MediMate: Medical Q&A App

Hyrum Eddington, PhD[1], Sanidhya Singal, MS[1], Shyam Renjith, MS[1]
[1]University of California San Diego, La Jolla, California, USA

Abstract
Medical chatbots using Large Language Models (LLMs) have the potential to improve patient experience and reduce physician workload. However, challenges regarding inaccurate responses due to hallucination make non-augmented LLM chatbots problematic. In this paper, we present MediMate, a medical Q&A application based on LLM technology that further leverages retrieval-augmented generation (RAG) to provide accurate chatbot responses, where RAG is implemented using the PubMedQA, NFCorpus, and CORD-19 datasets. MediMate application features include querying with and without RAG, speech input, and patient medical information upload for additional RAG context. Chatbot performance is consistently better with RAG as evidenced by higher ROUGE and BLEU scores for each dataset implemented, and patient questions related to dataset knowledge domains show repeated additional detail in the chatbot responses when using RAG. Although further exploration of additional data sources, such as knowledge graphs, is required to further reduce the probability of hallucination, we conclude that RAG shows great promise in improving the accuracy and detail of medical chatbot responses.

Introduction

Chatbots in Medical Contexts

The interest in large language models (LLMs), further spurned by the success of ChatGPT, has resulted in a surge of interest in leveraging this technology to improve patient experiences [4]. Medical chatbots capable of quickly and accurately answering patient questions may provide increased access to medical advice. Chatbots can be available at times when a physician is not present, and can provide quicker answers to queries than waiting for a nurse or physician to review a patient's question. Additionally, chatbots can provide general health information to communities that historically may have had reduced access and utilization of healthcare services, meaning that they have the added potential to ameliorate the informational disparity relating to health literacy among disadvantaged groups [9]. Appropriate use of chatbots also reduces physician workload as healthcare professionals may not be required to spend as much time answering basic medical questions.

LLMs and Hallucination

Due to the nature of how LLMs are trained, they do not intrinsically have access to websites, scientific articles, or other concrete resources related to a potential patient query. Rather, responses are generated probabilistically, at times resulting in 'hallucinations' [3] or misinformation that may sound correct from context but is medically inaccurate if cross-checked with medical sources. This presents a significant problem as false medical advice has the potential to spread misinformation regarding human health, or worse, adversely affect patient health through poor medical advice. Therefore, additional efforts are required to refine the use of these LLMs such that they provide medically accurate information, and the probability of hallucination is reduced.

Combining LLMs with RAG for Accurate Chatbot Responses

Retrieval-Augmented Generation (RAG) [14] is a method being currently explored for its potential to reduce LLM hallucination and increase response accuracy. RAG consists of leveraging a large database based on an additional corpus of field-relevant documents to add additional context to a user's query to an LLM. When the user, or patient in this case, asks the LLM-cased chatbot a question, documents related to that patient's question are retrieved from this database and prepended to the user's query as additional context, allowing the LLM to consider that context in generating its response. This approach has several advantages over the standard (without RAG) approach:

1. Additional context 'guardrails' the LLM and provides additional information to reduce the chances of hallucination.
2. The additional information retrieved includes findings from medical research that provide more refined, accurate query results.
3. The database can be modified and updated without the need to re-train the LLM.

Therefore, in this paper we present MediMate, a patient application that leverages RAG with LLM technology to provide medically-accurate, context aware responses to medical queries.

Datasets

PubMedQA

PubMEDQA [13] is a biomedical QA dataset with ~62k instances comprising several components including biomedical research questions (e.g.: Do pre-operative statins reduce atrial fibrillation after coronary artery bypass grafting?) and contexts, the latter containing a Pubmed abstract containing biomedical research results. This context is the portion we extract for RAG purposes.

NFCorpus

NFCorpus [11] contains ~3,000 natural language queries extracted from NutritionFacts.org along with relevance judgements for ~10,000 medical documents which can be extracted for RAG. Utilizing this dataset allowed us to use a more targeted set of medical documents related to nutrition. This allowed us to construct a database specifically relevant to general health and dietary questions, which we hypothesized might allow for more patient-centric responses than heavy-research jargon datasets such as PubMedQA.

CORD-19

CORD-19 [8] is a corpus of COVID-19 related research papers that was launched in March 2020, and actively curated until its final dataset version released in June 2022. Notably, this means that the full document corpus is further up-to-date than chatGPT, whose training data cutoff is 2021. As COVID-19 continues to be a topic of great importance in medicine, we implemented this dataset as an example of using RAG on an up-to-date research database for LLM querying.

Methods

The overall architecture of our application can be described in Figure 1. When the patient/clinician asks a query to the system, our information retrieval system uses one of aforementioned datasets to add additional context to the query. The context along with the query is passed to a LLM, which then generates a textual response to the patient/clinician.

