ISSN: 2473-4901
v10 n6170

RAG Chatbot for Healthcare related prompts using Amazon Bedrock

Oladapo Richard-Ojo
or01131@georgiasouthern.edu
Department of Information Technology
Georgia Southern University
Statesboro, GA 30460, USA

Hayden Wimmer
hwimmer@georgiasouthern.edu
Department of Information Technology,
Institute for Health Logistics and Analytics,
Georgia Southern University
Statesboro, GA 30460, USA

Carl M Rebman, Jr.
carlr@sandiego.edu
Knauss School of Business
Department of Supply Chain, Operations, and Information Systems
University of San Diego
San Diego, CA 92110, USA

Abstract

Applications of natural language processing (NLP) for use in large language models (LLMs) continue to evolve with technological advancements in the domain Generative AI (GenAI). The massive explosion of data, availability of scalable computing capacity and machine learning innovation, LLMs, have all led towards Generative AI (GenAI) becoming increasingly popular. A major challenge involved with base model LLMs is their tendency to hallucinate. Hallucination in LLMs refers to the output of inconsistent incoherent and sometimes incorrect information or response. This occurs as most LLMs are trained on a large amount of generic data and must be augmented using domain specific and external data for use in GenAI tasks such as chatbots, Q&A, summarization and for text generation. To address the challenge of hallucination, this study will make use of domain specific healthcare data, in the form of PDF files, alongside an FM to create a Retrieval Augmented Generation (RAG) chatbot. This study makes use of the base foundation model, Llama 2 from Amazon bedrock. Our domain specific healthcare data was sourced from relevant and reliable sources. The RAG chatbot was developed using Python and colab notebook and responses were evaluated using Rouge and Meteor, evaluation metrics for automatically generated text. The evaluation was based on three scenarios: responses less than 250 characters, more than 250 characters and combined responses from multiple LLMs. Our findings provide strong evidence that augmenting Foundation models (FMs) with domain specific data can improve the quality of the models' responses in providing reliable medical knowledge to patients.

Keywords—LLMs, Amazon Bedrock, GenAI, foundation models, llama2, hallucination.

ISSN: 2473-4901
v10 n6170

1. INTRODUCTION

Computing technologies have provided many benefits to humans, yet they were originally built for numerical processing or structured functions. One of the challenges and natural scientific inquiry is to develop a bridge to where computers can understand human language. We refer to this today as artificial intelligence. Natural language processing (NLP) models are a part of artificial intelligence and machine learning research that started in the 1950s. Traditional ML models require labeled data that is then trained and fed into the model to be used for several NLP tasks. Many people recognize NLP through semantic analysis. NLP has limitations in its ability to process language diversity, contextual understanding, and requires extensive computing resources which limits their ability for real time processing of simultaneous transactions. Zhu, Wang, Chen, and Liu (2020)

Large language models (LLMs) were created as response to the limitations of NLP. They undergo training to learn relationships from a vast amount of data such as text documents, in a self-supervised or semi supervised process. Unlike traditional machine learning models, LLMs use multi-dimensional vectors called word embeddings to ensure that words with similar context are correctly outputted. Some common examples of LLMs include Amazon Titan, Grok, OpenAI's GPT, Cohere and Anthropic. Ovadia, Brief, Mishaeli, and Elisha (2023)

Large language models have limitations because of their training from pre-existing data. This pre-existing data can have biases to which some question the fairness and accuracy of the outputs. Another major challenge with LLMs is that they have 'hallucinations.' Hallucination in large language models is typically associated with the generation of factually inaccurate and contextually inappropriate responses. Factors such as lack of domain specific data, presence of biased training data and a lack of real-world knowledge contribute to hallucination of LLMs. The implications of hallucination in the healthcare domain can be devastating as patients require accurate and informed responses from large language models for medical and/or drug related inquiries. Jin, Yang, Chen, and Lu (2023)

Generative artificial intelligence or Gen AI as it is more commonly known is the next step of NLP and LLM that aims to mimic human conversations and generate ideas and content. Figure 1 illustrates the data flow advancement of Gen Ai over machine and deep learning. Gen AI uses an array of data inputs such as text, images, sounds and animations to produce new content and output based on the input data. Ascorbe, Campos, Domínguez, Heras, and Terroba-Reinares (2023)

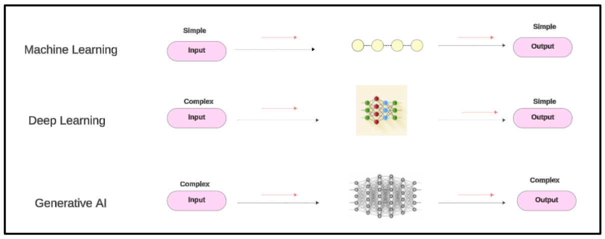


Figure 1: ML, DL, and Gen AI data flow comparison

Gen AI is the bedrock of Foundation models (FMs) mainly because FMs are trained on a large amount of unlabeled data that are used to perform several tasks. While machine learning and deep learning techniques provide simple outputs, Gen AI has the capability to accept complex data as input and output complex data. Gen AI models are enhanced by retrieval augmented generation (RAG) which can process outside information in addition to the models' original training data. Many people know RAG as a chatbot. Dou et al. (2023)

Amazon bedrock is a managed service by AWS that allows you to build and scale Gen AI applications using foundation models (FMs). It allows the extension of FMs alongside personal or proprietary data to build Retrieval Augmented Generation (RAG) applications.

This paper applies this technology to design a RAG application that caters to healthcare related queries further strengthening a domain that has limited specialized healthcare related virtual assistants or chatbots. More specifically this paper creates a RAG chatbot by augmenting a foundation model, Llama 2, with medical and healthcare related data using pdf files. The pdf files are ingested into the Llama 2 model, vector embeddings are stored in a vector store.

The development environment used for this project was Google Collab notebooks and the application was built using Python programming language. Our foundation model is accessed from Amazon bedrock using a Python library called Langchain. For the vector embedding, we used Amazon Titan Embedding which was accessed from the Amazon bedrock service while we used the FAISS (Facebook AI Similarity Search) library as our vector embeddings store.

