%. After chunking our corpus into documents, we generate the embeddings using a pre-trained Sentence Transformer. This step transforms the textual

[1]Our code and documentation for replicating results can be viewed at https://github.com/csce585-mlsystems/RoostAI

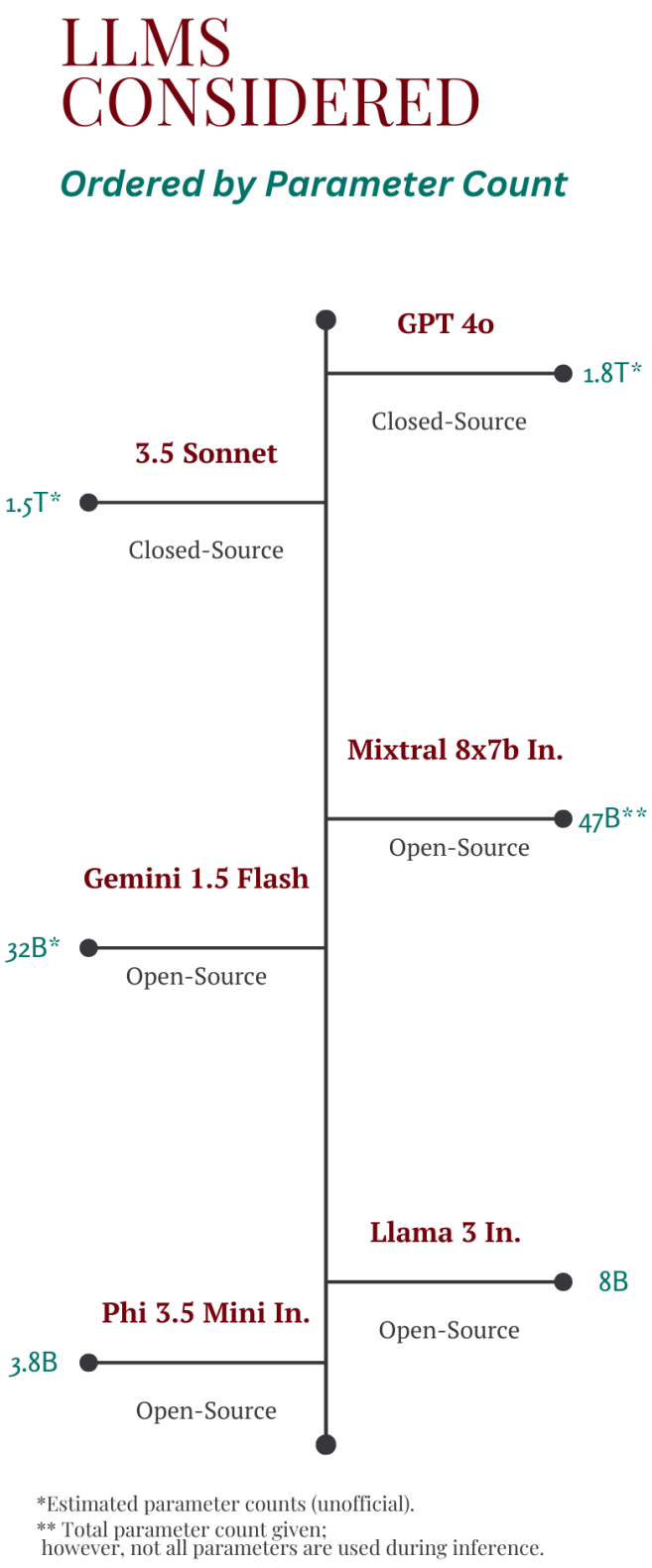

Fig. 2. Different LLMs considered for evaluation *RoostAI*. Chosen LLMs span a large parameter count range, allowing a comprehensive evaluation of the effect of the parameter count in response generation.

data into a vector representation, capturing semantic meaning. The resulting embeddings are stored in a vector database, facilitating efficient retrieval during query processing.