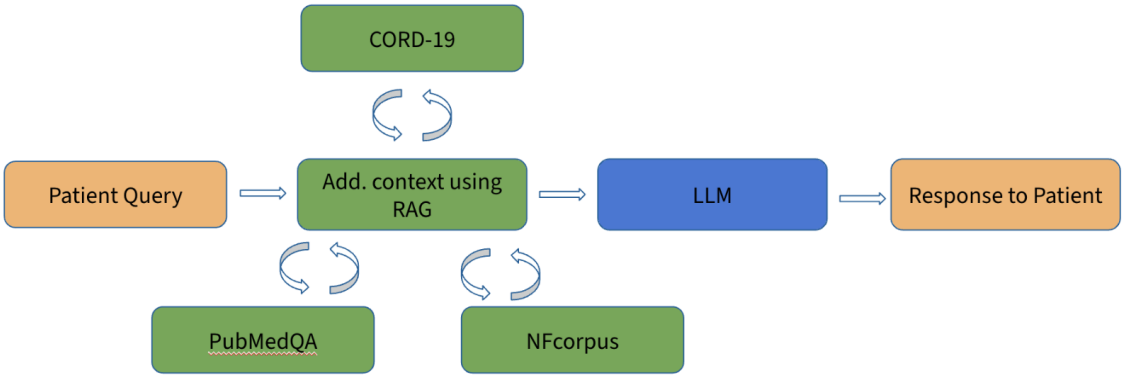

Figure 1. Flowchart depicting our methodology.

The State-of-the-art (SOTA) approach to solve the problem of Closed-Domain Question Answering (CDQA) involves using LLM-based embeddings for effective information retrieval, storing them in a vector database for fast querying, and using a LLM for generating the answer to the user's query. Vector Database Management Systems (VDBMSs), or VectorDBs, are the current state-of-the-art in Information Retrieval [15]. But to understand VectorDBs, we need to first understand vectors.

Deep Learning Embeddings / Vectors

Vectors, derived from Artificial Deep Neural Networks, are numerical representations of data in low-dimensional space. This technique of data representation allows us to represent words, sentences, or entire documents in a continuous vector space. In contrast to methods like one-hot encoding, deep learning embeddings capture semantic nuances and intricate relationships between words, providing a more nuanced understanding of language. Prominent instances in this domain include the **OpenAI Embeddings** [12], particularly the "text-embedding-ada-002-v2" model, recognized for efficiently condensing intricate text into meaningful vectors. The **LLAMA Embeddings** [10],

exemplified by the "llama-2-7b-chat" model, represent a noteworthy evolution in embedding models, playing a vital role in the advancement of natural language processing. In our project, we leverage OpenAI embeddings, mapping queries and documents to a 1536-dimensional space.

Vector DBMSs
Vector embeddings address the limitations of older methods but face computational challenges in storage and comparison, especially with high-dimensional sparse data. VectorDBs [15], specialized database management systems, excel in efficiently handling such data. They employ Approximate Nearest Neighbor (ANN) methods like ANNOY (Approximate Nearest Neighbors Oh Yeah) [2] and FAISS (Facebook AI Similarity Search) [6], along with distance metrics like Euclidean distance or cosine similarity, making them valuable for similarity searches, such as finding document vectors similar to a query vector.

Approximate Nearest Neighbors tackle the task of finding the top-k most relevant documents for a given query. Traditional methods involve computing distances between the query vector and every document, which is computationally expensive. ANNs, like ANNOY [2], introduce randomness to narrow down the candidate search space, sacrificing some accuracy for faster inference. This process involves randomizing the division of space into subspaces, selecting documents in the query's subspace through iterations, resulting in a smaller curated candidate set, as illustrated in Figure 2. The distance between the query vector and each document in the candidate set is then computed to retrieve the top-k most relevant documents.

In essence, ANNs, exemplified by tools like ANNOY, offer an innovative compromise between precision and computational efficiency in the quest for identifying the most relevant documents based on a given query.

Overall, deep learning-based embeddings enhance information retrieval through semantic understanding and contextual relevance. Through VectorDBs, they enable scalable and efficient retrieval. Despite the advantages, these techniques are dependent on domain-specific labeled data for effective training of embeddings, are resource intensive, and suffer from maintenance complexities in ensuring the quality of embeddings as well as VectorDBs.

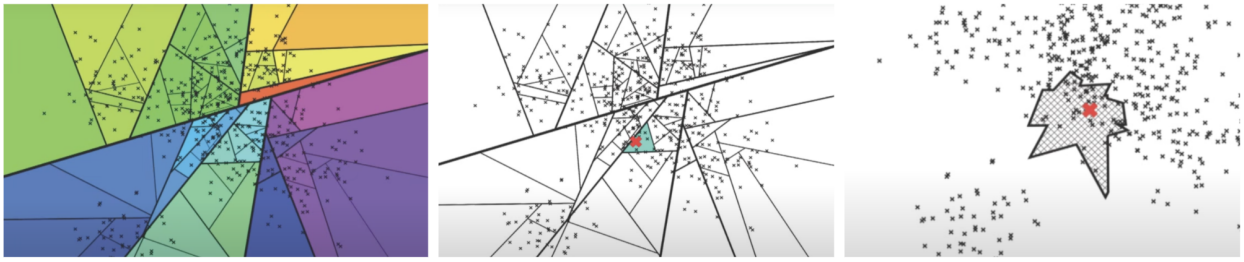

Figure 2. Using ANNOY for finding the top-k most relevant documents to a given query vector (denoted by red X).

Retrieval Augmented Generation
Retrieval Augmented Generation (RAG) is a technique that combines both retrieval-based and generation-based approaches to enhance the efficacy of large language models and mitigate issues like hallucinations. Retrieval-based methods entail fetching pertinent information from a knowledge base or document set based on an input query. This guarantees that the model has access to domain-specific knowledge, reducing the likelihood of hallucinations that might arise from extrapolating information from irrelevant or inaccurate sources.

