Development of an RAG-based LLM Chatbot for Enhancing Technical Support Service

Ho-Chit Lee, Kevin Hung, *Senior Member*, *IEEE*, Gary Man-Tat Man, Raymond Ho, *Member*, *IEEE*, and Monica Leung

***Abstract*— The global shortage of manpower for technical support is a critical issue in the digital transformation era. Recently, Large Language Models (LLMs) have made significant strides in natural language processing, leading to the development of AI chatbots to address this problem. However, LLMs have notable limitations in handling domain-specific information, often generating incorrect responses when queries go beyond the coverage of the training data or require the most up-to-date information. A promising solution is the Retrieval-Augmented Generation (RAG) approach, which incorporates domain-specific data retrieval into the generative process. Our team has developed a domain-specific and RAG-based LLM chatbot to enhance the software house technical support of an IT consultant in Canada. The chatbot was implemented and evaluated in real-world production environments. Preliminary results show that the system has achieved high scores of 38%, 188%, and 40% in the ROUGE-1, ROUGE-2, and ROUGE–L measures, respectively, compared to using only a general LLM model. End-user feedback also reflected that the enhanced system produced more accurate and efficient replies, thereby enhancing overall customer satisfaction.**

***Keywords* - Large language model, retrieval-augmented generation, chatbot, technical support**

I. INTRODUCTION

Digital transformation is a key enabler of smart city development. The push towards digital solutions became particularly evident during and following the COVID-19 pandemic. However, the technical talent shortage has become the most significant adoption barrier of digital transformation. More than 64% of IT executives agree that this problem will be one of the main threats to their business [1]. Recently, automation through artificial intelligence (AI) is becoming increasingly important in technical support. According to a study, 91% of businesses plan to use AI for customer service within the next few years in order to minimize the impact of talent shortage [2]. While traditional AI approaches can provide customers with quick service, they rely on rule-based systems or traditional machine learning algorithms to automate tasks, providing predefined responses to customer inquiries. Generative AI has the potential to reform customer service. It leverages large language models (LLMs) and deep learning techniques to understand complex inquiries and to offer more natural conversational responses [3].

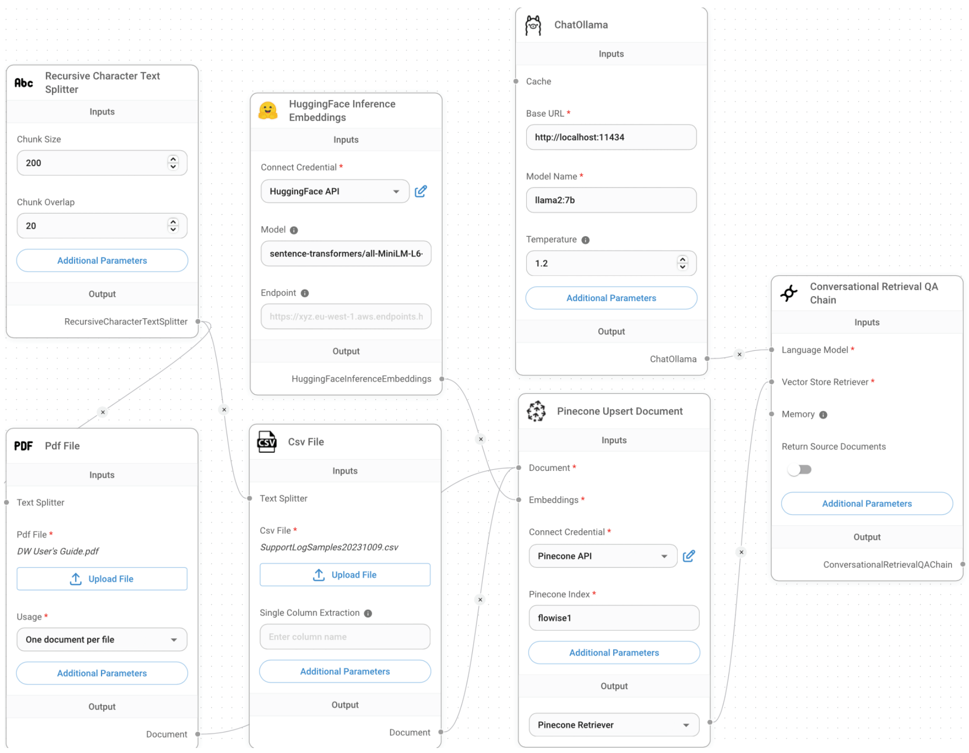
Despite these advancements, LLMs exhibit notable limitations, particularly in handling domain-specific or highly specialized inquiries. A common issue is the generation of incorrect responses, or “hallucinations”, especially when queries extend beyond the coverage of the model’s training data or require the most up-to-date information. These shortcomings underscore the impracticality of deploying LLMs as black-box solutions in real-world production environments without additional safeguards [4].