Amazon bedrock requires authentication in order to use services. To satisfy this requirement, we used AWS Command Line Interface (CLI) to validate the credentials required by the cloud service.

Lastly, we used Streamlit for our user-facing chatbot that provided answers to healthcare related prompts. Our results show that the Augmented Llama 2 model outperforms other models which provide support to our argument that augmenting foundation models with relevant and domain specific healthcare data improves the quality of generated responses and can improve access to healthcare knowledge.

The goal of this study and our motivation is to add to the scarce research studies available in this domain and by so doing improve the research field. Ultimately, we have come up with two research questions which we will seek to answer at the end of this study.

Research Question 1- Can we reduce hallucination in LLMs by using Retrieval augmentation?

Research Question 2- To what extent does combining LLM responses improve the quality of our responses vs the base models.

2. LITERATURE REVIEW

(Biswas, Islam, Shah, Zaghouani, & Belhaouari, 2023) conducted a preliminary study to determine the potential of using ChatGPT as a medical assistant chatbot that can achieve NLP tasks relating to symptom checking, health education, diagnosis etc. One unique element of their study was that they conducted testing using the Arabic language. They fine-tuned the LLM (GPT 3.5) using open-source question and answer datasets and the evaluation of the fine-tuned model was done using human and automated metrics.

Chen et al. (2023) was interested in investigating the effects of LLM on long form automated speech recognition (ASR). They utilized YouTube videos as their long form ASR. They then studied the impacts of using an LLM to reduce the word error rate and salient term error rate on those YouTube videos. Their dataset consisted of several thousand hours of long form utterances derived from YouTube videos along with short form utterances derived from Google. The authors then used two different LLMs, the T5 and PaLm and tested it on different sizes ranging from small (60 million parameters) to XXL (118 billion parameters).

Chen et al. (2023) analyze the impact of different factors, such as the language model size, the beam size, and the utterance length, on the rescoring performance. They conclude that their method is effective and scalable for long-form speech recognition tasks and that combining LLMs improves the results in contrast to using baseline LLM singularly.

To get improved results for NLP and text classification tasks, Wei et al. (2023) compared a Distil BERT LLM base model with a fine-tuned LLM model using domain specific legal dataset. Additionally, they set out to further evaluate the performance of both LLM models by scoring an entire document on one hand and scoring sentences in a document on the other. They carried out data preprocessing on three types of datasets: confidential, non-public, and real-world legal matters. The data was cleaned up and filtered to be used for fine tuning of the distil BERT. Hugging face API was used for the finetuning of the LLM and text classification. The testing data sets of each project were then scored using the standard pretrained Distil BERT LLM and the refined Distil BERT LLM. In the first text classification job, the models are applied to entire texts by predicting using only the first 512 text tokens. The Distil BERT token limit by default is set at this value. In the second text classification job, the documents are divided into text snippets, and the models are applied to these snippets. The score for the entire document is then determined by taking the highest scoring portion from each document. The results of the study demonstrated that optimizing the LLM can enhance the effectiveness of ensuing text categorization, particularly in legal document review. The findings also demonstrate that text classification using a refined LLM performed at the snippet level can outperform document-level classification depending on the project. Whatever the document segmentation option, the optimized LLM consistently outperforms the baseline pretrained version. Finally, when compared to the refined LLM, Logistic Regression models perform well at a range of recall rates. indicating that Logistic Regression should still play a significant role in text classification.

Zhu et al. (2020) were interesting trying to provide solutions to the challenges and limitations between artificial intelligence and human language. The intent of their study was to improve the model performance of a retrieval-and generation-based dialog system by augmenting the system with a query-to-answer dataset thereby creating a many to many dialogues corpus.

Their method consisted of three steps and is shown in Figure 2. (1) extracting keywords from the dialog history (retrieval system), (2) creating expanding the keywords using word embeddings and external knowledge bases, and (3) generating new utterances based on the expanded keywords.

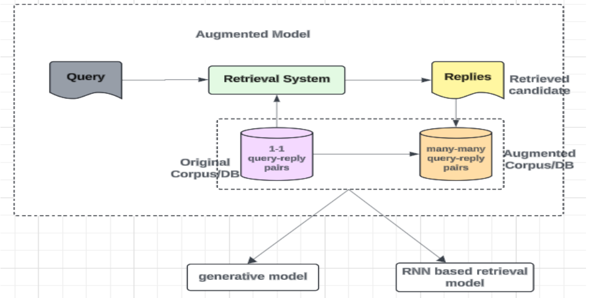


Figure 2: Dialog system for RAG application

They used two benchmark datasets and showed that it can enhance the diversity and relevance of the responses, as well as the overall quality of the dialog systems. The study results also demonstrated that as the amount of data used for training increases, the retrieval and generative models perform remarkably better when the original dataset size is set to 10K.

This helped to demonstrate how important data augmentation is. A thorough case study demonstrates their framework's capacity to produce more varied and significant expressions. The results also indicated a limit to how much data may be added to the training dataset because of the opposite performance in both the retrieval and generative dialog systems as the amount of the original data increases.

Jin et al. (2023) aimed to augment LLMs with domain specific tools leading to the reduction of hallucinations and the retrieval of specialized knowledge. The authors presented a novel method called GeneGPT which uses web APIs to answer questions relating to genomics. They used in-context learning and an enhanced decoding algorithm that recognizes and executes API requests to prompt Codex to answer the GeneTuring tasks using NCBI Web APIs. With an average score of 0.83, the results demonstrated that GeneGPT achieves state-of-the-art performance on eight tasks in the GeneTuring benchmark, significantly outperforming retrieval-augmented LLMs like the new Bing (0.44), biomedical LLMs like BioMedLM (0.08) and BioGPT (0.04), GPT-3 (0.16), and ChatGPT (0.12).

Dou et al. (2023) proposed a novel Large Language Model (ShennongGPT) specifically designed for the Chinese language to provide medication guidance and predict adverse drug reactions. ShennongGPT employs a two-stage training strategy and is illustrated in Figure 3.