Retrieval Augmented Generation with LLMs is a significant advancement in the field of Information Retrieval and Question Answering systems. Its advantages include context-aware question-answering, tailored for nuanced queries, and seamless integration with LLMs for precise and contextually relevant generative capabilities, particularly beneficial in specialized domains like medicine.

Ensuring fairness in evaluating an LLM's response with RAG for CDQA in medicine involves:

- A **comparative analysis**, assessing the combined approach against standalone LLMs to gauge the added value of contextual relevance.
- Defining **performance metrics**, including accuracy and response precision.

- Utilizing **diverse datasets and queries** to ensure RAG's effectiveness across various domains and subjects.

Our approach consists of two variants: one uses an LLM alone, resembling Open Domain Question Answering, while the other employs RAG for Closed Domain Question Answering. In the latter, top-k relevant documents are identified based on the user query and presented as additional context to the LLM along with the original query. Our hypothesis is that incorporating RAG enhances the LLM's ability to deliver accurate, contextually relevant answers, underscoring the importance of RAG.

Tools and Libraries

- **Backend and QA Engine**:
  For our backend, we use **Python** (v3.8) as the primary programming language, along with a combination of various tools and libraries to facilitate the implementation of our core API services and the Q&A engine:
  1. **NumPy** (v1.26.2): For efficient numerical computations in Python.
  2. **OpenAI** (v1.3.5): To leverage OpenAI's API for generating embeddings.
  3. **NLTK** (v3.8.1): For text processing, corpus handling, and calculating BLEU scores.
  4. **Python-dotenv** (v1.0.0): To load environment variables for secure management of sensitive information like OpenAI's API keys.
  5. **FAISS-CPU** (v1.7.4): For efficient similarity search of high-dimensional vectors.
  6. **Langchain** (v0.0.344): For developing applications powered by language models; Connects LLM to sources of context (prompt instructions, few-shot examples, content to ground its response in, etc.), thus enabling RAG.
  7. **Rouge_score** (v0.1.2): To calculate ROUGE scores for evaluation purposes
- **Frontend**:
  For our patient/clinician facing responsive UI, **ReactJS** is our primary programming language, coupled with numerous tools and plugins to allow for a seamless communication interface:
  1. **Tailwind CSS** (v3.3.5): For implementing elegant styling of layout and accessibility components.
  2. **webkitSpeechRecognition** (v2.2.2): To implement real-time speech-to-text transcription.
  3. **lottie-react** (v2.4.0): For smooth and relevant animations for aesthetic enhancements in the UI.
  4. **@tabler/icons-react** (v2.44.0): For chat box icons, user profile icons, and animations.
  5. **axios** (v1.6.2): To facilitate asynchronous API calls with CORS from UI components to backend Flask application.
- To dockerize the entire application, we use **docker-engine** (v24.0) and the corresponding "docker-compose.yml" file to ensure cross-platform compatibility and ease of build deployment.

Implementation Details

We meticulously structure our implementation into distinct steps, each contributing to the overall efficacy of our closed-domain question answering model:

- **User Interface**

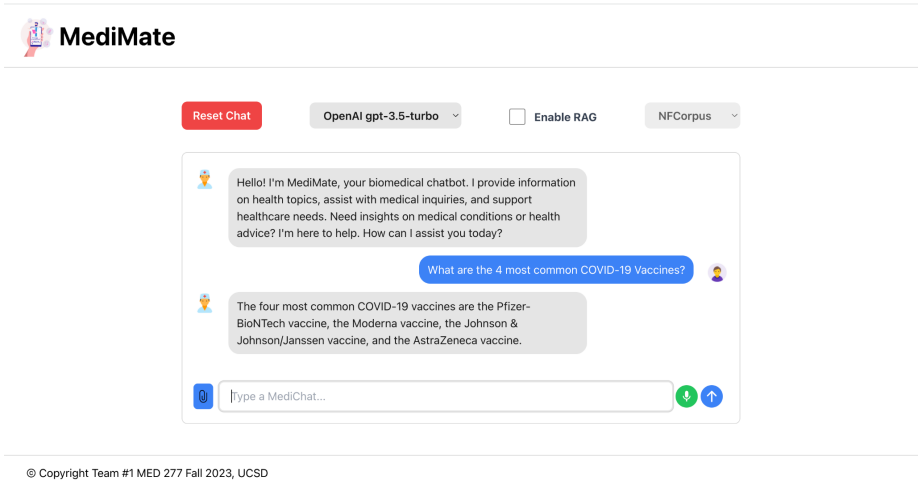

Figure 3. MediMate chat interface and functionalities.

The User Interface (UI) facilitates natural interactions for patients as well as clinicians through a straightforward and user-friendly experience. It supports multi-modal input, allowing users to interact through text, file uploads, or speech, catering to users with different preferences and accessibility needs. It aims to analyze and compare response variations in scenarios with and without RAG across diverse models and datasets that are tailored to medical domains.

Additionally, the user interface plays a vital role as a Proof of Concept (POC) for advanced Medical QA systems. It establishes a groundwork for future innovations in scalable medical conversational interfaces and has the potential to lead to the integration of cutting-edge technologies in healthcare communication, enhancing patient-practitioner interactions and medical information retrieval.