Retrieval-augmented generation (RAG) is an LLM learning technique that merges the retrieval mechanisms and generative capabilities to enhance the performance of LLMs [5]. Generally, retrieval models are good at searching vast external knowledge bases and finding relevant information for a given prompt. In contrast, generative models excel at utilizing this information to generate new text. This hybrid approach often leads to the generation of more accurate, informative, and in-context results as compared to using retrieval and generative models separately. In general, RAG is suitable for AI technical support applications [6-7]. Chatbots with RAG capabilities can easily pull relevant information from an organization’s instruction manuals and technical documents in order to respond to customers’ queries with context-aware answers, while the risk of hallucinations is reduced. Compared with other methods such as fine-tuning, RAG performs exceptionally well in dynamic settings. This is because it regularly requests the most recent data from external knowledge bases without the need for frequent highly computational loading retraining. This ensures that the information generated by RAG-powered models is always up-to-date [8].

In collaboration with OptiMicro Technologies Inc. (OptiMicro), a Canadian software company, a domain-specific LLM chatbot using RAG has been developed for enhancing technical support service of the company, with the aim to improve overall customer satisfaction. The remainder of this paper is structured as follows. Section 2 presents the methodologies, which includes a brief overview of the RAG method, a detailed step-by-step implementation guide by Flowise AI [9], and evaluation strategies based on real production queries from OptiMicro. Section 3 highlights the results obtained. Section 4 provides a detailed analysis of these results and discusses potential future developments. Section 5 concludes the paper.

H..C. Lee, Kevin Hung, Gary Man-Tat Man, and Monica Leung are with the School of Science and Technology, Hong Kong Metropolitan University, Hong Kong (e-mail: s1266144@live.hkmu.edu.hk, khung@hkmu.edu.hk, mtman@hkmu.edu.hk, mleung@hkmu.edu.hk). Raymond Ho is with OptiMicro Technologies Inc., Canada (e-mail: raymondho@ieee.org)

TENCON 2024 - 2024 IEEE Region 10 Conference (TENCON) | 979-8-3503-5082-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/TENCON61640.2024.10902801



Recursive Character Text Splitter
Inputs
Chunk Size: 200
Chunk Overlap: 20
Additional Parameters
Output
RecursiveCharacterTextSplitter

HuggingFace Inference Embeddings
Inputs
Connect Credential *: HuggingFace API
Model: sentence-transformers/all-MiniLM-L6-
Endpoint: https://xyz.eu-west-1.aws.endpoints.h
Output
HuggingFaceInferenceEmbeddings

ChatOllama
Inputs
Cache
Base URL *: http://localhost:11434
Model Name *: llama2:7b
Temperature: 1.2
Additional Parameters
Output
ChatOllama

Conversational Retrieval QA Chain
Inputs
Language Model *
Vector Store Retriever *
Memory
Return Source Documents
Additional Parameters
Output
ConversationalRetrievalQAChain

PDF Pdf File
Inputs
Text Splitter
Pdf File *: *DW User's Guide.pdf*
Upload File
Usage *: One document per file
Additional Parameters
Output
Document

Csv File
Inputs
Text Splitter
Csv File *: *SupportLogSamples20231009.csv*
Upload File
Single Column Extraction: Enter column name
Additional Parameters
Output
Document

Pinecone Upsert Document
Inputs
Document *
Embeddings *
Connect Credential *: Pinecone API
Pinecone Index *: flowise1
Additional Parameters
Output
Pinecone Retriever

Fig. 1. Implementation of the RAG-based LLM system using Flowise AI. The system incorporates customer contact log, user manual, text splitter, embedding model, vector database, and an LLM.