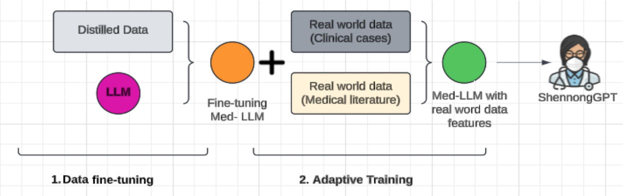

Distilled Data
LLM
Fine-tuning Med- LLM
+
Real world data (Clinical cases)
Real world data (Medical literature)
Med-LLM with real word data features
ShennongGPT
1. Data fine-tuning
2. Adaptive Training

Figure 3: ShennonGPT architecture

Initially, the model learns from distilled drug databases to gain foundational knowledge on drug interactions. Subsequently, it simulates human-like decision-making processes using real-world patient data, which enhances the model's relevance and applicability in providing guidance. This approach allows ShennongGPT to excel in predicting potential adverse drug reactions and offering personalized medication advice, aiming to improve medication safety and healthcare quality. Rigorous evaluations by healthcare professionals and AI experts have underscored the superiority of ShennongGPT over existing general and specialty LLMs. However, the authors emphasize that while ShennongGPT is dedicated to research and holds promise for healthcare applications, it is not intended to replace professional medical advice. The accuracy of responses generated by LLMs cannot be guaranteed, and in cases of discomfort or distress, seeking the guidance of a qualified medical professional is strongly recommended.

Ascorbe et al. (2023) offered a novel Retrieval Augmented Generation (RAG) system designed to improve the accessibility and quality of information available for suicide prevention. Their system endeavored to address the challenge of providing accurate and reliable information to individuals seeking help in this critical area. The RAG system leverages advanced computational techniques to enhance the process of information retrieval and generation, ensuring that users receive the most relevant and supportive content.

The authors detailed the architecture of the system, which integrates a sophisticated search mechanism with a generative model, enabling it to produce responses that are not only factually correct but also contextually appropriate. The paper emphasizes the importance of such a system in the context of mental health and suicide prevention, where timely and accurate information can be lifesaving. Their research and model contributed to the field by offering a potential solution to bridge the gap between the vast amount of information available and the specific needs of individuals seeking help. The system's design is grounded in the latest advancements in artificial intelligence and machine learning, promising to enhance the capabilities of existing digital platforms in providing support for suicide prevention. The authors' work is a significant step towards harnessing technology to address complex health challenges and improve outcomes in public health domains.

Ovadia et al. (2023) compared two prevalent methods for enhancing the knowledge base of large language models (LLMs): unsupervised fine-tuning and retrieval-augmented generation (RAG). The study scrutinized the efficacy of these approaches across various knowledge-intensive tasks and domains. The study revealed that while unsupervised fine-tuning can lead to some improvements, RAG consistently surpasses it in performance. This is true for both knowledge previously encountered during the training of the LLMs and entirely new information. The authors also discovered that LLMs face challenges in assimilating new factual data through unsupervised fine-tuning. However, they found this can be mitigated by exposing the models to multiple variations of the same fact during the training phase. This research contributed to the understanding of how LLMs can be adapted to incorporate new knowledge and refine their existing capabilities, which is crucial for their application in dynamic and specialized fields.

Ren, Guo, Xu, and Xiao (2023) presented a framework for generating guided more diverse and human-like questions, which is a significant contribution to the field of natural language processing. The authors propose a three-stage process: retrieve, generate, and rerank, to produce questions that closely mimic the way humans inquire. Initially, the framework retrieves relevant information from a vast dataset, ensuring that the generated questions are grounded in factual content. Subsequently, the generation phase employs advanced language models to formulate coherent and contextually appropriate questions. Finally, the reranking stage evaluates the questions, prioritizing those that most effectively reflect human curiosity and information-seeking behavior. Their method not only streamlines the question generation process but also enhances the quality of the questions produced, making them more useful for applications such as virtual assistants and educational tools. The paper's findings indicate that this approach can significantly improve the relevance and human-likeness of generated questions, marking a step forward in the development of intelligent systems capable of engaging in meaningful dialogue.

Ahn, Lee, Shim, and Park (2022) introduced a retrieval-augmented response generation model designed for knowledge-grounded conversations. Their model was distinct in its ability to retrieve a range of documents relevant to both the topic and local context of a conversation, which it then uses to generate informed responses. Unlike previous models that focused on single documents or disregarded the conversation topic, this approach considers multiple representations derived from both the conversation's topic words and the tokens preceding the response. Their model's innovative data-weighting scheme is noteworthy as it encourages the generation of knowledgeable responses without relying on ground truth knowledge. The authors' evaluations, both automatic and human, indicate that their model outperforms state-of-the-art models in generating responses that are more knowledgeable, diverse, and contextually relevant.

Zhang, Xiao, Liu, Dou, and Nie (2023) was interested in addressing the challenges faced by large language models (LLMs) due to their inherent limitations in knowledge, memory, alignment, and action. The authors argued that these limitations cannot be overcome by LLMs alone and proposed a novel approach they called the LLM-Embedder. This approach aims to support the diverse retrieval augmentation needs of LLMs through a unified embedding model. The LLM-Embedder is designed to handle various retrieval tasks that capture distinct semantic relationships, which are often subject to mutual interference. The paper details a systematic optimization of the training methodology for the LLM-Embedder, including reward formulation based on LLMs' feedback, stabilization of knowledge distillation, multi-task fine-tuning with explicit instructions, and homogeneous in-batch negative sampling. These strategies have led to the LLM-Embedder outperforming both general-purpose and task-specific retrievers in various evaluation scenarios. The authors have made their checkpoint and source code publicly available, contributing to the field of information retrieval and the enhancement of LLMs' capabilities.