Apart from the aforementioned details, the UI has several additional features:

1. **Speech Recognition and Transcription**: Leveraging advanced speech recognition technology, this feature enables users to verbally communicate their medical queries. The spoken words are transcribed into text, broadening the UI's accessibility for users who prefer or require voice input. This also facilitates a more natural and flexible interaction.
2. **Model Switching:** Users benefit from the flexibility to switch between two distinct OpenAI models. The first, "gpt-3.5-turbo", represents a cutting-edge language model, while the second, "Davinci", serves as a legacy baseline. This feature empowers users to assess and compare responses generated by different models for diverse use cases.
3. **Dataset Switching**: This feature allows users to switch across different datasets. For our Medical QA system, it means the ability to adapt to specific medical domains/specialties. It also means that our system can support several user roles, such as patients, clinicians, and practitioners. In healthcare, where clinicians possess specialized training to quickly process information, the system accommodates the needs of the average person, providing context-aware responses without overwhelming medical jargon. The goal is to promote "Evidence-Based Medicine" by delivering appropriate information tailored to different user roles and levels of expertise.
4. **Conversation History**: The UI meticulously logs the conversation history including all interactions between users and the system. This allows both patients and practitioners to revisit previous messages within a discussion, fostering continuity. It ensures that each conversation is part of an ongoing dialogue, promoting a personalized and informed interaction.
5. **Medical History Upload**: This standout feature allows users to upload their medical history as a file, which may include medical records, test results, and other relevant documents. This data is parsed by the system and saved as context, enabling the users to query the application for hyper-personalized information based on their own medical history. This allows us to take a step towards "Personalized Medicine" as well as "Health Literacy".

- **Backend**
  1. **API Endpoints:** The Flask-based backend employs distinct API endpoints to handle various modes of interaction. The "/v1/chat/simple" endpoint is designed for testing and debugging, while "/v1/chat/openai" utilizes gpt-3.5-turbo for generating responses without RAG. The "/v1/chat/openai/rag" endpoint incorporates RAG for context-aware responses.
  2. **Model Selection:** Model selection plays a pivotal role in response generation. Users can opt for different models based on their preferences and the nature of their queries, offering a nuanced approach to information retrieval and generation.
  3. **Real-time Communication:** The Flask server enables real-time communication between the UI and backend with smooth handling of asynchronous API calls. This ensures that users receive prompt and seamless responses during their interactions, fostering a dynamic and responsive user experience. This real-time communication is crucial for the application's effectiveness in providing timely medical information and advice.

- **Data Pre-processing**

  In this foundational step, we employ "nltk", the Natural Language Toolkit, to perform essential data pre-processing tasks. Documents as well as query texts are converted to lowercase, and words are tokenized using nltk's "word_tokenizer", laying the groundwork for subsequent analyses. To enhance the quality of the text, we go beyond simple tokenization – stop-words and words with lengths less than 2 are

systematically removed; Punctuation is discarded, and the "WordNetLemmatizer" is applied to each word, ensuring that each is transformed to its lemma.

- **OpenAI Embeddings**
  To incorporate contextual understanding, we integrate OpenAI Embeddings into our model. Leveraging OpenAI's API and the "text-embedding-ada-002" model, this step enriches our model's comprehension of intricate language structures, enabling it to grasp context more effectively. Here, we map our documents and queries vectors to a 1536-dimensional vector space.

- **Retriever**
  Efficient document retrieval is facilitated by VectorDBs. We utilize the Facebook AI Similarity Search (FAISS) vector store available in "langchain.vectorstores" to build a VectorDB of embeddings obtained in the previous step. We make sure that each embedding is a unit vector and employ the "dot" product metric to find similarity between any two vectors. (Note that cosine similarity with unit vectors is the same as dot product). This allows us to retrieve the top-k documents, with "k" set to 5, optimizing the relevance of retrieved information in the subsequent steps.

- **Chain**
  This step harnesses the power of OpenAI's language model available in "langchain.llms". Specifically, we base our models on the "gpt-3.5-turbo-instruct" and "da-vinci-0.0.3", enhancing our ability to generate coherent and contextually relevant responses. Furthermore, to perform RAG, we create a chain by first defining a prompt based on a fixed "ChatPromptTemplate" template. Then, we pass the prompt along with the question and the context to the LLM.

```python
from langchain.llms import OpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.schema.output_parser import StrOutputParser
from langchain.schema.runnable import RunnablePassthrough


llm = OpenAI(model_name=model,
             openai_api_key=os.getenv("OPENAI_API_KEY"))

template = """Answer the last question of the conversation,
given this additional context: {context}
Conversation: {question}"""
if medical_history != "":
    template += "\nWhile answering, also consider this as my medical history:\n"
    template += medical_history
prompt = ChatPromptTemplate.from_template(template)

chain = ({"context": retriever | format_docs, "question": RunnablePassthrough()}
         | prompt
         | llm
         | StrOutputParser())
chain.invoke(query)
```

Figure 4. Python code depicting the usage of "langchain" for the purpose of performing RAG using OpenAI's LLM based on the last message in the conversation along with provided medical history, if any.