II. METHODOLOGIES

A. *Overview of RAG architecture*

The RAG architecture typically comprises three primary stages: indexing, retrieval, and generation. For the indexing process, the original data is purified, extracted, converted into standardized plain text, and divided into smaller segments. These segments are then converted into vector representations using an embedding model, which aids similarity comparisons during the retrieval stage. During the retrieval process, the system uses the same encoding model used in the indexing stage to convert the input into a vector representation. It then calculates the similarity scores between the query vector and the vector segments within the indexed corpus. The system prioritizes and retrieves the top K segments that show the highest similarity to the query. These segments are then used as the expanded contextual basis for addressing the user's request. In the final generation stage, the posed query and selected documents are combined into a coherent prompt to which an LLM is assigned to generate a response. The model's response strategy may vary based on task-specific criteria, allowing it to either utilize its inherent parametric knowledge or limit its responses to the information within the provided documents.

B. *System Implementation using Flowise AI*

In our proposed framework, Flowise AI, a low-code tool for LLM development, is employed to build the LLM chatbot. Figure 1 illustrates the overall implementation of our RAG-based LLM chatbot. The system incorporates the stages for indexing, retrieval, and generation.

Indexing stage: For data preparation, customer contact logs recorded from OptiMicro's previous technical support sessions, as well as user guide of DentalWare, a software developed by OptiMicro for dental service application, are collected and converted to "chunks" of plain text via Flowise AI's built-in text splitting function. The chunk size is set to 200, and the chunk overlap is set to 20 for the text splitting. In the data embedding process, the Sentences-Transformer model, all-MiniLM-L6-v2, is used through the HuggingFace platform via the Application Programming Interface (API) method. Vector databases, a relatively new type of database that can store and query unstructured data such as images, text and video, are gaining popularity among developers who want to build generative AI applications such as chatbots, recommendation systems, and content creation. On the other hand, Pinecone has a faster searching speed with real-time queries, while Chroma is more flexible but has a slower searching speed and lower throughput. For faster performance, Pinecone vector database is employed in our system. Both the data embedding and data splitting processes are connected to the Pinecone vector database block. The system would be able to convert the inputted customer contact logs to vector data and store the vector data in the Pinecone vector database.

Retrieval stage: During the retrieval process, the conversational retrieval QA chain API is used. The API uses the same embedding model employed in the indexing stage to convert the input query into a vector representation. It then calculates the similarity scores between the query vector and the vector segments within the indexed chunks corpus. The system prioritizes and retrieves the top K segments that show the highest similarity to the query.

Generation stage: According to a study which compared the truthfulness of common open-source LLM models of the same size, the performance of LLaMA 2 was better than that of other models [10]. Take the size of 7B as an example, the truthfulness was 29.13% in MPT, 25.95% in Falcon, 27.42% in LLaMA 1, and 33.29% in LLaMA 2. The percentage of LLaMA 2 was the highest in truthfulness, which means that the output of LLaMA 2 was the most accurate and informative. Therefore, in our work, LlaMA 2 7B is used. It was run locally on our computer's Ollama platform. An LLM's temperature represents the creativity of the LLM, ranging from 0 to 2. If the temperature is too low, the responses would lack details and may not be able to determine similar information from the inputted data. If the temperature is too high, the responses would not be related to the inputted data. After several runs and trials, the value of temperature was set to 1.2 in our simulation.

C. System Evaluation

To evaluate our RAG-based LLM chatbot, real production data from the customer contact log of OptiMicro was employed. There were 75 queries in total, and all these questions were related to technical support about billing, electronic data interchange, and other issues of the DentalWare software developed by OptiMicro. In the test, these questions were inputted to both the RAG-based LLM chatbot and a general LLM chatbot. The responses from these chatbots were analyzed and compared with the original replies of human technical support. The performance metrics consisted of ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores [11] and benchmarking by human expert. ROUGE is a set of metrics that evaluate the quality of summaries by comparing them to a set of reference summaries. ROUGE scores fall on a scale from 0 to 1, where a score closer to 1 suggests that the LLM response is strikingly similar to the human-generated one. In our simulation, ROUGE-1, ROUGE-2, and ROUGE-L were applied on 20 selected questions. On the other hand, the responses from these chatbots of all 75 questions were also benchmarked with those by human experts of OptiMicro. The marking scheme computes the overall scores and scores of each question type. It takes 1 score for good answer, 0.5 score for fair answer, and 0 score for poor answer.