Feng, Feng, Zhao, Yang, and Qin (2024) proposed an innovative approach to augmenting large language models (LLMs) through a retrieval-generation synergy. Their method sought the challenge of obtaining effective documents for knowledge-intensive tasks. Traditional methods rely on either retrieving information from an external knowledge base or generating documents directly from LLMs. The authors proposed an iterative retrieval-generation collaborative framework that leverages both parametric (inherent model knowledge) and non-parametric (external databases) knowledge sources. This synergy allows the model to find the correct reasoning path, which is crucial for multi-step reasoning tasks. The framework, named ITRG (Iterative Retrieval-Generation), consists of two steps: generation augmented retrieval (GAR) and retrieval augmented generation (RAG). In GAR, the model expands queries with pseudo-documents generated by LLMs, while in RAG, it uses retrieved documents to inform further document generation. They report empirical results from experiments on four question answering datasets, including single-hop and multi-hop QA tasks, showing that ITRG significantly improves the reasoning abilities of LLMs and outperforms previous baselines.

3. METHODS

This section will discuss the methods used for the implementation of the RAG application. The methods selected and used in this paper were based on those that were felt would produce the most optimum results for healthcare related prompts.

Design Overview

The proposed RAG system consists of a language model (Llama), a retrieval system and a chatbot. Amazon bedrock is a platform containing several foundation models that can be utilized in designing Gen AI applications.

The retrieval system and chatbot are designed using python programming language. Using Langchain library, we import bedrock and bedrock embeddings from the Amazon bedrock cloud service. We use AWS CLI (Command line interface) to interact with AWS services using commands.

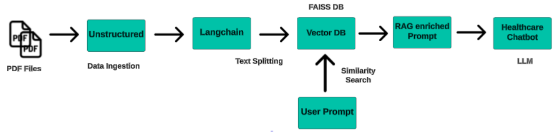


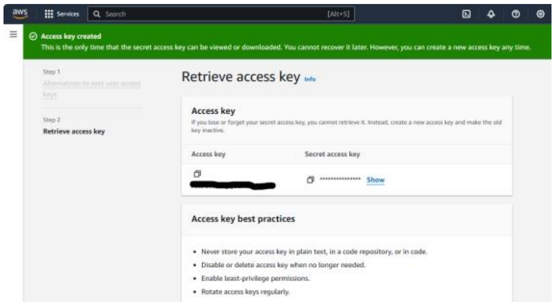
Figure 4: RAG Architecture

Figure 4 illustrates our RAG architecture and how it involves ingesting our dataset into the system as unstructured data, the domain specific data which is in the form of PDF files is gotten from reliable healthcare sources such as cdc.gov, cancer.gov and so on.

Using Langchain, a python library for importing LLMs from Amazon bedrock, we split the text in the PDF files and store them as vector embeddings in the FAISS database. For user prompts, a similarity search is done based on the vector embeddings in the vector store and a response that is augmented with specific healthcare information is provided. The data to be augmented into the foundation LLM is ingested into the application using a PyPDF loader function from Langchain. The text in the documents is then split using a text splitter (Recursive character splitter).

We accessed the Llama 2 foundation model and our vector embedding (Amazon Titan embedding) using access keys as retrieved from Amazon bedrock cloud service. A secret access key and an access key is provided for validation by our RAG application using the AWS command line interface and is illustrated in Figure 5.

To address the identified gaps in the literature, specifically the need for more robust models capable of handling research related to LLMs or base foundation models, fine-tuning and retrieval augmented generation all in one suite, we employed the use of Amazon bedrock, an AWS cloud service known for its performance in such use cases.



Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Retrieve access key

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
[redacted]	***************

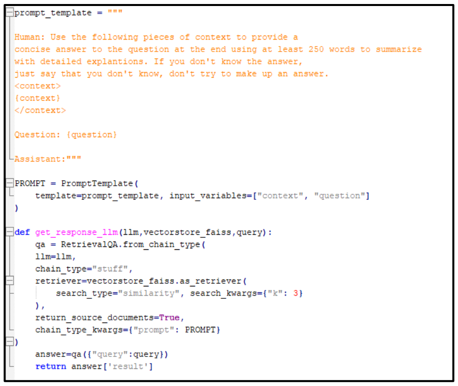
Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

Figure 5: Retrieving access keys on AWS bedrock

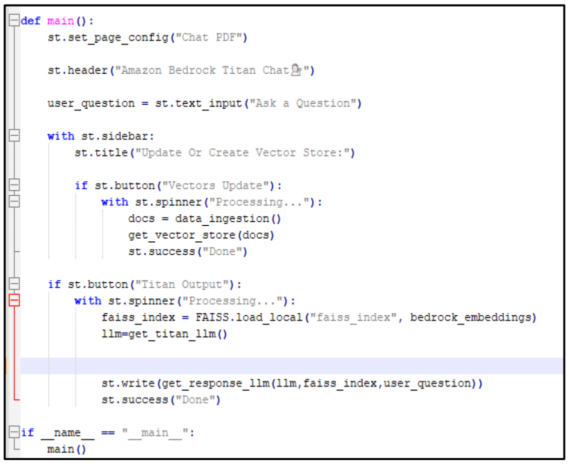
ISSN: 2473-4901
v10 n6170

We used Streamlit for our user-facing chatbot that provided answers to healthcare related prompts. Figure 6 shows the sample code for the prompt template, and Figure 7 shows the code for Streamlit chatbot.



```
prompt_template = """

Human: Use the following pieces of context to provide a
concise answer to the question at the end using at least 250 words to summarize
with detailed explantions. If you don't know the answer,
just say that you don't know, don't try to make up an answer.
<context>
{context}
</context>

Question: {question}

Assistant:"""

PROMPT = PromptTemplate(
    template=prompt_template, input_variables=["context", "question"]
)

def get_response_llm(llm,vectorstore_faiss,query):
    qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=vectorstore_faiss.as_retriever(
        search_type="similarity", search_kwargs={"k": 3}
    ),
    return_source_documents=True,
    chain_type_kwargs={"prompt": PROMPT}
)
    answer=qa({"query":query})
    return answer['result']
```