- **Metrics**
  We evaluate our QA system using the following metrics and report the mean of each metric across all queries for each dataset:
  1. **BLEU Score (Bilingual Evaluation Understudy)**: The BLEU score is commonly used to evaluate the quality of machine-generated text by comparing it to one or more reference texts. BLEU focuses on precision and it involves counting the number of overlapping n-grams (typically

up to 4-grams) between the generated and reference texts.

$$\text{BLEU} = \min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)\left(\prod_{i=1}^{4} \text{precision}_i\right)^{\frac{1}{4}}$$

2. **ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation)**: ROUGE evaluates the quality of summaries by comparing them to one or more reference summaries. It provides a comprehensive measure of the overlap between system-generated summaries and human reference summaries, focusing on recall. For example, ROUGE-N (overlap of n-grams) and ROUGE-L (longest common subsequence), where:

$$\text{ROUGE-N} = \frac{\Sigma_{S \in \text{reference-summaries}} \Sigma_{n\text{-gram} \in S} \text{count}_{\text{match}}(n\text{-gram})}{\Sigma_{S \in \text{reference-summaries}} \Sigma_{n\text{-gram} \in S} \text{count}(n\text{-gram})}$$

By delineating our implementation into these distinct steps, we ensure a systematic and comprehensive approach to building a robust closed-domain question answering system. Each step contributes unique functionalities, collectively culminating in a model that excels in understanding, retrieving, and generating contextually relevant responses. For more implementation details, please refer to our GitHub repository: [shy982/Med-Q&A-App](shy982/Med-Q&A-App).

Experiments

Application in Action

Here, we attempt to show how a typical interaction between a user (clinician or patient) and our MediMate application takes place. We aim to demonstrate how the responses look like:

- On asking a nutrition related question to gpt-3.5-turbo LLM without RAG, we get the following response:

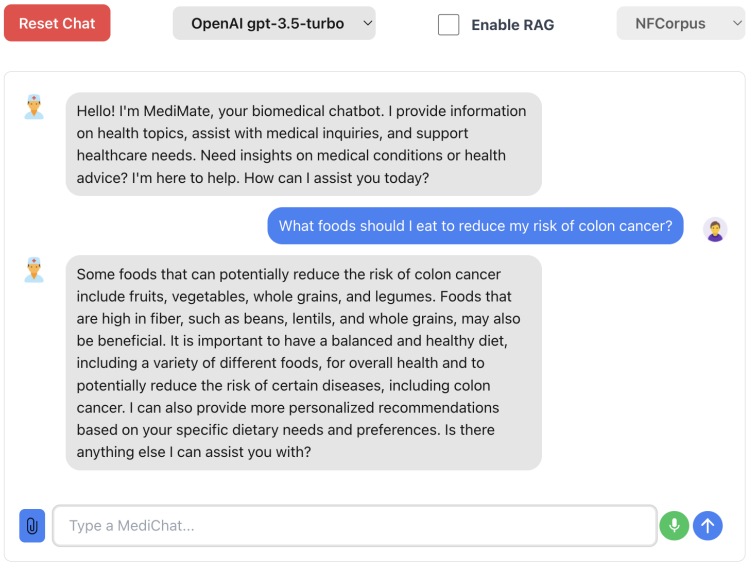

Figure 5. Response to a nutrition related question using gpt-3.5-turbo **without** RAG.

- Now, let's see how this answer changes with RAG turned on and NFCorpus as our preferred database. We also ask a follow up question to check its context awareness and conversation maintaining capability.

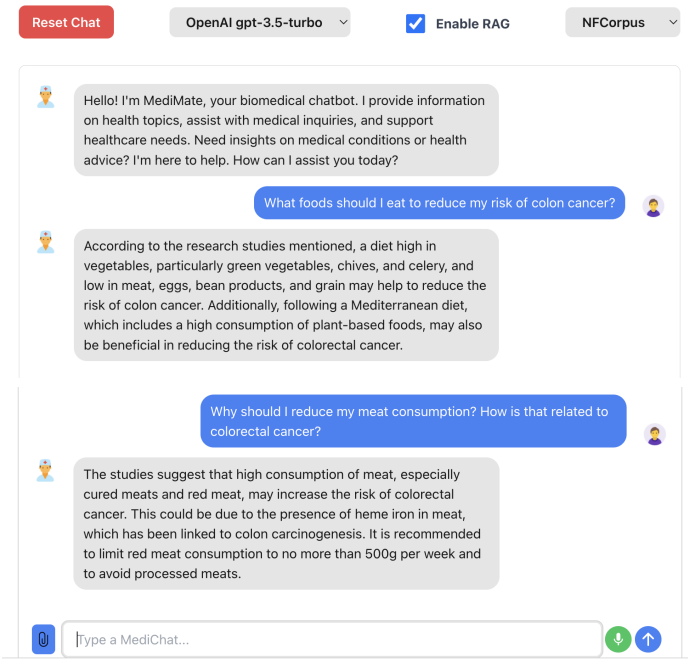

Figure 6. Response to nutrition-related questions using gpt-3.5-turbo **with** RAG on NFCorpus.

- Let's check how RAG performs with the PubMedQA dataset when asked a nuanced clinical research question on CPAP machines.