B. Query Pipeline

The Query Pipeline is designed to handle user interactions seamlessly, leveraging advanced NLP techniques and a robust infrastructure to ensure precise and contextually relevant responses. The stages involved are as follows:

1) **User Query Handling**: Upon receiving a user query, the system processes it using the SentenceTransformer library [20], which provides state-of-the-art pre-trained models for generating embeddings. The query is cleaned and transformed into an embedding, ensuring consistency in semantic representation with the data pipeline.
2) **Database Query and Retrieval**: The generated query embedding is used to perform a nearest-neighbor search in the vector database managed by Chroma [1], a high-performance vector database system. This search retrieves the top-k documents that are most semantically similar to the user's question, leveraging the cosine similarity metric for efficient retrieval.
3) **Reranking and Filtering**: The initially retrieved documents are reranked using a Cross-Encoder model from the Hugging Face Transformers library [26]. This model evaluates the relevance of each document with respect to the query, scoring them based on semantic alignment. Documents that meet a predefined threshold, specified in the configuration settings, are selected for further processing.
4) **Quality Check**: A Quality Checker component assesses the overall relevance of the filtered documents. This step ensures that the information provided to the Large Language Model (LLM) is of sufficient quality to generate a reliable response. The quality check is based on a weighted scoring system that aggregates document scores, considering both quantity and quality.
5) **LLM Response Generation**: If the quality check is passed, the refined set of documents is used to construct a prompt for the LLM. The LLM Manager, utilizing the Hugging Face Inference API [26], generates a natural language response that encapsulates the knowledge extracted from the documents. This response is tailored to provide accurate and informative answers to the user's query.
6) **User Interaction**: If the system is unable to generate a confident response, it informs the user accordingly and suggests reformulating the query. Otherwise, the response is delivered to the user, completing the interaction loop.The entire process is orchestrated within an asynchronous framework using Python's asyncio library, ensuring efficient handling of multiple concurrent queries.

This systematic workflow, supported by a robust backend architecture, enables the RoostAI chatbot to deliver precise and contextually relevant answers while maintaining the integrity and quality of information sourced from USC's publicly available documents from the web. The use of advanced embeddings, vector retrieval, and cross-encoder ranking ensures that user queries are addressed with high accuracy and semantic fidelity.

C. Technologies: Data Procurement and System Deployment

Our *RoostAI* system employs a diverse array of technologies to facilitate both data procurement and query processing, ensuring efficient information retrieval and response generation.

a) Data Procurement: To scrape the USC web domain, we utilized *Playwright* [18], which offers robust capabilities for handling dynamic web pages that load asynchronously. This enabled us to gather a comprehensive dataset from various USC-affiliated websites, forming the basis of our knowledge base. Document chunking was performed using the *LlamaIndex* [23] framework, which supports multiple chunking strategies, including fixed-token and semantic chunking, to optimize information retrieval. For the remainder of our data preprocessing, we used native Python libraries, such as *requests*. We faced a few difficulties with web scraping, included page request time-outs and failures due to redirecting links. For the scope of this project, we ignored these problems, and procured a dataset consisting of $\sim$80k HTML files.

b) Query Pipeline Technologies: The query pipeline leverages several state-of-the-art tools and models to process user queries and generate accurate responses:

- **SentenceTransformer**: For generating query embeddings, we utilized the *all-MiniLM-L6-v2* model provided by the SentenceTransformer library. This model efficiently captures semantic nuances in user queries, facilitating accurate vector representation and retrieval.
- **Cross-Encoder Model**: For reranking the retrieved documents, we employed the *cross-encoder/ms-marco-MiniLM-L-6-v2* model from the Hugging Face Transformers library. This model evaluates the relevance of document-query pairs, providing a refined ranking that prioritizes semantically coherent responses.
- **Quality Checker**: A custom-built quality checker module evaluates the *aggregated relevance scores* of the top-k retrieved documents, which were the top-5 documents for *RoostAI*, ensuring that only high-quality information is passed to the LLM. This step is crucial for maintaining the accuracy and reliability of the final response.

c) System Deployment: Our system is deployed on *Chameleon Cloud* [15], utilizing a Kernel-based Virtual Machine (KVM) instance to ensure scalability and accessibility. Apache 2 [10] serves as the HTTP server, providing a robust and reliable platform for handling user requests. This cloud-based deployment ensures that *RoostAI* remains available to users across campus and beyond, with minimal downtime and seamless performance.

