Technical Report: Development of a Medical QA System

1. Introduction

This project aimed to develop an advanced **Medical Question Answering (QA)** system capable of providing preliminary diagnostic assessments. The system focuses on three key medical departments: Cardiology, Neurology, and Gastroenterology.

The primary goals were to:

1. Integrate multiple data sources for comprehensive information retrieval.

2. Implement an efficient vector search mechanism for quick and accurate information retrieval.

3. Utilize a large language model (LLM) for generating coherent and contextually appropriate diagnostic assessments.

4. Create an agentic system that can handle complex medical queries effectively.

2. Methodology

2.1 Data Preparation

The system utilizes multiple data sources to ensure comprehensive coverage:

1. **Custom Medical Dataset:** A JSON file (`dev.json`) containing medical questions and answers.
2. **Wikipedia**: For general medical information retrieval.
3. **PubMed Central (PMC) Open Access Subset**: For accessing recent biomedical research articles.
4. The custom dataset undergoes preprocessing:
   - Text cleaning (removing special characters, lowercasing)
   - Sentence tokenization
   - Stopword removal

2.2 Retrieval System

The retrieval system employs multiple techniques:

1. **FAISS (Facebook AI Similarity Search)**:
   - Sentence embeddings are generated using the `SentenceTransformer` model 'all-MiniLM-L6-v2'.
   - A FAISS index is created for efficient similarity search.
   - The index and processed documents are pickled for faster loading in subsequent runs.

2. **Wikipedia Retriever**:
   - Utilizes the `WikipediaRetriever` from LangChain to search for relevant Wikipedia articles.

3. **PMC Open Access Search**:
   - Implements a custom function to search the PMC Open Access Subset using the NCBI E-utilities API.

2.3 LLM Integration

The system uses the Google Generative AI model "gemini-1.5-pro" for:
1. Generating diagnostic assessments
2. Synthesizing information from multiple sources

2.4 Agentic System

An agentic system is implemented using a `Crew` class, which manages multiple `Agent`s and `Task`s:

1. **Medical Information Researcher**: Searches Wikipedia for general medical information.
2. **Medical Data Specialist**: Utilizes the FAISS index to search the custom medical dataset.
3. **PMC Open Access Specialist**: Searches for relevant biomedical research articles.
4. **Diagnostic Specialist:** Provides preliminary diagnostic insights based on the collected information.

3-Logic and Approach:
System Architecture
Components:

- Query Processor: Accepts and formats user queries.
- Agent Ensemble: A collection of specialized AI agents.
- Corpus Aggregator: Combines agent outputs into a unified corpus.
- Response Synthesizer: Generates the final intelligent response.

Workflow

Query Input and Processing

The system accepts a medical query, which may include a text description and optional lab report data. The query is formatted into a standardized MedicalQuery object.

Multi-Agent Information Retrieval

The query is simultaneously sent to multiple specialized agents:

a. Wikipedia Agent: Retrieves general medical information from Wikipedia.
b. FAISS QA Agent: Searches a pre-indexed custom medical dataset using vector similarity.
c. PMC Open Access Agent: Fetches recent biomedical research articles from PubMed Central.
d. Diagnostic Specialist Agent: Provides preliminary diagnostic insights based on the query.

PS-:Each agent processes the query independently, leveraging its specific data source and expertise.

Corpus Creation

- The system collects responses from all agents.
- These diverse pieces of information are combined into a single, comprehensive corpus.
- The corpus represents a multi-faceted view of the medical query, incorporating general knowledge, specific medical data, recent research, and preliminary diagnostic insights.

Intelligent Response Generation

- The aggregated corpus, along with the original query, is sent to a large language model (Google's Gemini 1.5 Pro).
- The LLM acts as an "intelligent synthesizer," analyzing the corpus in the context of the query.
- It generates a final response that integrates and contextualizes the information from all sources.

Output Delivery

The system presents the synthesized response to the user, providing a comprehensive and intelligent answer to the original medical query.

Key Aspects of the Approach

- **Modularity**: The multi-agent design allows for easy updates or replacements of individual components without affecting the overall system.
- **Diverse Information Sources**: By leveraging multiple data sources, the system ensures a well-rounded and up-to-date knowledge base.
- **Efficient Retrieval**: The use of FAISS for vector search enables quick and accurate retrieval from large datasets.
- **No Fine-Tuning Required:** The system relies on pre-trained models and specialized retrieval methods, eliminating the need for resource-intensive fine-tuning.
- **Contextual Synthesis**: The final LLM-based synthesis step ensures that the diverse information is coherently integrated and contextualized for the specific query.
- **Scalability**: New agents or data sources can be easily added to expand the system's capabilities.

4. Results

The system's performance was evaluated based on its ability to:
1. Retrieve relevant information from multiple sources
2. Generate coherent and contextually appropriate diagnostic assessments

Qualitative assessment of the system's output shows:
- Comprehensive information gathering from diverse sources
- Contextually relevant diagnostic insights
- Clear presentation of potential diagnoses, symptoms, and suggested diagnostic procedures

Quantitative metrics were not provided in the given code, but future iterations could include:
Precision and recall for information retrieval
BLEU or ROUGE scores for generated diagnostic assessments
Expert evaluation of the system's diagnostic accuracy

5. Challenges and Solutions

1. **Data Integration:**
   - Challenge: Integrating diverse data sources with different formats and access methods.
   - Solution: Implemented custom functions for each data source and standardized the output format.

2. **Efficient Information Retrieval:**
   - Challenge: Quickly searching through large datasets for relevant information.
   - Solution: Utilized FAISS for vector similarity search, enabling fast and accurate retrieval.

3. **Context Maintenance**:
   - Challenge: Maintaining context across multiple information sources and queries.
   - Solution: Implemented an agentic system with specialized roles and a final synthesis step.

4. **Error Handling**:
   - Challenge: Dealing with potential API failures or data inconsistencies.
   - Solution: Implemented robust error handling and provided fallback options where possible.

5-**Time**:
  -It is taking time as we are using lots of databases.

6. Conclusion

The developed Medical QA System demonstrates the potential of integrating multiple data sources, efficient vector search, and large language models for preliminary medical diagnostics. The agentic approach allows for specialized handling of different aspects of the diagnostic process.

Future improvements could include:
1. Expanding the system's knowledge to cover more medical specialties
2. Implementing a more sophisticated dialogue system for follow-up questions
3. Integrating a medical knowledge graph for improved reasoning capabilities
4. Conducting extensive testing with medical professionals to validate and refine the system's outputs
5. Make it faster and feasible to our users.

7-Links

- Images-Output-{https://drive.google.com/drive/folders/1lDeH6rNnl8YQDtd7wmhdh-YVcC87aWki?usp=sharing}
- Code-{Github}-{https://github.com/Aryamantiwari17/Med_Mitra}