Figure 6: Sample code for Prompt template



```
def main():
    st.set_page_config("Chat PDF")

    st.header("Amazon Bedrock Titan Chat💁")

    user_question = st.text_input("Ask a Question")

    with st.sidebar:
        st.title("Update Or Create Vector Store:")

        if st.button("Vectors Update"):
            with st.spinner("Processing..."):
                docs = data_ingestion()
                get_vector_store(docs)
                st.success("Done")

    if st.button("Titan Output"):
        with st.spinner("Processing..."):
            faiss_index = FAISS.load_local("faiss_index", bedrock_embeddings)
            llm=get_titan_llm()


            st.write(get_response_llm(llm,faiss_index,user_question))
            st.success("Done")

if __name__ == "__main__":
    main()
```

Figure 7: Sample and Streamlit chatbot

Evaluation metrics

To evaluate the quality of our responses in comparison to other base LLMs, we employed some evaluation techniques that measure automatically generated responses versus a set of reference(s). Consequently, we utilized "**ROUGE**" and "**METEOR**" as evaluation metrics and "**WikiMed**" as our reference or "ground truth". WikiMed is the largest encyclopedia for medical and health related articles.

ROUGE- Recall-oriented understudy for gisting evaluation; measures the overlap of unigrams (single words) between the system summary and reference summary.

$$Recall = \frac{Number\ of\ overlapping\ unigrams}{Total\ number\ of\ unigrams\ in\ reference}$$

METEOR- Metric for Evaluation of Translation with Explicit Ordering; is a metric used for evaluating machine translation output. It calculates the similarity between the system output and the reference translations.

$$Recall = \frac{m}{Wr}$$

Human evaluation is also a valid evaluation technique despite the subjective nature of it. Experts in the medical field such as doctors, psychologists, physicians can evaluate the quality of the automatically generated text (RAG response) and rate it versus the base model outputs.

The decision to use rouge and meteor for our evaluation stems from their efficacy in accurately calculating similarities between machine generated outputs, coupled with their lack of use in the literature studied, with other researchers preferring to use human evaluation, cosine similarity or other F1 measurements. Ascorbe et al. (2023)

In this research, we have opted to use the Rouge and Meteor metrics for evaluation, however, there are other metrics that can be applied to evaluate the quality of the machine generated output/response.

RESULTS

To evaluate the quality of the response generated by the RAG chatbot (Augmented Llama model), we compare it with chat GPT 3.5 and a base Llama model.

Using multiple short prompts relating to chronic diseases such as; **What is Hepatitis? What is Diabetes? What is Tuberculosis?** etc. The responses are evaluated against a reference text gotten from "WikiMed".

Three scenarios were tested using responses less than 250 characters response, more than 250 characters response and a combination of more than one LLM response.

Using the "**Rouge 1**" and "**Meteor**" metrics, the results show that;

-**Augmented Llama model** outperforms other models using Rouge 1 and < 250 characters
-**Chat GPT 3.5** outperforms other models using Rouge 1, Meteor and > 250 characters
-**Base Llama model** outperforms other models using Meteor and with <250 characters

However, the results considerably improved when we combined two models and evaluated it against the reference output.

LLM	ROUGE 1	METEOR
ChatGPT 3.5 model	0.28	0.21
Augmented Llama model	**0.34**	0.23
Base Llama model	0.28	**0.24**

Table 1: LLM responses < 250 characters

LLM	ROUGE 1	METEOR
ChatGPT 3.5 model	**0.44**	**0.26**
Augmented Llama model	0.42	0.23
Base Llama model	0.41	0.18

Table 2: LLM responses >250 characters

LLM	ROUGE 1	METEOR
ChatGPT 3.5 model + Augmented Llama	**0.47**	**0.35**
Chat GPT 3.5 model	0.44	0.26
Augmented Llama model	0.42	0.23
Base Llama model	0.41	0.24

Table 3: LLM combined responses vs standalone responses

Table 1. shows that for responses smaller than 250 characters, the augmented llama and chat GPT models perform well using the rouge and meteor evaluation metrics respectively. Table 2. On the other hand, has chat GPT outperforming the augmented llama model for responses greater than 250 characters on both rouge and meteor measures. These results lead us to believe that with a larger character count, the chat GPT LLM outperforms the augmented llama model. However, as seen in Table 3. We record a 7% and 35% increase in the rouge and meteor scores respectively, when we combine responses from both LLMs.

In comparison with studies done by Zhu et al. (2020), with a larger corpus the model performance improves but starts to decline after the corpus surpasses 1M characters (query and response). Previous research also shows a positive correlation between the evaluation metric BLEU and human evaluation.

Furthermore, these results provide answers to our research questions.

RQ1- Our results show that retrieval augmented generation reduces hallucination of LLMs.

RQ2- The combination of LLM responses provided an improvement in our evaluation scores using Rouge and Meteor with a 7% and 35% increase respectively.

CONCLUSION

The research goal of this study is to create a RAG chatbot using Gen AI and cloud services to further the scarce research around healthcare chatbots and/or virtual assistants and improve hallucination challenges in LLMs for healthcare related prompts.

Our study shows the potential of using RAG systems for healthcare chatbots in terms of improving accuracy of the responses generated by these chatbots. Augmenting the base LLM models with external healthcare data improves the information available to individuals and enhances their access to healthcare knowledge.

While this study exposes a viable use-case for RAG-based LLMs in the healthcare domain, there are some limitations to the study. The small and limited dataset used for this study leaves room for limited generalizability and higher chances of statistical anomalies. Another limitation also lies in the fact that our study only makes use of three LLMs.

Future work can go a step further by carrying out similar research using other languages and for individuals without access to or bad internet who need accurate healthcare information. Larger datasets and the use of other LLMs can also be studied in the future.

REFERENCES

Ahn, Y., Lee, S.-G., Shim, J., & Park, J. (2022). Retrieval-augmented response generation for knowledge-grounded conversation in the wild. *IEEE Access, 10*, 131374-131385.

Ascorbe, P., Campos, M. S., Domínguez, C., Heras, J., & Terroba-Reinares, A. R. (2023). *Towards a Retrieval Augmented Generation System for Information on Suicide Prevention.* Paper presented at the 2023 IEEE EMBS Special Topic Conference on Data Science and Engineering in Healthcare, Medicine and Biology.