These technologies collectively empower *RoostAI* to deliver precise, contextually relevant responses, enhancing the user experience for the University of South Carolina community.

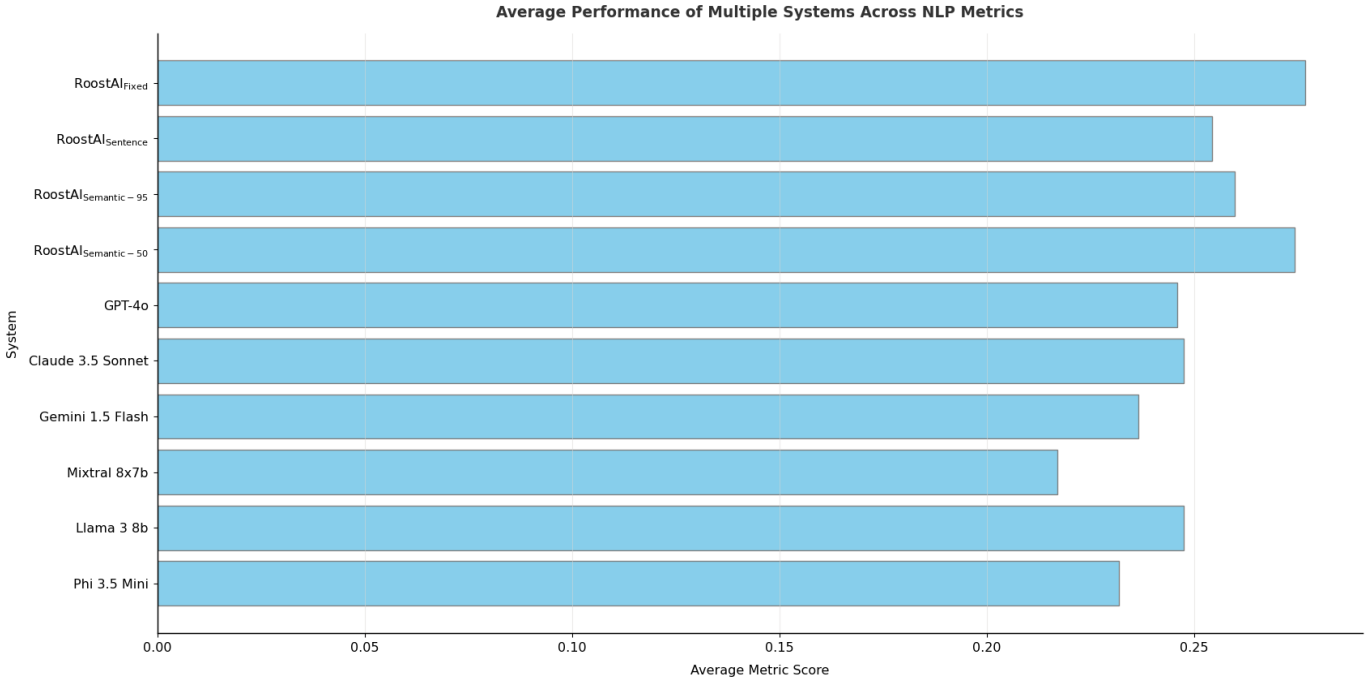

Fig. 3. Bar chart illustrating the average performance of *RoostAI* variants and baseline LLMs across multiple NLP metrics. *RoostAI* demonstrates higher average scores, highlighting its effectiveness in delivering precise and contextually relevant responses.

IV. Experimental Setup

To evaluate the effectiveness of our RAG-based university chatbot, we conducted a comprehensive comparison against state-of-the-art LLMs using a carefully curated FAQ dataset. Our experimental setup was designed to assess both the accuracy and reliability of responses in the specific context of university-related queries.

