RoostAI: Building a University-Centric Chatbot with Retrieval Augmented Generation (Results Discussion)

Vansh Nagpal
Computer Science and Engineering
University of South Carolina, Columbia
vnagpal@email.sc.edu

Nitin Gupta
Computer Science and Engineering
University of South Carolina, Columbia
niting@email.sc.edu

***Abstract*—In this manuscript, we present an evaluation of our ML system, *RoostAI*, a chatbot catered for the University of South Carolina, against state-of-the-art Large Language Models (LLMs). *RoostAI* is catered to people curious about USC and allows them to have an informed conversation with an intelligent agent. This application makes use of the LLM *Mixtral-8x7B-Instruct-v0.1* as its core response engine and the Retrieval Augmented Generation (RAG) methodology, which draws from a USC-specific knowledge base.**

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

I. Introduction and Related Work

In today's academic environment, students, staff, and visitors alike rely on digital tools to quickly access relevant information about the university they are part of. At the University of South Carolina (USC), no unified and interactive platform enables seamless access to vital information specific to the campus. Existing resources are often dispersed across various websites and portals, making it difficult for users to quickly find accurate and up-to-date answers.

Our project seeks to address that gap by developing a Retrieval-Augmented Generation (RAG) [1] based chatbot tailored for the USC community. The chatbot will serve as an intelligent virtual assistant, allowing users to ask natural language questions and receive precise, factual responses drawn from a USC-specific knowledge base. Leveraging advanced web-scraping, natural language processing (NLP), and machine learning techniques, the system will collect, index, and retrieve information from relevant USC web domains and other local resources. Our goal is to create a user-friendly, efficient, and highly accurate platform to enhance the overall campus experience.

This goal entailed a lot of smaller efforts, such as procuring a knowledge base of information pertinent to the University of South Carolina, setting up a RAG pipeline with retrieval, filtering, and re-ranking algorithms, testing the effectiveness of the system, and deploying a functional prototype of our system.

[1]We include a brief overview of RAG systems in Appendix sections A, B

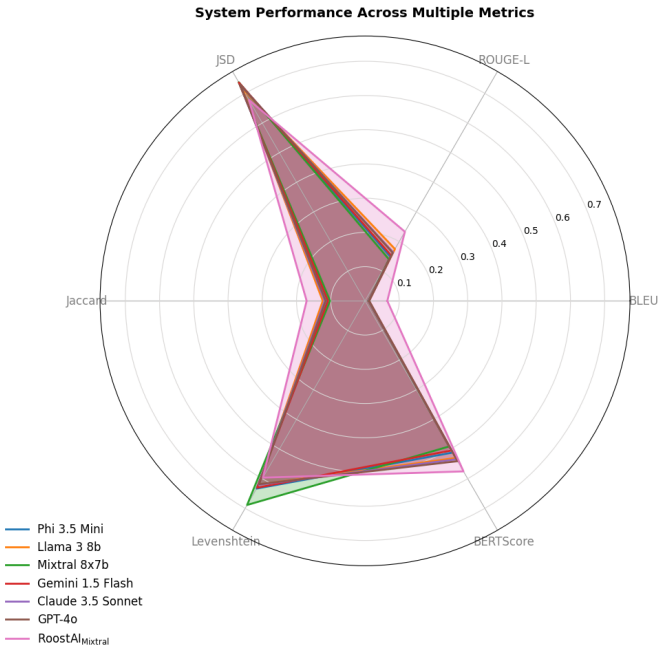


Fig. 1. Radar plot showing system performance across six evaluation metrics. The plot compares RoostAI against various LLM baselines, with scores normalized between 0 and 1. A larger area indicates better overall performance.

II. Experimentation

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.

a) Experimentatal Setup: 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.