ISSN: 2473-4901
v10 n6170

Biswas, M. R., Islam, A., Shah, Z., Zaghouani, W., & Belhaouari, S. B. (2023). *Can ChatGPT be Your Personal Medical Assistant?* Paper presented at the 2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS).

Chen, T., Allauzen, C., Huang, Y., Park, D., Rybach, D., Huang, W. R., . . . Moreno, P. J. (2023). *Large-scale language model rescoring on long-form data.* Paper presented at the ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Dou, Y., Zhao, X., Zou, H., Xiao, J., Xi, P., & Peng, S. (2023). *ShennongGPT: A Tuning Chinese LLM for Medication Guidance.* Paper presented at the 2023 IEEE International Conference on Medical Artificial Intelligence (MedAI).

Feng, Z., Feng, X., Zhao, D., Yang, M., & Qin, B. (2024). *Retrieval-generation synergy augmented large language models.* Paper presented at the ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Jin, Q., Yang, Y., Chen, Q., & Lu, Z. (2023). Genegpt: augmenting large language models with domain tools for improved access to biomedical information. arXiv. In.

Ovadia, O., Brief, M., Mishaeli, M., & Elisha, O. (2023). Fine-tuning or retrieval? comparing knowledge injection in llms. *arXiv preprint arXiv:2312.05934*.

Ren, Z., Guo, J., Xu, Y., & Xiao, B. (2023). *Retrieve, Generate and Rerank: Simple and Effective Framework for Guided Human-Like Questions Generation.* Paper presented at the 2023 8th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC).

Wei, F., Keeling, R., Huber-Fliflet, N., Zhang, J., Dabrowski, A., Yang, J., . . . Qin, H. (2023). *Empirical Study of LLM Fine-Tuning for Text Classification in Legal Document Review.* Paper presented at the 2023 IEEE International Conference on Big Data (BigData).

Zhang, P., Xiao, S., Liu, Z., Dou, Z., & Nie, J.-Y. (2023). Retrieve anything to augment large language models. *arXiv preprint arXiv:2310.07554*.

Zhu, Q., Wang, X., Chen, C., & Liu, J. (2020). *Data augmentation for retrieval-and generation-based dialog systems.* Paper presented at the 2020 IEEE 6th International Conference on Computer and Communications (ICCC).