We constructed a test dataset consisting of 108 FAQ entries scraped from official University of South Carolina websites. Each entry contains (1) The user query and (2) the corresponding ground truth answer from official university documentation. We initialize 4 separate versions of *RoostAI* corresponding to the 4 chunking strategies, as mentioned in section III, corresponding to the names $\mathsf{RoostAI}_{\mathsf{Fixed}}$, $\mathsf{RoostAI}_{\mathsf{Sentence}}$, $\mathsf{RoostAI}_{\mathsf{Semantic-50}}$, and $\mathsf{RoostAI}_{\mathsf{Sematic-95}}$ respectively.

For evaluation against existing LLMs, we choose six leading LLMs spanning a range of parameter counts, as shown in Figure 2: GPT 4o by OpenAI [12], 3.5 Sonnet by Anthropic [4], Mixtral 8x7b Instruct by Mistral AI [13], Gemini 1.5 Flash by Google [22], Llama 3 8B Instruct by Meta [7], and Phi-3.5 Mini Instruct by Microsoft [2]. Preliminary evaluations seen in Figure 3 suggest that Mixtral 8x7B has the poorest performance for our use case. Therefore, we chose Mixtral 8x7B as our base model to motivate that the informedness granted by a RAG-based approach can elevate the performance of the weakest LLM. Our system accesses the Mixtral 8x7B model via the freely available Hugging Face Inference API to synthesize information from the retrieved documents to generate contextually appropriate and factually accurate answers.

V. Evaluation

For evaluating the performance of *RoostAI*, we compared the four instances against the six state-of-the-art LLMs, as detailed previously. The evaluation focuses on the *RoostAI*'s ability to accurately and reliably respond to university-specific queries. By utilizing a set of reference-based NLP metrics and RAG metrics, we aim to capture various aspects of the **(1)** generated response quality and the **(2)** quality of the retrieved contexts, thus providing a comprehensive assessment of the system's capabilities.

A. NLP Metrics Evaluation

We employed a diverse set of reference-based metrics to capture different aspects of the generated response quality. All outputs were normalized to a scale of 0 to 1, where 1 indicates a perfect match between the two texts. The chosen metrics were:

1) **ROUGE-L**: For measuring the longest common subsequence overlap
2) **BLEU**: For assessing n-gram precision
3) **BERTScore**: For semantic similarity evaluation
4) **Jaccard Similarity**: For token-level overlap
5) **Levenshtein Ratio**: For character-level difference measurement.
6) **Jensen-Shannon Divergence**: For distributional similarity assessment

We queried each of the LLMs and the *RoostAI* systems with a set of frequently asked questions (FAQs) and stored their responses. The university-provided responses were used as ground truth references. Each system's responses were

Fig. 4. Radar plots comparing multiple systems, including multiple *RoostAI* variants and leading LLM baselines, across various NLP evaluation metrics. *RoostAI* exhibits superior performance in semantic alignment and factual accuracy, as indicated by larger plot areas.

