Research Report on MedicalGPT

1 Introduction

Introducing MedicalGPT, the advanced medical chatbot that offers interactive conversations to provide precise, evidence-based answers to medical queries. Specializing in medical-related questions, MedicalGPT ensures accuracy and reliability by drawing from vast datasets and trusted sources. With the ability to analyze documents, PDFs, and website links, it caters to various information formats, empowering users to make informed health decisions. Rest assured, your privacy is a top priority as MedicalGPT keeps all interactions confidential. Embrace the future of medical information retrieval and experience the convenience of personalized medical assistance with MedicalGPT today.

2 Features

2.1 Interactive chatting with users

The MedicalGPT chatbot is designed to engage in interactive and natural conversations with users. Through its advanced language processing capabilities, it can understand and respond to a wide range of medical-related queries. The chatbot employs context-awareness, allowing it to remember past interactions and maintain continuity in conversations. It adapts its responses based on the user's input, ensuring a personalized and user-friendly experience.

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

2.2 Provide factual answers to only medical-related queries

MedicalGPT is designed with a strict focus on providing accurate and evidence-based information exclusively for medical-related queries. To achieve this, the chatbot is integrated with up-to-date medical databases, reputable medical literature, and verified sources of medical knowledge.

When a user poses a medical question, the chatbot uses its natural language processing (NLP) capabilities to understand the context and intent behind the query. It then searches its vast database of medical information to retrieve relevant and reliable answers. The chatbot's responses are filtered to ensure they only present factual and scientifically supported information, minimizing the risk of spreading misinformation.

2.3 Answer questions from documents: MedicalGPT extends its utility beyond simple text-based queries by accommodating questions from different file formats. The chatbot can handle documents in popular formats like DOC, DOCX, and PDF. Users can upload these files, and the chatbot will process the content to extract relevant medical information.

For documents in DOC, DOCX, and PDF formats, the chatbot utilizes Optical Character Recognition (OCR) technology. OCR allows the chatbot to convert scanned documents or images of text into machine-readable text, making it possible to analyze and respond to queries based on the content within these documents.

3 Technical Components

The system comprises three core components: LLM (Language Model) for question-answering, a Knowledge Base serving as a vector database and data retriever, and a Framework that provides essential tools for building the entire pipeline. The LLM employs advanced language processing to understand user queries and generate precise responses, while the Knowledge Base stores vast information related to the subject domain and uses semantic searching to retrieve relevant data. The Framework ensures seamless integration of these components, facilitating the interactive and accurate functioning of the chatbot system.

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

3.1 Large Language Model (LLM)

For question answering, in our case chatting, we employed two techniques:

1. OpenAI API
2. Open source LLM's

Open source LLM's we have employed so far are given in the table.

Model name	Parameters	Type
falcon-7b-instruct	7b	fine-tuned
alpaca-13b	13b	base
mpt-7b-chat	7b	fine-tuned
llama-2–7b-chat	7b	fine-tuned
medalpaca-7b	7b	fine-tuned
medalpaca-13b	13b	fine-tuned

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

LoRA

We have used Low-Rank Adaptation technique for loading these Large Language Models. For inferring, we have loaded the models in 8-bit and 4-bit so that we can load our model using our VRAM.

3.2 Knowledge Base

A knowledge base is a method where we create a dataset with information about our subject area. We store this data in a vector format. When a user asks a question, we use semantic searching to find the most suitable answer from the dataset. This answer is then given to the language model (LLM) as context. The LLM uses this context to respond to the user's questions.

This is done using two steps:

1. Information storing
2. Information retrieval using semantic searching

3.2.1 Information storing

For information storing we have used two databases.

1. **Faiss index :** Facebook AI Similarity Search (Faiss) is a library for efficient similarity search and clustering of dense vectors. It contains algorithms that search in sets of vectors of any size, up to ones that possibly do not fit in RAM. It also contains supporting code for evaluation and parameter tuning. Uses a db file to store data, and we can retrieve data by semantic searching. Easy to use.

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

2. **Elastic search:** Elasticsearch is a distributed, RESTful search and analytics engine. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elastic search stores data in the cloud and we can retrieve data through REST API's.

We have employed both data stores, and compared the performance.

Store name	**Stores in**	**Visualization**	**Retrieval speed**
Faiss index	Local directory	Locally/ easy	Fast
Elastic search	Cloud	Through REST API/ complex	Comparatively slow

Comparatively, faiss index is more convenient for our chatbot. So, we have used the faiss index to store our data.

3.2.2 Information retrieval

For information retrieval we have used embedding models/ sentence transformers. Using those models, we do semantic searching and retrieve the most relevant result for the user query. We have tried out these models and compared performance.

Model name	Cost	Model Size (GB)	Performance (correct retrieval out of 10 queries)	Model load time (s)	Time needed to fetch output (s)	Memory needed for loading model (GB)
OpenAI embedding	$0.0001 / 1K tokens	API	9	-	0.4	-
all-MiniLM-L6-v2	free	0.09	2	0.14	0.85	1.2
all-mpnet-base-v2	free	0.44	4	0.48	0.87	1.8
instructor-xl	free	4.96	8	13.58	1.49	10.5
instructor-large	free	1.34	5	1.44	0.98	2.2
e5-large-v2	free	1.34	7	1.37	1.00	2.00

a: House 57, Road 04, Block C, Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

We had to make a tradeoff between different models and found out that the `**e5-large-v2**` model is most suitable for our task. Performance is good and model size is not very large. Almost all models have very little inference time, so inference time was not a concern to us. So, the tradeoff was done on accuracy and model size.

3.3 Framework

1. Langchain
2. Llama-index
3. Customized

We have used the Langchain framework for building our chatbot. LangChain is a framework for developing applications powered by language models. It provides wide ranges of features including Document stores and retrievers, LLM chains, Chatbot Agents and so on. Also, document storing is customizable (can use open source sentence transformer models).

Llama index also provides features such as storing and retrieval and use of LLMs, but its features are limited. Here is a comparison between Langchain and LlamaIndex.

Langchain	LlamaIndex
General-purpose	Indexing and retrieval
Text splitting, chain management, agent integration	Graph indexes

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

Applications that require a wide range of features	Applications that require only efficient indexing and retrieval

As Langchain is providing a wide range of features (document store and retrieval, chat models, llm agents, chains etc.), we have used langchain to build our chatbot.

4 Document Processors

When storing data for knowledge base, our chatbot accepts docs (docs, doc, .docx), pdfs, images, csv's, web links and text (.txt) files. We had to create pipelines for storing data into our knowledge base for all kinds of files.

DOCS & TEXTS

For doc files, we have used the Langchain Document module. It supports (.docs, .docx, .doc, .txt) formats and stores data into the vector database.

CSV

For csv files, we have used the Langchain CSVLoader module. It can read the csv file rows and store them in a vector database.

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc

PDF

There are two types of pdf, one type needs OCR and the other does not. We have two pipelines for both the types.

1. Without OCR

    - PyPDFLoader: Langchain offers a PyPDFLoader module which can store data which doesn't need OCR. But texts cannot be extracted from pdfs that need OCR using this module.

2. With OCR

    - Google Vision API [paid]
    - Haystack PDFToTextOCRConverter module [pytesseract need to be installed in computer]

a: House 57, Road 04, Block C,
Banani, Dhaka
p: +8801747810786
m: info@anchorblock.vc
w: www.anchorblock.vc