APPENDIX

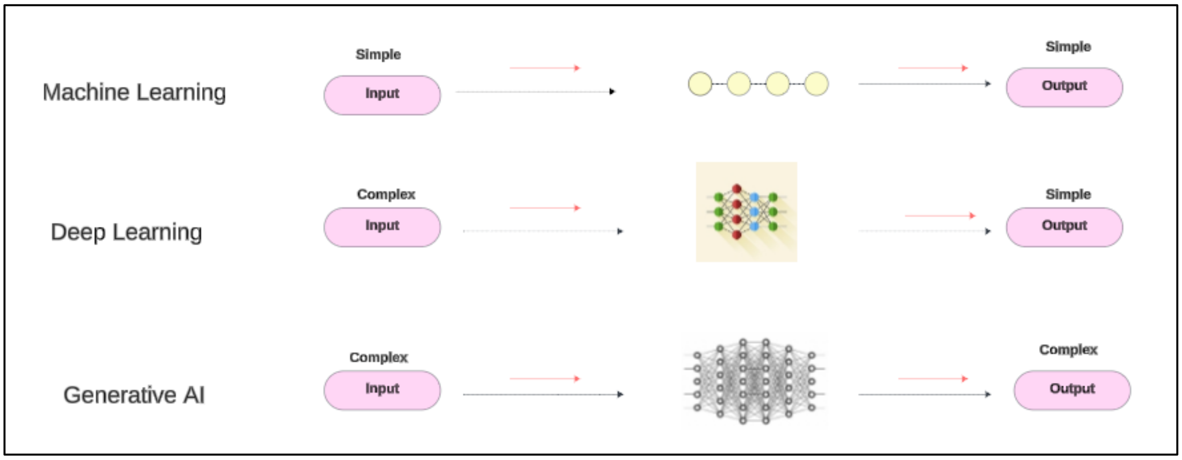


Figure 1: ML, DL, and Gen AI data flow comparison

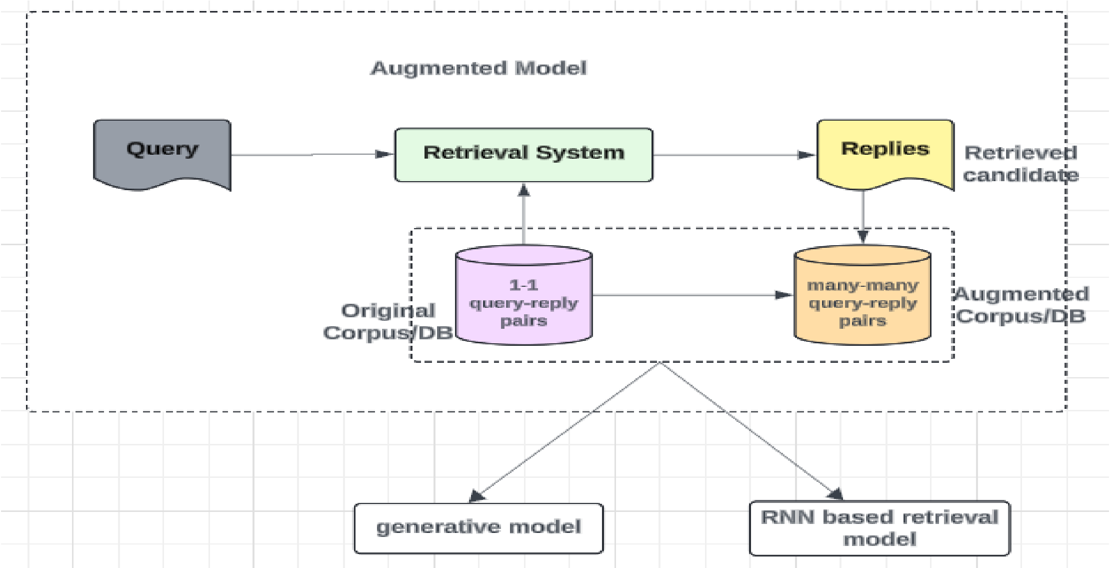


Figure 2: Dialog system for RAG application

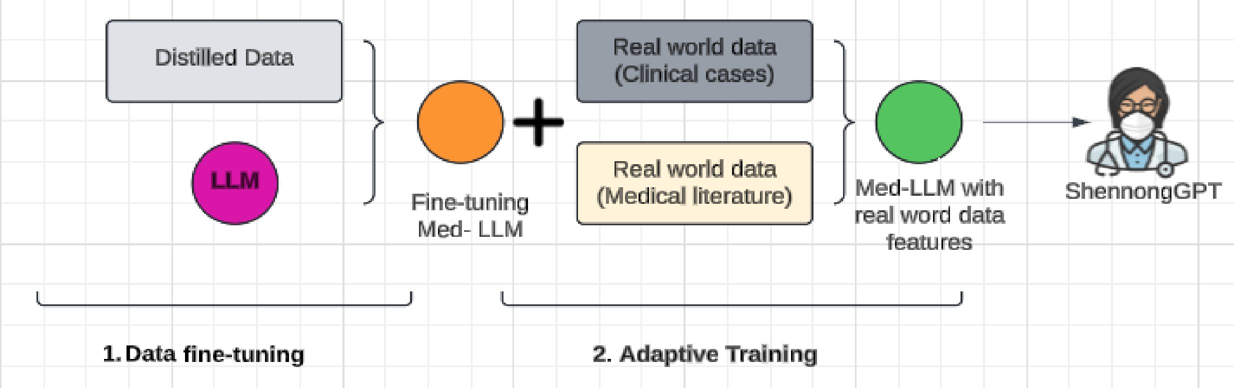

Distilled Data
LLM
Fine-tuning Med- LLM
Real world data (Clinical cases)
Real world data (Medical literature)
Med-LLM with real word data features
ShennongGPT
1. Data fine-tuning
2. Adaptive Training

Figure 3: ShennonGPT architecture

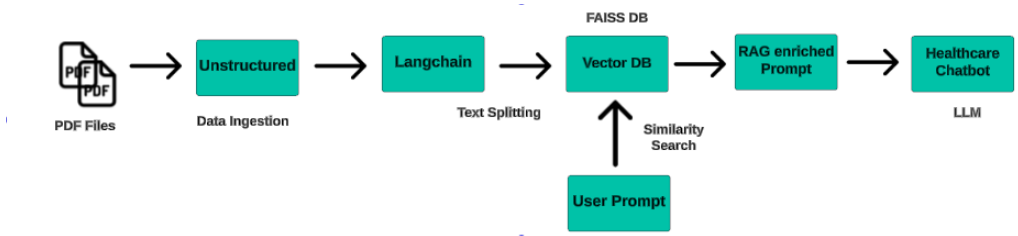
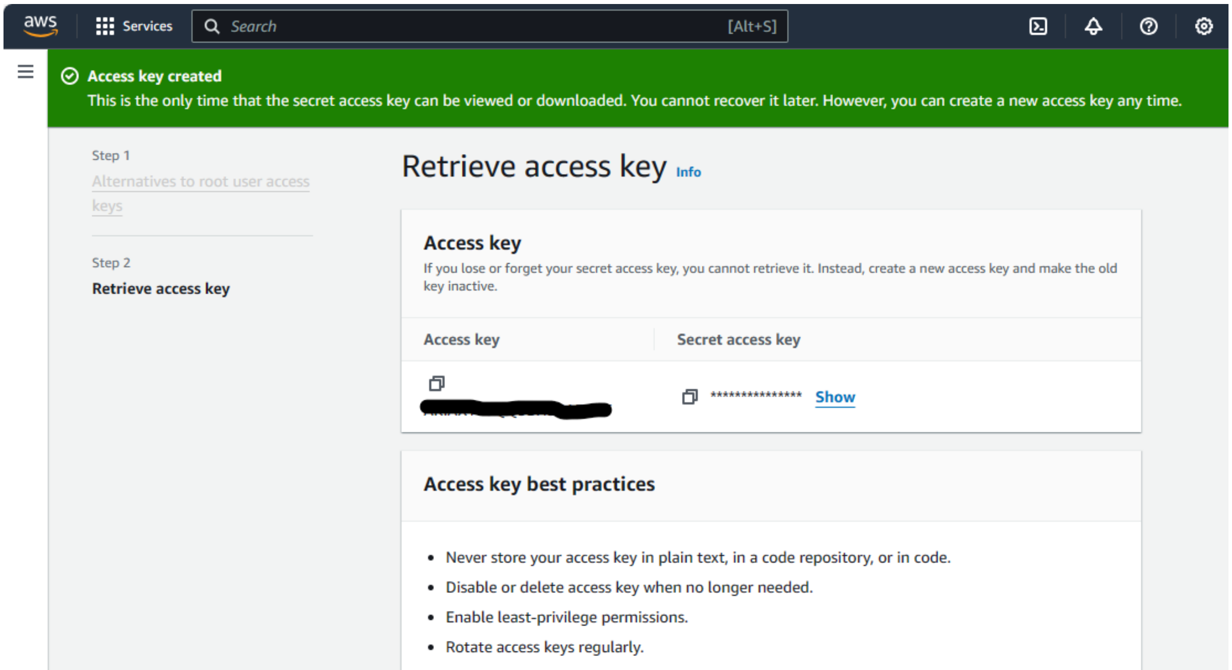


Figure 4: RAG Architecture



Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.