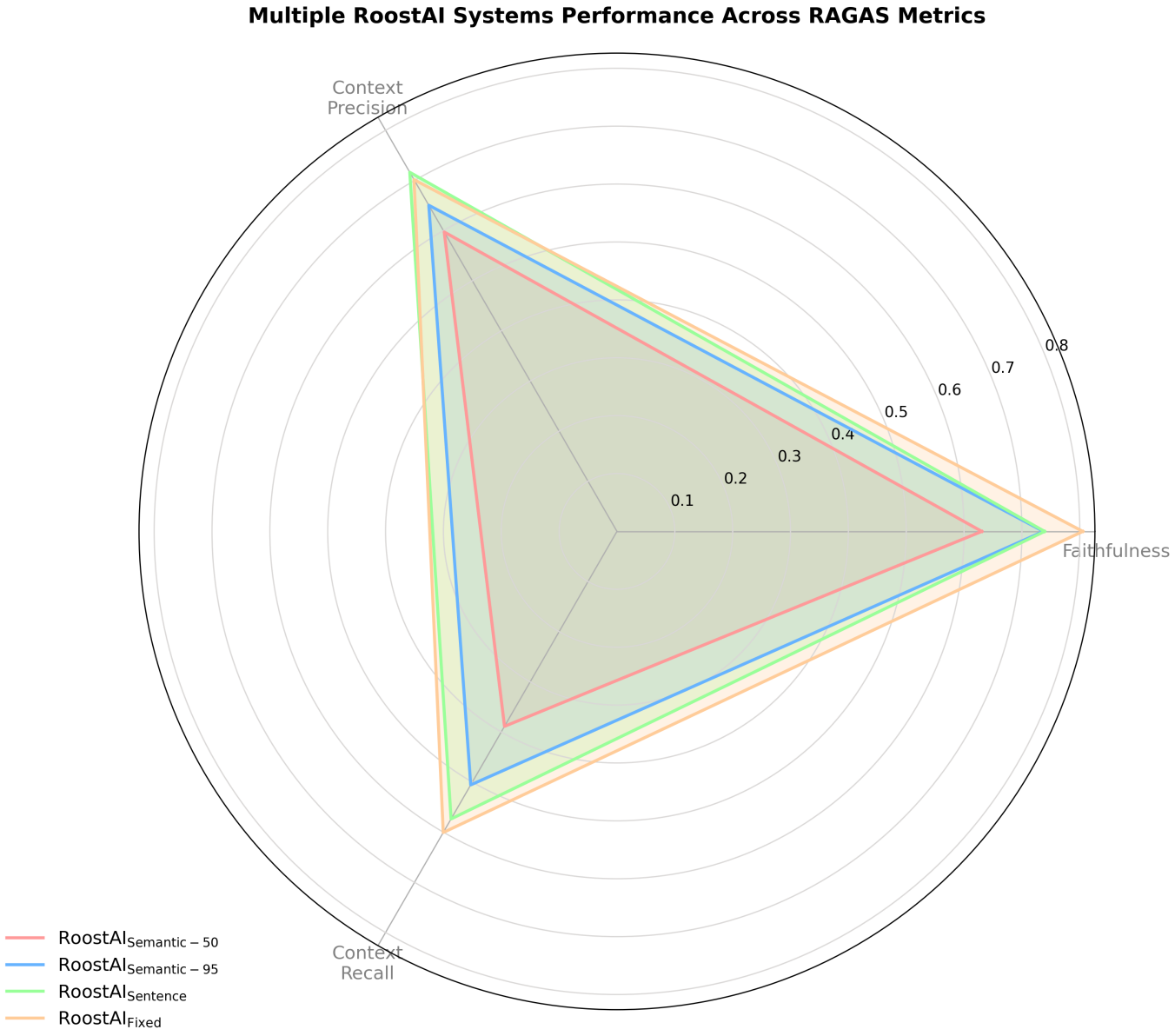

Fig. 5. Radar plot comparing 4 *RoostAI* variants on context recall, context precision, and faithfulness. $\text{RoostAI}_{\text{Fixed}}$ and $\text{RoostAI}_{\text{Sentence}}$ perform much better than their semantic chunking alternatives, indicated by the larger areas spanned.

evaluated using these metrics, and the results are displayed in Figures 3 and 4.

Figure 3 shows a bar chart that displays the average performance of each system across all metrics. The average metric score is calculated to provide a single comparative measure of each system's capability. On the other hand, figure 4 presents radar charts illustrating the performance of various systems across the six NLP metrics. Each chart corresponds to one *RoostAI* instance compared to the chosen LLMs. Each axis represents one of the metrics, and the area covered by each system's polygon reflects its overall performance. A larger area suggests better performance across the metrics.

B. RAG Metrics Evaluation

We employ three metrics from the RAGAS [8] framework, a library providing a suite of LLM-based reference-based and reference-free metrics for evaluating the quality of RAG systems. Specifically, we use the following RAG metrics to evaluate the quality of the retrieved documents and generated responses for our 4 RAG systems in order to determine which chunking strategy is most effective in our use case.