For evaluation, we chose several leading LLMs, shown below in decreasing order of estimated parameter counts:

- GPT-4o: 1.8T
- Claude 3.5 Sonnet: 1.5T
- Gemini 1.5 Flash: 47B
- Mixtral 8x7b: 32B
- Llama 3 8b: 8B
- Phi 3.5 Mini: 3.8B

Our RoostAI implementation utilized the *Mixtral-8x7b-Instruct-v0.1* model as its base LLM, thus the name $\text{RoostAI}_{\text{Mixtral}}$,enhanced with the RAG pipeline described in previous sections. For specific details on the RAG system, please refer to Appendix section B.

b) Evaluation Setup: We employed a diverse set of metrics to capture different aspects of response quality:

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. Note that for this setup, higher Levenshtein score indicates a better match
6) **Jensen-Shannon Divergence**: For distributional similarity assessment

For this milestone, we focus on the quantitative comparison between our RAG-based approach and standalone LLMs to demonstrate the advantages of RAG for our specific use case. The results are presented in subsequent sections, analyzing both the positive outcomes and areas requiring improvement.

III. Positive Outcomes

Our evaluation of the RoostAI system against baseline LLM approaches demonstrates several notable strengths, as shown in Figures 1 and 2[2].

1) **ROUGE-L Performance**: The system achieved superior ROUGE-L scores compared to standalone LLMs, indicating better alignment with reference answers in terms of longest common subsequences. This suggests that our RAG implementation effectively preserves the semantic structure and factual accuracy of the source materials.
2) **BLEU Score Improvements**: The higher BLEU scores demonstrate that our system generates responses with greater n-gram overlap with ground truth answers. This indicates more precise lexical choices and better adherence to official university terminology.
3) **BERTScore Enhancement**: The improved BERTScore metrics suggest a stronger semantic similarity between generated responses and reference answers, indicating that our system better captures the underlying meaning of university-related content compared to baseline LLMs.
4) **Jaccard Similarity**: Higher Jaccard similarity scores indicate better overlap between the generated responses and ground truth answers at the token level, suggesting more accurate information retrieval and integration.
5) **Coverage Rate**: Of the 108 test queries, the system provided meaningful responses to 63 queries (58.3% coverage), demonstrating responsible behavior by declining to answer when confidence was low, rather than providing potentially incorrect information.

[2]For detailed results for each metric, please refer to Appendix section C.

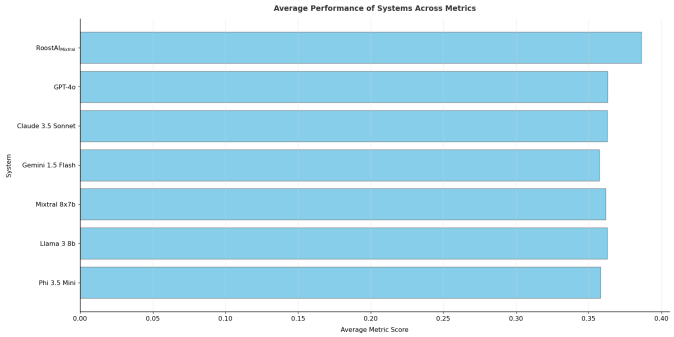


Fig. 2. Average performance comparison across different systems including RoostAI and baseline LLM models. Higher scores indicate better overall performance across all evaluation metrics.

IV. Shortcomings

Despite the encouraging results, several limitations warrant attention:

1) **Levenshtein Distance**: The system showed lower Levenshtein distances compared to baseline LLMs, indicating greater character-level differences from reference answers. This suggests that while semantic accuracy is maintained, the exact phrasing differs from official documentation. This could be attributed to:
   - The RAG system's tendency to reformulate retrieved information
   - Variation in response styles between the LLM and official documentation
2) **Jensen-Shannon Divergence**: Lower JS divergence scores indicate greater distributional differences between generated responses and reference answers. This may be due to:
   - The system incorporates information from multiple sources, leading to more diverse vocabulary usage
   - Different writing styles between the LLM's natural language generation and official university documentation
3) **Coverage Limitations**: The system's inability to answer 41.7% of queries indicates potential knowledge base gaps or retrieval mechanism limitations. This suggests a need for:
   - Expanding the knowledge base coverage
   - Refining the document chunking strategy
   - Improving the retrieval pipeline to better handle edge cases

These results suggest that while our RAG system excels at semantic accuracy and information relevance, there is room for

improvement in maintaining stylistic consistency with official documentation and expanding knowledge coverage.