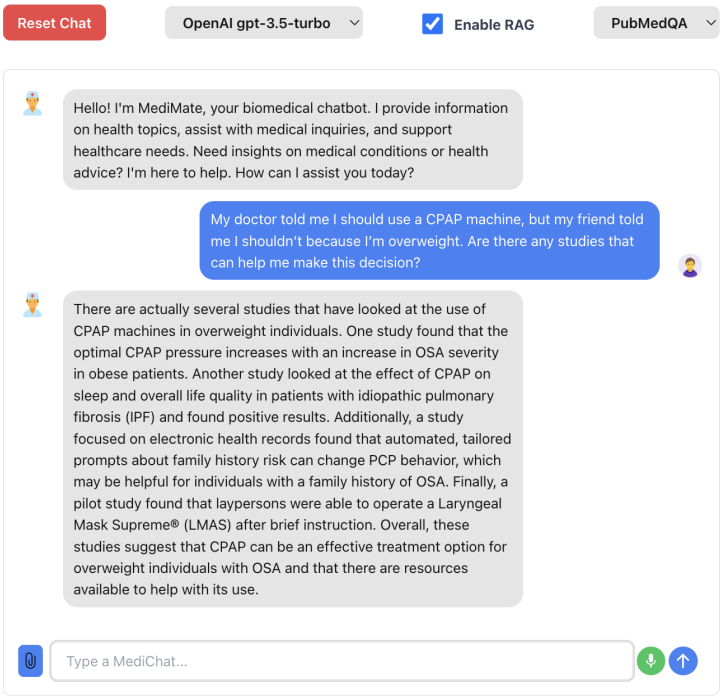

Figure 7. Response to a subjective clinical research related question using gpt-3.5-turbo **with** RAG on PubMedQA.

We see that the response is extremely detailed and answered with nuance. Clearly we see RAG + LLM models outperforming the typical LLM only response.

- Now let's consider a case of **personalized medicine**, in which a patient has the option to upload a medical history file seamlessly through our UI and subsequent responses to questions take into account the pre-existing conditions and medical diagnosis and prescription history of the patient.

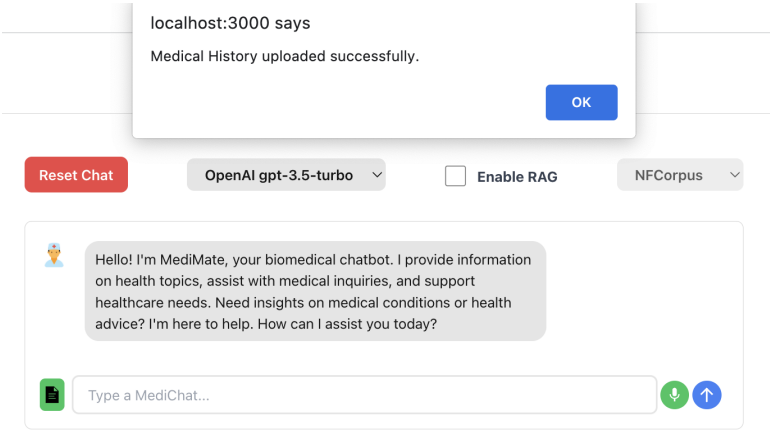

Figure 8. Depiction of patient medical history file upload through the UI.

- Now, suppose the patient asks questions related to their medical history, we should see a more patient-specific answering style where past illnesses and medications are closely taken into account by the model to provide tailored responses.

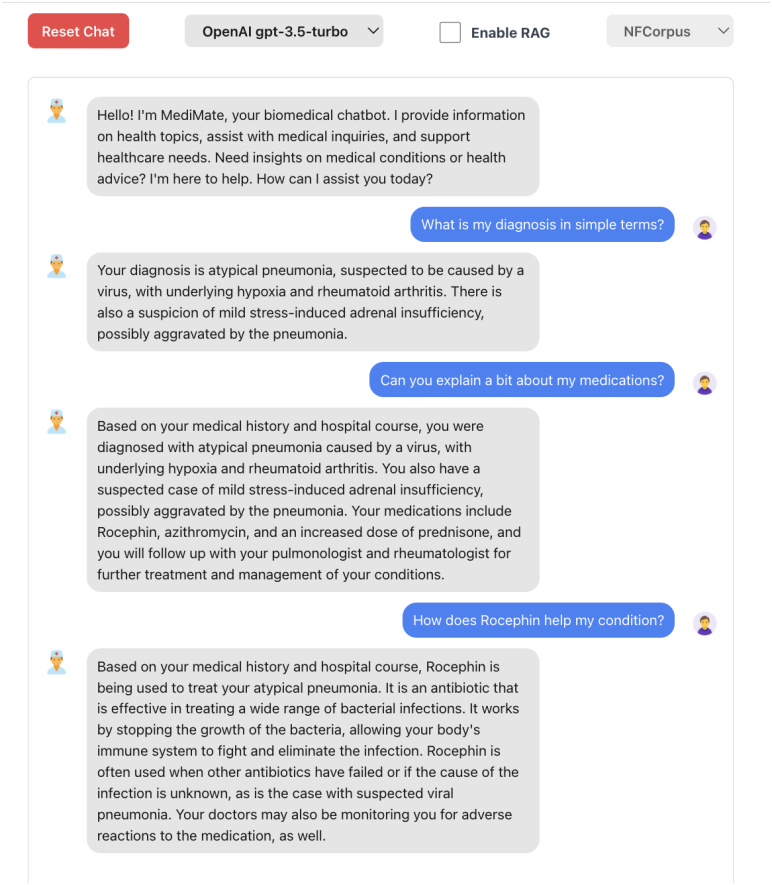

Figure 9. Tailored responses to the patient based on medical history.