1) **Context Precision**: For measuring the proportion of relevant documents in the retrieved documents
2) **Context Recall**: For measuring how many of the relevant documents were successfully retrieved
3) **Faithfulness**: For measuring the factual consistency of

the generated answer against the given context

Figure 5 presents a radar chart highlighting the performance of each RAG system across the 3 RAG metrics, with each axis representing one of the metrics and the area of the system's polygon representing its overall performance, with a larger area suggesting a better performance.

VI. Discussion

This section elucidates the results of our evaluations, focusing on the performance of *RoostAI* compared to baseline models, and explores the implications for university information systems.

A. NLP Metrics Discussion

The results presented in Figures 3 and 4 highlight the superior performance of *RoostAI* variants compared to state-of-the-art baseline LLMs across multiple NLP evaluation metrics. Notably, the performance trends are consistent across all four *RoostAI* instances, with significant gains observed in metrics measuring semantic similarity and overall contextual accuracy.

a) Average Metric Performance (Figure 3): From the bar chart, we observe that all *RoostAI* variants achieve higher average metric scores than the baseline systems. Specifically, $\mathsf{RoostAI}_{\mathsf{Fixed}}$ demonstrates the highest overall performance, followed closely by $\mathsf{RoostAI}_{\mathsf{Semantic-50}}$ and $\mathsf{RoostAI}_{\mathsf{Semantic-95}}$. This indicates that the fixed chunking strategy yields more effective results when compared to dynamic or semantic-based approaches, likely due to better retrieval precision and fewer disruptions in contextual flow.

Among the baseline LLMs, Llama 3 8B and Claude 3.5 Sonnet exhibit the strongest performances, outperforming other base LLMs. However, even these top-tier systems fall short of the average scores achieved by *RoostAI*. This underscores the effectiveness of retrieval-augmented generation (RAG) combined with tailored chunking strategies in handling domain-specific tasks such as university FAQs.

b) Metric-Specific Insights (Figure 4): The radar plots offer a more granular view of the systems' performances across individual metrics:

- **BERTScore**: Across all *RoostAI* variants, the BERTScore values are notably higher compared to baseline systems. This highlights the superior semantic alignment of *RoostAI*'s responses with the ground truth answers, a critical factor for ensuring contextually relevant outputs.
- **ROUGE-L and BLEU**: While baseline models perform reasonably well on n-gram overlap metrics, *RoostAI* achieves slight improvements, indicating more accurate token-level matching. The improvements are particularly evident in $\mathsf{RoostAI}_{\mathsf{Fixed}}$ where the chunking strategies optimize for precise content retrieval.
- **Levenshtein Ratio and Jaccard Similarity**: These metrics further confirm the consistency of *RoostAI* in producing responses that closely mirror the ground truth in both structure and content. The fixed chunking strategy again yields superior performance, likely due to reduced token fragmentation.
- **Jensen-Shannon Divergence (JSD)**: While most systems achieve competitive scores on JSD, the higher divergence values observed for *RoostAI* suggest a higher degree of distributional similarity between the retrieved and ground truth responses. This reinforces the system's ability to maintain factual accuracy.

In the realm of NLP metrics, $\mathsf{RoostAI}_{\mathsf{Fixed}}$ consistently outperforms the dynamic ($\mathsf{RoostAI}_{\mathsf{Sentence}}$) and semantic-based ($\mathsf{RoostAI}_{\mathsf{Semantic-50}}$ and $\mathsf{RoostAI}_{\mathsf{Semantic-95}}$) strategies. While semantic chunking theoretically improves contextual relevance, the higher computational overhead and occasional alignment issues may explain its slightly lower scores compared to the fixed strategy. Nevertheless, $\mathsf{RoostAI}_{\mathsf{Semantic-50}}$ demonstrates competitive performance, suggesting that semantic chunking can strike a balance between precision and efficiency.

Despite their robust language modeling capabilities, baseline LLMs fall short in this domain-specific task. This can be attributed to their lack of access to specific university knowledge, which limits their ability to produce contextually accurate answers without fine-tuning or external retrieval mechanisms. The results emphasize the importance of integrating RAG pipelines for specialized information systems, as demonstrated by *RoostAI*'s superior performance.

