Project Report

MediBot

— AI Medical Chatbot using Retrieval-Augmented Generation (RAG)

Submitted by
SHAHBAZ NOOR (FA24-MSDS-0014)
KEEMAT RAI (FA24-MSDS-0024

Table of Contents

1. Introduction
2. Literature Review
3. System Architecture
4. Methodology
5. Implementation
6. Results and Discussion
7. Conclusion
8. Future Work
9. References

1. Introduction

With the increasing volume of medical literature, clinical documents, and research data, healthcare professionals face challenges in quickly retrieving and understanding relevant information. MediBot is designed to provide an AI-driven solution to this problem by leveraging Retrieval-Augmented Generation (RAG). This chatbot can interact with users in natural language, retrieving and generating responses based on uploaded documents. This report presents the architecture, implementation, and potential of MediBot as a modern medical information retrieval assistant.

2. Literature Review

The advancement of large language models (LLMs) such as BERT, GPT, and Mistral has transformed natural language processing. Retrieval-Augmented Generation combines retrieval of relevant documents with generative capabilities of LLMs. Langchain, HuggingFace, and FAISS are among the key technologies enabling RAG systems. Previous work on document-based QA, medical dialogue systems, and embedding-based search laid the foundation for MediBot.

3. System Architecture

MediBot consists of three main phases:
- Phase 1: Document processing and embedding into FAISS vector store.
- Phase 2: Connecting the vector store with an LLM using Langchain.
- Phase 3: Building an interactive chatbot UI using Streamlit.

The architecture includes embedding models (MiniLM), vector storage (FAISS), LLM (Mistral 7B), and a frontend built in Streamlit.

4. Methodology

The methodology involves the following steps:

1. Preprocessing documents: PDFs are chunked into smaller segments.
2. Embedding generation: Using MiniLM embeddings via HuggingFace.
3. Vector storage: Embeddings are stored and indexed using FAISS.
4. Retrieval: Relevant document chunks are fetched based on query similarity.
5. Response generation: Mistral 7B model generates responses using retrieved context and a prompt template.
6. Interaction: Users query the system through a web-based chat interface.

5. Implementation

The implementation stack includes:
- Python for backend development
- Langchain for RAG chaining
- HuggingFace API for accessing LLMs and embeddings
- FAISS for vector search and retrieval
- Streamlit for interactive UI

The `medibot.py` script integrates all components into a modular pipeline. Caching, error handling, and contextual prompts are added for robustness.

6. Results and Discussion

The system successfully retrieves context-aware responses from documents. Key strengths include:
- Domain-specific document grounding
- Response traceability
- Efficient and interactive UI

Limitations include dependency on HuggingFace API latency and potential lack of generalization to unseen document structures.

7. Conclusion

MediBot demonstrates the utility of combining vector-based retrieval with generative models for domain-specific QA. It shows strong potential as an educational and assistive tool for navigating complex medical texts. The modular design enables future upgrades such as document upload, authentication, and integration with clinical systems.

8. Future Work

Future improvements include:
- Support for uploading multiple documents
- UI-level authentication and access control
- Offline inference with quantized models
- Unit testing of the complete pipeline
- Integration with secure clinical data repositories

9. References

- Devlin et al., BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- Facebook AI Research, FAISS: Efficient Similarity Search

- HuggingFace Documentation: Transformers, Datasets, and Inference API
- Langchain Documentation
- Mistral AI: Mistral-7B Model Overview