From the responses in Figure 9, we clearly see the power of RAG as a technique in being capable to aid an LLM based simplistic chatbot application to be more context aware, domain specific, and in this case, even patient-specific.

While this section explored the qualitative performance of the Q&A Application and RAG in general. Let's now quantitatively analyze the performance of LLMs with and without RAG across multiple datasets.

Results

For each dataset, we report the mean BLEU score and the mean ROUGE score for LLM without RAG as well as LLM with RAG methods in Table 1.

	LLM without RAG		**LLM with RAG**	
Dataset	**Mean BLEU Score**	**Mean ROUGE Score**	**Mean BLEU Score**	**Mean ROUGE Score**
NFCorpus	0.7441	0.2167	0.8377	0.2226
PubMedQA	0.2721	0.2189	0.6937	0.2739
CORD19	0.4709	0.1713	0.6099	0.2024

Table 1. Comparison of LLM with and without RAG approaches across 3 different datasets.

We make the following inferences from the above results:

1. The inclusion of the Retrieval-Augmented Generation technique led to an enhancement in model performance across all datasets. This is evident from the increased Mean BLEU and ROUGE scores with RAG compared to without RAG, and highlights the effectiveness of incorporating RAG into Large Language Models for performing Medical QA.
2. The impact of RAG varies across datasets. The dataset PubMedQA, which involves more complex and technical content, sees a considerable improvement with the inclusion of RAG. In other words, a generic LLM (i.e., without RAG) performs poorly when provided with medical domain-specific questions where the language is more specialized or intricate.
3. As LLMs tend to excel further when provided with richer context, we can also infer that our Information Retrieval system is working as expected, contributing to a more comprehensive contextual understanding for the LLM.

Please note that we report the above results with "gpt-3.5-turbo-instruct" as the LLM, but we also experimented with the "da-vinci-0.0.3" model as a baseline. The results in Table 2 compare the two LLMs without RAG and with RAG (using only the NFCorpus dataset).

	Without RAG		**With RAG**	
LLM	**Mean BLEU Score**	**Mean ROUGE Score**	**Mean BLEU Score**	**Mean ROUGE Score**
da-vinci-0.0.3	0.6094	0.1603	0.7937	0.2061
gpt-3.5-turbo-instruct	0.7441	0.2167	0.8377	0.2226

Table 2. Comparison of with RAG and without RAG approaches across 2 different LLMs.

From the analysis, it is clear that GPT-3.5 performs exceptionally well, outshining Davinci whether RAG is implemented and not. Furthermore, the application of RAG consistently boosts the performance of both LLMs as observed by the notable enhancements in both Mean BLEU and ROUGE scores.

Therefore, the combination of LLMs with RAG proves to be a justified and effective technique for question answering in the field of medicine. This aligns with our hypothesis that performing RAG on the input user query can aid the LLM to provide accurate, contextually relevant answers, thereby highlighting the significance of RAG.

Discussion

Interpretation and Evaluation of Results

- **LLM with RAG**: The integration of LLMs with RAG methods has proven impactful in contextualizing medical inquiries. Leveraging diverse datasets, including NFCorpus, PubMedQA, and CORD19, significantly enhanced the application's performance. This synergy exhibited a marked increase in both Mean BLEU and ROUGE scores, emphasizing the improved quality of responses, especially when informed by the NFCorpus dataset.

  The application's ability to provide tailored responses based on individual medical histories showcased a breakthrough in personalized healthcare. This not only transformed the application into a patient-centric tool but also underscored the potential of advanced AI models to revolutionize healthcare engagement.

  The inclusion of multi-modal inputs in the UI, supporting both text and speech interactions, reflected a commitment to accessibility. This design choice ensures a versatile and inclusive user experience, catering to individuals with diverse preferences and needs.

- **LLM without RAG**: In contrast, the absence of RAG in the application's architecture revealed limitations in context awareness, particularly evident in baseline performance with datasets like PubMedQA. The Mean BLEU and ROUGE scores were comparatively lower, highlighting the importance of leveraging RAG for nuanced medical queries.

  Responses without RAG tended to be more generic, lacking the context-specific tailoring observed in NFCorpus-informed queries. This underscores the pivotal role of RAG in enhancing the application's ability to generate more accurate and relevant responses.

Novel Questions Arising

- **Optimal Dataset Selection**: The variations in performance across datasets raise questions about optimal dataset selection. Understanding which datasets are most effective in different medical contexts could further refine the application's knowledge base, ensuring optimal performance.
- **Balancing Specificity and Generalization**: Achieving a balance between context-specific responses and general medical knowledge is a nuanced challenge. Future work could explore mechanisms to fine-tune this balance, ensuring that the application is robust across a spectrum of medical queries.

Limitations

- **Ethical Considerations**: The discussion on ethical considerations, especially regarding patient data privacy, highlights a limitation. Implementing local specialized LLMs is imperative, but challenges in achieving HIPAA compliance should be addressed for widespread clinical adoption.
- **Risk of Over-Specification**: The risk of hallucinations or over-specification due to the absence of a ground truth reference knowledge base necessitates further investigation. Implementing such a reference system may be crucial for preventing misinformation in medical responses.
- **Scalability and Deployment**: While the potential for integration into healthcare systems is promising, scalability and secure deployment within intra-networks present technical challenges that demand careful consideration.