B. RAG Metrics Discussion

Our evaluation of the 4 RAG systems also yields interesting insights regarding chunking strategies. As seen in Figure 5, fixed-token chunking and sentence-splitting chunking yield the best performance by far specifically in the precision and recall metrics, while semantic chunking with both selected thresholds performs as noticeably worse alternatives. This is a somewhat counterintuitive result regarding semantic chunking, which groups together semantically related chunks, presumably leading to more effective document retrieval. However, not only is this method more computationally expensive due to requiring embedding text and constantly calculating BERT scores, it also leads to less effective document retrieval overall. Even more interestingly,$\mathsf{RoostAI}_{\mathsf{Semantic-50}}$ performs worse than $\mathsf{RoostAI}_{\mathsf{Semantic-95}}$, even though the former method leads to more granular chunks due to a higher threshold for similarity. This may be due to user queries requiring additional context present in the coarse-grained chunks retrieved by $\mathsf{RoostAI}_{\mathsf{Semantic-95}}$. From these results, we conclude that fixed-token chunking and sentence-splitting chunking are more effective chunking strategies for our use case of RAG when considering key retrieval metrics. An interesting future extension of our work would be to determine the optimal token length chunk size for fixed-token chunking.

C. Future User Study

In addition to the aforementioned evaluation using RAG and NLP metrics, we had plan to conduct a user study of our RAG application to get feedback from our target audience. Although we were able to deploy a survey version of our application to present a user study in this report, we were unable to present it

as a substantial evaluation because we did not receive enough survey responses from users. We have plans to deploy this application as a helpful tool in conjunction with the university, and we will conduct a large-scale user study to evaluate the efficacy of our system.

VII. Conclusion

This study presented *RoostAI*, a Retrieval-Augmented Generation (RAG) based chatbot designed to address the fragmented access to information at the University of South Carolina. Our evaluations demonstrate that *RoostAI* outperforms baseline models across several NLP metrics, notably in terms of semantic and factual accuracy. By leveraging a USC-specific knowledge base, *RoostAI* delivers precise and contextually relevant responses, highlighting its potential as an effective tool in university information systems.

We also note our contributions with regards to our RAG metric evaluation, which gave interesting results with regards to picking the correct chunking strategy for a future RAG system implementation. While results suggest that fixed-token chunking or sentence-splitting chunking are the best choices for our use case, semantic-based approaches might offer promising results for systems requiring more nuanced contextual understanding. Future work can explore hybrid strategies that dynamically select chunking mechanisms based on query complexity.

Future research will focus on several key areas. Expanding the dataset to incorporate a broader range of university-related content and optimizing the retrieval pipeline will be essential for improving system coverage and accuracy. Additionally, exploring advanced chunking strategies and integrating feedback from user studies will inform iterative enhancements to the system's architecture.

The long-term vision for *RoostAI* involves its widespread deployment across university platforms, serving as a centralized, intelligent resource for students, staff, and visitors. By streamlining information retrieval processes, *RoostAI* can significantly enhance user satisfaction and efficiency, reducing the time spent navigating disparate resources. Its scalability and adaptability also position it as a model for similar applications at other educational institutions, contributing to the evolution of digital academic support tools.

In conclusion, *RoostAI* represents a promising advancement in university information systems, offering a robust, tailored solution to the challenges of accessing campus-specific information. Continuous improvements and strategic deployment will ensure its sustained impact, transforming the way academic communities engage with digital resources.

Acknowledgment

We acknowledge Professor Pooyan Jamshidi and our classmates in CSCE 585 for their guidance, feedback, and assistance throughout this project.

References

[1] Chroma. https://www.trychroma.com/. [Accessed 13-12-2024].

[2] Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A. A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219* (2024).

[3] Allahverdiyev, R. A. I. S. S. I., Taha, M., Akalin, A., and Zhu, K. Chunkrag: Novel llm-chunk filtering method for rag systems. *arXiv preprint arXiv:2410.19572* (2024).

[4] Anthropic. Introducing Claude 3.5 Sonnet. https://www.anthropic.com/news/claude-3-5-sonnet. [Accessed 13-12-2024].