III. RESULTS

After executing all 75 queries on the RAG-based LLM chatbot and standard LLM chatbot, and comparing their responses with the original technical support's replies, it is evident that the performance of RAG-based LLM chatbot outperformed that of the standard LLM chatbot. In general, the responses of the RAG-based LLM chatbot were domain-specific and were closely related to the DentalWare software documents. On the other hand, the answers of the standard LLM chatbot lacked specific contents about the software and could not fulfill the purpose of AI technical support.

A. ROUGE Measures

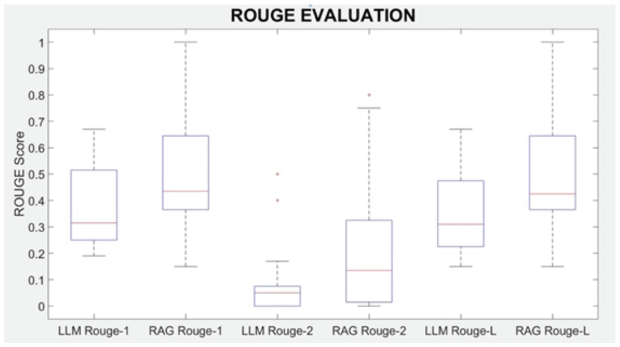


Fig. 2. Boxplot diagram showing results of the ROUGE measures.

Results of ROUGE measures are summarized in Figure 2. The upper quartile, median, and lower quartile of the measures of the RAG-based LLM is higher than that of the general LLM. This shows that the RAG-based LLM performed better text summarization than the general LLM. The RAG-based LLM achieved 0.51 scores in ROUGH-1, 0.23 scores in ROUGH-2, and 0.49 scores in ROUGH-L, which are 38%, 188%, and 40% higher than that of the general LLM, respectively.

B. Human Expert Benchmarking

As shown in Table I, our RAG-based LLM outperformed the general LLM in benchmarking with the actual responses. The RAG approach had 20 good replies and 18 fair replies, achieving a score of 29 in responding to the 75 questions. The general LLM had 3 good replies and 2 fair replies, and only obtained a score of 4.

TABLE I. COMPARISON OF PERFORMANCE OF RAG-BASED LLM AND GENERAL LLM BENCHMARKED WITH THE ACTUAL RESPONSES OF THE COMPANY.

LLM Chatbot	Good (1.0)	Fair (0.5)	Poor (0.0)	Total Score
General	3	2	70	4 / 75
RAG	20	18	37	29 / 75

IV. DISCUSSION

An RAG-based LLM chatbot is built via the Flowise AI platform. From the evaluation results, it can be verified that the performance of our chatbot is much better than that of the standard LLM chatbot. The key advantages of RAG are summarized as follows:

- Cost-efficient: RAG implementation is more cost-effective than training foundational models from scratch, offering a significant advantage in terms of resource allocation.
- Easy updates: The knowledge base can be updated without the need to retrain the entire model, allowing quick adaptation to changing information landscapes.
- Room for upgrade: The system can catch up with the technology improvements by updating the LLM model.
- Reduced risk of hallucinations: Generative AI models often create responses disconnected from reality, known as hallucinations. By integrating RAG, these models gain access to external knowledge bases and documents enhancing their ability to generate responses rooted in factual information.
- Trustworthiness: By citing sources in its responses, RAG enhances the credibility and reliability of the chatbot answers, fostering user trust and confidence.

While the proposed algorithm yielded promising results, it is important to note that these findings are based on a relatively small dataset. This limitation may affect the generalizability of the conclusions. Future studies should aim to replicate these results using larger and more diverse datasets to validate the performance of the application of RAG-based LLM chatbot. Further efforts should be focused on data preprocessing, quantitative evaluation scheme, and multi-dialogue retrieval mechanism. In the simulations, all the training data are purified before embedding into the retrieval vector database. However, in a real-world scenario it is impossible for all the data to be perfect. Therefore, data preprocessing is very important in a RAG-based system. In data preprocessing, one critical process is to remove the clients' information from the data for privacy concerns. Short forms should also be converted, typo mistakes should be corrected, and some incomplete data should be avoided.