Step 1
Alternatives to root user access keys

Step 2
Retrieve access key

Retrieve access key

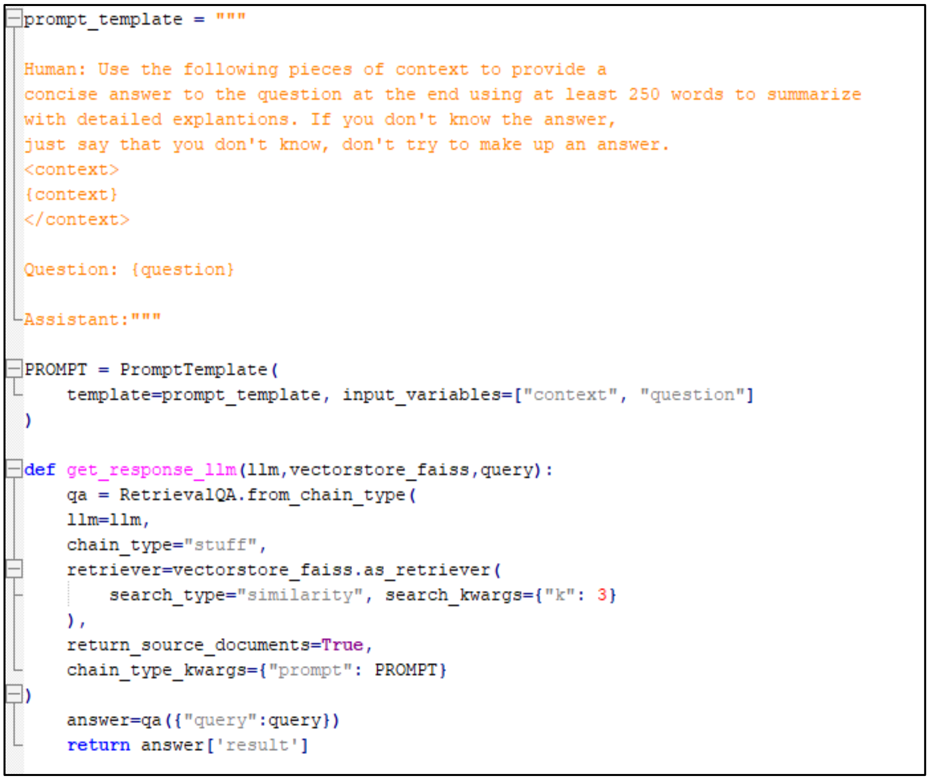
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
[redacted]	*************** Show

Access key best practices

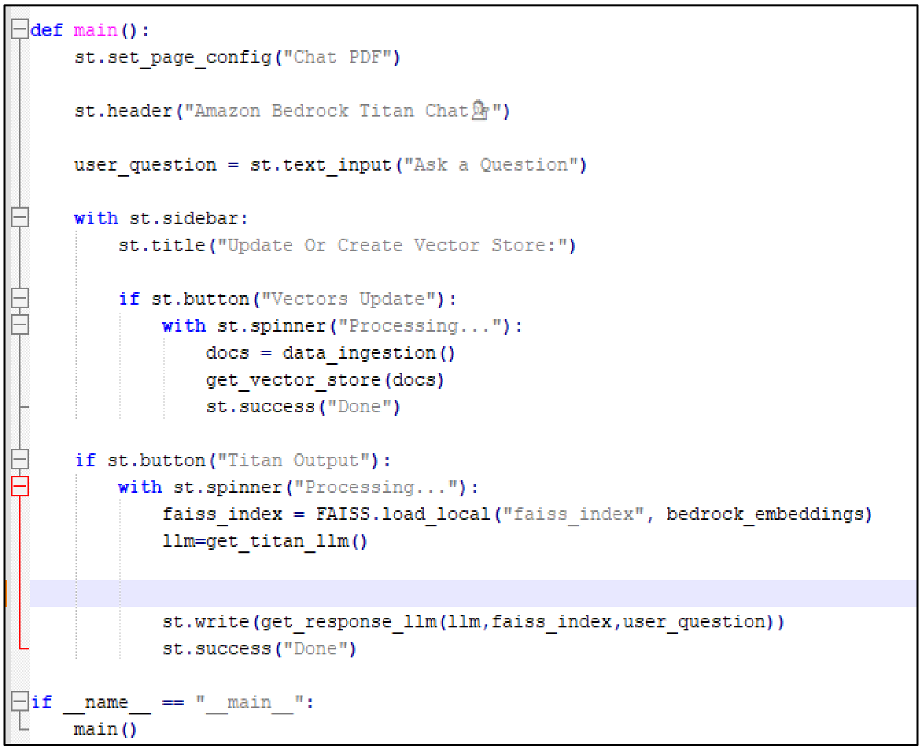
- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

Figure 5: Retrieving access keys on AWS bedrock



```
prompt_template = """

Human: Use the following pieces of context to provide a
concise answer to the question at the end using at least 250 words to summarize
with detailed explantions. If you don't know the answer,
just say that you don't know, don't try to make up an answer.
<context>
{context}
</context>

Question: {question}

Assistant:"""

PROMPT = PromptTemplate(
    template=prompt_template, input_variables=["context", "question"]
)

def get_response_llm(llm,vectorstore_faiss,query):
    qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=vectorstore_faiss.as_retriever(
        search_type="similarity", search_kwargs={"k": 3}
    ),
    return_source_documents=True,
    chain_type_kwargs={"prompt": PROMPT}
)
    answer=qa({"query":query})
    return answer['result']
```

Figure 6: Sample code for Prompt template



```
def main():
    st.set_page_config("Chat PDF")

    st.header("Amazon Bedrock Titan Chat💁")

    user_question = st.text_input("Ask a Question")

    with st.sidebar:
        st.title("Update Or Create Vector Store:")

        if st.button("Vectors Update"):
            with st.spinner("Processing..."):
                docs = data_ingestion()
                get_vector_store(docs)
                st.success("Done")

    if st.button("Titan Output"):
        with st.spinner("Processing..."):
            faiss_index = FAISS.load_local("faiss_index", bedrock_embeddings)
            llm=get_titan_llm()


            st.write(get_response_llm(llm,faiss_index,user_question))
            st.success("Done")

if __name__ == "__main__":
    main()
```

Figure 7: Sample and Streamlit chatbot