[5] Burgan, C., Kowalski, J., and Liao, W. Developing a retrieval augmented generation (rag) chatbot app using adaptive large language models (llm) and langchain framework. *Proceedings of the West Virginia Academy of Science 96*, 1 (2024).

[6] Dan, Y., Lei, Z., Gu, Y., Li, Y., Yin, J., Lin, J., Ye, L., Tie, Z., Zhou, Y., Wang, Y., et al. Educhat: A large-scale language model-based chatbot system for intelligent education. *arXiv preprint arXiv:2308.02773* (2023).

[7] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[8] Es, S., James, J., Espinosa-Anke, L., and Schockaert, S. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217* (2023).

[9] Fernández-Pichel, M., Pichel, J. C., and Losada, D. E. Search engines, llms or both? evaluating information seeking strategies for answering health questions. *arXiv preprint arXiv:2407.12468* (2024).

[10] Group, D. Essentials¶.

[11] Huang, Y., and Huang, J. A survey on retrieval-augmented text generation for large language models, 2024.

[12] Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[13] Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[14] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[15] Keahey, K., Anderson, J., Zhen, Z., Riteau, P., Ruth, P., Stanzione, D., Cevik, M., Colleran, J., Gunawi, H. S., Hammock, C., Mambretti, J., Barnes, A., Halbach, F., Rocha, A., and Stubbs, J. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

[16] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems 33* (2020), 9459–9474.

[17] Maryamah, M., Irfani, M. M., Raharjo, E. B. T., Rahmi, N. A., Ghani, M., and Raharjana, I. K. Chatbots in academia: a retrieval-augmented generation approach for improved efficient information access. In *2024 16th International Conference on Knowledge and Smart Technology (KST)* (2024), IEEE, pp. 259–264.

[18] Microsoft. Fast and reliable end-to-end testing for modern web apps.

[19] Neupane, S., Hossain, E., Keith, J., Tripathi, H., Ghiasi, F., Golilarz, N. A., Amirlatifi, A., Mittal, S., and Rahimi, S. From questions to insightful answers: Building an informed chatbot for university resources. *arXiv preprint arXiv:2405.08120* (2024).

[20] Reimers, N., and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (11 2019), Association for Computational Linguistics.

[21] Saha, B., and Saha, U. Enhancing international graduate student experience through ai-driven support systems: A llm and rag-based approach. In *2024 International Conference on Data Science and Its Applications (ICoDSA)* (2024), IEEE, pp. 300–304.

[22] TEAM, G., GEORGIEV, P., LEI, V. I., BURNELL, R., BAI, L., GULATI, A., TANZER, G., VINCENT, D., PAN, Z., WANG, S., ET AL. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530* (2024).

[23] TEAM, L. I. Llamaindex: Build ai knowledge assistants over your enterprise data.

[24] THWAY, M., RECATALA-GOMEZ, J., LIM, F. S., HIPPALGAONKAR, K., AND NG, L. W. Harnessing genai for higher education: A study of a retrieval augmented generation chatbot's impact on human learning. *arXiv preprint arXiv:2406.07796* (2024).

[25] WINKLER, R., AND SÖLLNER, M. Unleashing the potential of chatbots in education: A state-of-the-art analysis. In *Academy of Management Proceedings* (2018), vol. 2018, Academy of Management Briarcliff Manor, NY 10510, p. 15903.

[26] WOLF, T. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[27] XIE, W., LIANG, X., LIU, Y., NI, K., CHENG, H., AND HU, Z. Weknow-rag: An adaptive approach for retrieval-augmented generation integrating web search and knowledge graphs. *arXiv preprint arXiv:2408.07611* (2024).

[28] YEPES, A. J., YOU, Y., MILCZEK, J., LAVERDE, S., AND LI, R. Financial report chunking for effective retrieval augmented generation. *arXiv preprint arXiv:2402.05131* (2024).

[29] ZHANG, T., KISHORE, V., WU, F., WEINBERGER, K. Q., AND ARTZI, Y. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).