Conclusion

In conclusion, this comprehensive endeavor embarked on formulating the Medical QA application as a CDQA problem, laying the foundation for a nuanced and domain-specific inquiry platform. Through meticulous research, the team identified and curated pertinent medical datasets, enhancing IR capabilities and subsequently adopting Retrieval-Augmented Generation techniques. The integration of RAG not only elevated accuracy but also facilitated contextual understanding in the realm of domain-specific Question and Answer sessions, demonstrating the application's efficacy in handling complex medical inquiries.

The implementation of an intuitive, accessible, and multi-modal UI further solidified the application's user-friendliness, ensuring seamless and conversational interactions. The strategic utilization of advanced language models, coupled with robust information retrieval mechanisms, empowered the application to deliver precise, relevant, and context-aware responses to intricate domain-specific queries.

This holistic approach showcases the potential of the Medical QA applications such as MediMate to be a valuable tool in the healthcare domain, fostering informed discussions and augmenting the accessibility of medical information for users.

Future Work

We envision several key enhancements to fortify the Medical QA application. The user interface will undergo refinement to seamlessly integrate multi-modal outputs, fostering a more interactive experience through the incorporation of both text and voice interactions. Additionally, a comprehensive chat history feature will be implemented, allowing users to revisit and build upon past interactions, thereby augmenting continuity and knowledge retention. The pursuit of greater response accuracy and heightened context awareness will involve the training of a specialized local LLM. Leveraging advanced techniques like transfer learning, this model will be fine-tuned on healthcare-specific datasets, aligning it more closely with the intricate nuances of medical language and user queries. To address scalability and accessibility, future efforts will focus on scaling and deploying the application on intra-networks, ensuring high availability within specific healthcare environments and facilitating ease of access for medical professionals. This strategic evolution reflects a commitment to advancing the application's capabilities, making it an indispensable tool in the realm of medical question answering.

References

1. Kia MA, Garifullina A, Kern M, Chamberlain J, Jameel S. Adaptable Closed-Domain Question Answering Using Contextualized CNN-Attention Models and Question Expansion. IEEE Access. 2022;10:45080–92.
2. Bernhardsson E. annoy: Approximate Nearest Neighbors in C++/Python optimized for memory usage and loading/saving to disk. [Internet]. [cited 2023 Dec 10]. Available from: https://github.com/spotify/annoy
3. Alkaissi H, McFarlane SI. Artificial Hallucinations in ChatGPT: Implications in Scientific Writing. Cureus. 15(2):e35179.
4. Altamimi I, Altamimi A, Alhumimidi AS, Altamimi A, Temsah MH. Artificial Intelligence (AI) Chatbots in Medicine: A Supplement, Not a Substitute. Cureus. 15(6):e40922.
5. Thakur N, Reimers N, Rücklé A, Srivastava A, Gurevych I. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models [Internet]. arXiv; 2021 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/2104.08663
6. Johnson J, Douze M, Jégou H. Billion-scale similarity search with GPUs [Internet]. arXiv; 2017 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/1702.08734
7. Tran P, Nguyen D, Tran HA, Nguyen T, Tran T. Building a Closed-Domain Question Answering System for a Low-Resource Language. ACM Trans Asian Low-Resour Lang Inf Process. 2023 Mar 10;22(3):82:1-82:14.
8. Wang LL, Lo K, Chandrasekhar Y, Reas R, Yang J, Eide D, et al. CORD-19: The Covid-19 Open Research Dataset. ArXiv [Internet]. 2020 Apr 22 [cited 2023 Dec 14]; Available from: https://www.semanticscholar.org/paper/CORD-19%3A-The-Covid-19-Open-Research-Dataset-Wang-Lo/4a10dffca6dcce9c570cb75aa4d76522c34a2fd4
9. Coughlin SS, Vernon M, Hatzigeorgiou C, George V. Health Literacy, Social Determinants of Health, and Disease Prevention and Control. J Environ Health Sci. 2020;6(1):3061.
10. Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, et al. Llama 2: Open Foundation and Fine-Tuned Chat Models [Internet]. arXiv; 2023 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/2307.09288
11. NFCorpus: A Full-Text Learning to Rank Dataset for Medical Information Retrieval [Internet]. StatNLP Heidelberg. 2023 [cited 2023 Dec 14]. Available from: http://www.cl.uni-heidelberg.de/statnlpgroup/nfcorpus/
12. OpenAI Platform [Internet]. [cited 2023 Dec 10]. Available from: https://platform.openai.com
13. Jin Q, Dhingra B, Liu Z, Cohen WW, Lu X. PubMedQA: A Dataset for Biomedical Research Question Answering [Internet]. arXiv; 2019 [cited 2023 Dec 14]. Available from: http://arxiv.org/abs/1909.06146
14. Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Internet]. arXiv; 2021 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/2005.11401
15. Taipalus T. Vector database management systems: Fundamental concepts, use-cases, and current challenges [Internet]. arXiv; 2023 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/2309.11322
16. Lin J, Pradeep R, Teofili T, Xian J. Vector Search with OpenAI Embeddings: Lucene Is All You Need [Internet]. arXiv; 2023 [cited 2023 Dec 10]. Available from: http://arxiv.org/abs/2308.14963