V. Conclusion and Future Steps

Even though we have some promising results that motivate that a RAG system proves to be more useful than other LLMs, our work can be improved in several ways. As previously mentioned, it would be beneficial to refine our chunk retrieval strategy and expand our knowledge base, [3]. Another aspect of our vectorized database we were interested in exploring was our chunking strategy. For example, there are many types of chunking strategies described in RAG literature [9, 7], including *semantic chunking*, *fixed size chunking*, and *recursive chunking*, all of which have their strengths and weaknesses. Our database consists of *fixed size chunks*, each 1024 tokens in length. In the future, we want to instantiate separate copies of our knowledge base corresponding to different chunking strategies and evaluate the best approach for our use case. We would carry out this study by utilizing metrics specific to RAG systems [3], such as *faithfulness*. After finalizing our RAG system's architecture, we would like to deploy our system using a cloud service, such as Chameleon Cloud, and conduct a user survey. A user survey would allow us to gauge the usability and usefulness of our system to an end user, which can't be measured with the coarse-grained metrics we present in this manuscript.

This concludes the presentation of our results and discussion of future works.

Acknowledgment

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

References

[1] Zahra Abbasiantaeb et al. "Let the llms talk: Simulating human-to-human conversational qa via zero-shot llm-to-llm interactions". In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 8–17.

[2] Yang Deng et al. "Rethinking Conversational Agents in the Era of LLMs: Proactivity, Non-collaborativity, and Beyond". In: *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 2023, pp. 298–301.

[3] Shahul Es et al. "Ragas: Automated evaluation of retrieval augmented generation". In: *arXiv preprint arXiv:2309.15217* (2023).

[4] Yizheng Huang and Jimmy Huang. *A Survey on Retrieval-Augmented Text Generation for Large Language Models*. 2024. arXiv: 2404.10981 [cs.IR]. URL: https://arxiv.org/abs/2404.10981.

[5] Patrick Lewis et al. "Retrieval-augmented generation for knowledge-intensive nlp tasks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.

[6] Gabrijela Perković, Antun Drobnjak, and Ivica Botički. "Hallucinations in LLMs: Understanding and Addressing Challenges". In: *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*. 2024, pp. 2084–2088. DOI: 10.1109/MIPRO60963.2024.10569238.

[7] Renyi Qu, Ruixuan Tu, and Forrest Bao. *Is Semantic Chunking Worth the Computational Cost?* 2024. arXiv: 2410.13070 [cs.CL]. URL: https://arxiv.org/abs/2410.13070.

[8] Alireza Salemi and Hamed Zamani. "Evaluating retrieval quality in retrieval-augmented generation". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 2395–2400.

[9] Antonio Jimeno Yepes et al. *Financial Report Chunking for Effective Retrieval Augmented Generation*. 2024. arXiv: 2402.05131 [cs.CL]. URL: https://arxiv.org/abs/2402.05131.

[3] It is important to note here that we initially had a much larger knowledge base, which was deleted, and we were not able to procure another dataset in time

APPENDIX

A. RAG: A Brief Overview

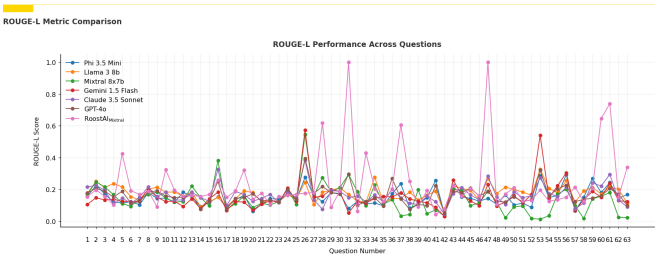
RAG systems utilize an LLM as the core response engine because the recent increase in the usage of LLMs has shown them to be extremely conversational, as compared to other rule-based chatbot systems [1, 2]. However, LLMs are known to hallucinate [6] and provide responses that may not be grounded in the truth, which not only reduces the efficacy of the system but also reduces the user's trust in the system. To combat this issue, many efforts [5, 8, 4] have established a technique, coined RAG, to grant generative models such as LLMs the benefit of information retrieval methodologies in an attempt to generate more informed and accurate responses. In short, a RAG pipeline entails the following steps: (1) The user makes a query q to the system, (2) A vectorized embedding for q, v_q is generated, (3) A retrieval algorithm utilizes v_q to retrieve a list of potential useful contexts, $C(q) = \{c_1(q), c_2(q), ..., c_n(q)\}$ that are similar and relevant to the original query q, (4) A re-ranking and filtering algorithm produces a short list of contexts $C'(q) \subset C(q)$, (5) The contexts in $C'(q)$ are appended to the original query q and is inputted to the chosen LLM as a prompt, (6) the response $r(q)$ is presented to the user.