Furthermore, a quantitative evaluation scheme should be employed for the optimization of the performance of the RAG-based LLM chatbot. There are two important factors to consider while benchmarking a chatbot: accuracy and usefulness. Chatbot accuracy is a measure of how factually correct it is, while usefulness measures to what extent the chatbot meets the user's needs [12]. Based on this quantitative evaluation method, it is possible to optimize the system parameters such as the setting of the chunk size, chunk overlap, and the number of chunks during retrieval, in order to achieve better performance in AI technical support. In addition, only single-turn conversation is assumed, that is there is only one exchange between the user and the chatbot. However, not all tasks can be communicated in single-turn conversations. Complex tasks may require multi-turn conversation. Therefore, RAG-based LLM chatbot should handle multi-turn conversations. To handle user-triggered multi-turn conversations, context retention support should be added to the chatbot. Context retention is the ability of the chatbot to remember the conversation history and generate more relevant and informative responses. The chatbot can then keep track of what has been said before in order to provide helpful and consistent responses. Supporting multi-turn conversations is crucial for delivering natural user interactions across a wide range of services.

V. CONCLUSION

This paper has presented the development work of an RAG-based LLM chatbot for enhancing the technical support service of a software company. It was developed using the Flowise AI platform, and was tested using real production data from the company. Preliminary results revealed that the proposed solution has 38%, 188%, and 40% higher scores in ROUGE-1, ROUGE-2, and ROUGE–L measures as compared with that of the raw LLM model. Benchmarked with actual technical support responses from the company, our proposed chatbot provided more accurate responses as compared with the standard LLM chatbot.

ACKNOWLEDGMENT

The authors would like to thank OptiMicro Technologies Inc., and their R&D team for providing the technical advices and collaboration opportunity in this project.

REFERENCES

[1] M. Rimol, "Gartner survey reveals talent shortages as biggest barrier to emerging technologies adoption," Gartner, Inc., https://www.gartner.com/en/newsroom/press-releases/2021-09-13 (accessed February 9, 2024).
[2] B. Samonte, "The future of Tech support and help desk: What C-suites need to know about the latest global industry trends," Superstaff.com, https://www.superstaff.com/blog/the-future-of-tech-support-and-help-desk/ (accessed February 9, 2024).
[3] R. Giovis, and E. Rozsa, "Transforming customer service: How generative AI is changing the game," IBM Artificial Intelligence, https://www.ibm.com/blog/ (accessed February 14, 2024)
[4] Y. Gao et al, "Retrieval-augmented generation for large language models: A survey," *arXiv Computation and Language (cs.CL)*. [Online]. Available: https://doi.org/10.48550/arXiv.2312.10997.
[5] P. Lewis et al, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proceedings of the 34th Int. Conf. on Neural Information Processing Systems (NIPS'20)*, Vancouver BC Canada, December 6-12, 2020, pp. 9459-9474.
[6] C. Jeong, "A study on the implementation of generative AI services using an enterprise data-based LLM application architecture," *arXiv Artificial Intelligence (cs.AI)*, September 18, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2309.01105
[7] M. Kulkarni, P. Tangarajan, K. Kim, and A. Trivedi, "Reinforcement learning for optimizing RAG for domain chatbots," *arXiv Computation and Language (cs.CL)*, January 10, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.06800
[8] A. Haponik, "RAG vs. fine-tuning: A comparative analysis of LLM learning techniques," Addepto.com, https://addepto.com/blog/ (accessed February 14, 2024)
[9] Flowise. [Online]. Available: https://github.com/FlowiseAI/Flowise
[10] H. Touvron et al., "Llama 2: Open Foundation and Fine-Tuned Chat Models," *arXiv Computation and Language (cs.CL)*, Jul. 19, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.09288
[11] C.Y. Lin, "ROUGE: A package for automatic evaluation summaries," in Summarization Branches Out, Barcelona, Spain, Association for Computational Linguistics, 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013.pdf.
[12] D. Banerjee, P Singh, A. Avadhanam, and S. Srivastava, "Benchmarking LLM powered chatbots: Methods and metrics," *arXiv Computation and Language (cs:CL)*, August 8, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.04624