B. Implementation Details

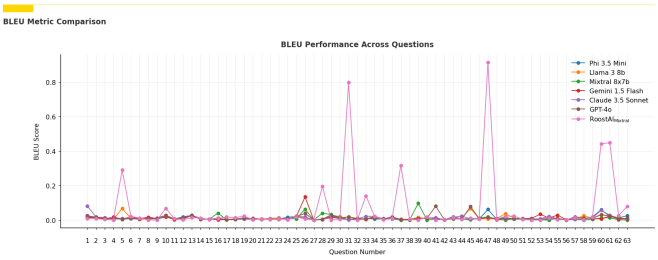
Our RAG pipeline was configured with:

1) Vector store: Chroma DB with HNSW indexing
2) Embedding model: all-MiniLM-L6-v2
3) Cross-encoder: ms-marco-MiniLM-L-6-v2
4) Base LLM: Mixtral 8x7b
5) Batch size: 100 documents
6) Top-k retrieval: 5 documents

C. Detailed Performance Metrics

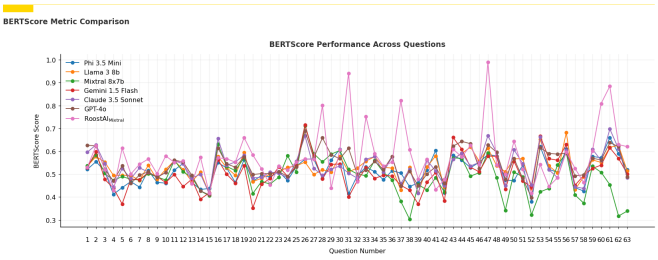


(a) ROUGE-L Scores

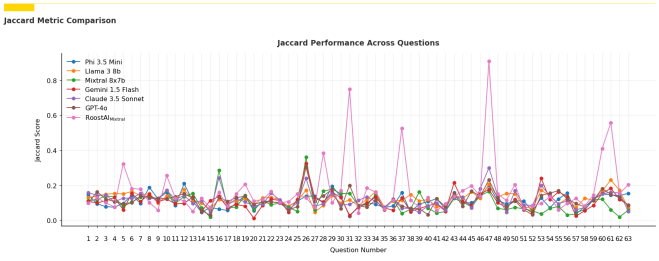


(b) BLEU Scores

Fig. 3. Question-by-question performance comparison for ROUGE-L and BLEU metrics

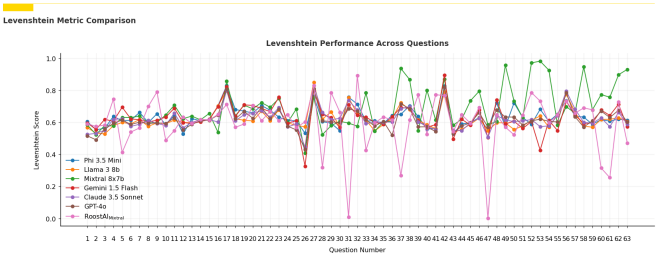


(a) BERTScore

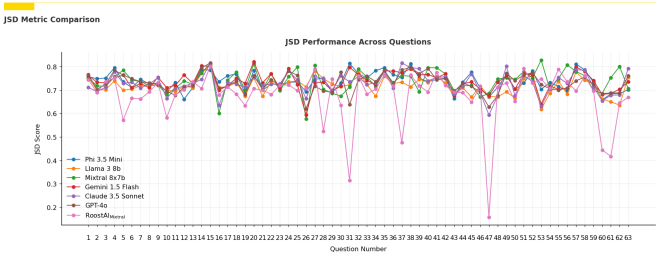


(b) Jaccard Similarity

Fig. 4. Question-by-question performance comparison for semantic similarity metrics



(a) Levenshtein Distance



(b) Jensen-Shannon Divergence

Fig. 5. Question-by-question performance comparison for distance metrics