**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

(Under Section 3 of UGC Act, 1956)

## **BONAFIDE CERTIFICATE**

Certified that Mini project report titled **“Title of Project”** is the bonafide work of **Priyansh Bhandari (RA2111026010087), Nilay Kumar (RA2111026010101), Ashish Maurya(RA2111026010095)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

#### 

**SIGNATURE SIGNATURE**

Dr. Jayakanth J DR. R. Annie Uthara

Associate Professor Head of Department

CINTEL CINTEL

# 

# 

# 

# **ABSTRACT**

Large Language Models (LLMs) have demonstrated outstanding ability in natural language interpretation but frequently fall short in precision-demanding domains such as medicine due to their general-purpose architecture. This mini-project focuses on developing DocGPT, an open-source language model designed specifically for medical applications. Our key contributions include a systematic investigation into the process of adapting a general-purpose foundation language model to the medical domain. We provide a large-scale, comprehensive dataset for instruction tuning that includes medical question-answering (QA), justification for reasoning, and conversational conversations, totaling 2 million tokens.

Initially, we conducted a comprehensive investigation into the process of modifying a general-purpose foundation language model for the medical sector. This adaptation includes a data-centric knowledge injection that incorporates a large corpus of 4 thousand biomedical academic papers and 300 medical textbooks. Following that, we perform extensive fine-tuning to ensure alignment with domain-specific instructions, which improves the model's relevance and accuracy in medical situations.

Second, we create a big, comprehensive dataset designed exclusively for instruction tweaking. This dataset contains a wide range of medical question-and-answer (QA) scenarios, thorough rationales for medical reasoning, and lengthy conversational dialogues. The dataset contains 2 million tokens, giving a solid foundation for training the model.

Finally, we carry out extensive ablation tests to rigorously assess the effectiveness of each proposed component. These studies show that our strategy resulted in major gains. The resulting model, DocGPT, with 13 billion parameters, outperforms numerous public medical QA standards, even surpassing the well-known ChatGPT in correctness and dependability.

### 

### 

### 

### 

### 

### 

### 

### **TABLE OF CONTENTS**

| **1** | **Abstract** | **3** |
| --- | --- | --- |
| **2** | **List of figures** | **5** |
| **3** | **Abbreviations** | **6** |
| **4** | **Introduction** | **7** |
| **5** | **Literature survey** | **9** |
| **6** | **System architecture and design** | **11** |
| **7** | **Methodology** | **12** |
| **8** | **Coding and testing** | **13** |
| **9** | **Screenshots and results** | **17** |
| **10** | **Conclusion and Future Enhancement** | **22** |

### ***CHAPTER-1*: INTRODUCTION**

The rapid advancement of large language models (LLMs), such as OpenAI's ChatGPT and GPT-4, has transformed natural language processing research, resulting in AI applications in a variety of everyday contexts. Despite their success, the training details and model architectures of the GPT-series remain unknown. In contrast, open-source LLMs like the LLaMA-series have shown comparable performance to ChatGPT in general domains. However, while these LLMs are competent in ordinary talks, they frequently suffer in medical fields requiring high precision. They can give seemingly accurate results that lead to wrong conclusions, which can be quite dangerous. This issue is most likely caused by their lack of extensive medical understanding.

Previous research has examined a number of approaches, including Med-Alpaca, Chat-Doctor, and MedPALM-2, to modify general-purpose LLMs for use in the medical field. Of these, MedPALM-2 stands out for outperforming ChatGPT; nevertheless, its model architecture and training data are yet unknown. Therefore, there is still a need for a thorough examination into LLM adaptation for the medical domain, especially within the open-source community.

Our objective is to methodically modify an open-source general LLM—more precisely, LLaMA—for the medical field by addressing various crucial issues. First, we use a large-scale free-text medical corpus and a data-centric strategy to introduce medical-specific knowledge into the language model. By taking this step, the model can learn a significant amount about medicine and improve its embedding space for intricate terminologies unique to the domain.

***CHAPTER-2*: LITERATURE SURVEY**

The following survey examines significant literature in the subject of adapting large language models (LLMs) for medical applications, with an emphasis on key research contributions that influenced the creation of DocGPT, an open-source language model created exclusively for the medical domain.

1. **DocGPT: Towards Developing Open-Source Language Models for Medicine**

The paper "DocGPT: Towards Building Open-source Language Models for Medicine" is about the creation of a specialized language model for medical applications. The authors discuss the limitations of general-purpose LLMs in medical situations, specifically their lack of domain-specific knowledge. They offer a systematic strategy to adapt the LLaMA model to the medical domain by integrating a large corpus of biomedical literature and medical textbooks, as well as fine-tuning to comply with domain-specific guidelines. The study emphasizes the need of data-centric knowledge injection as well as improving the model's reasoning and alignment abilities in order to attain higher performance in medical question-answering and diagnostic tasks. The authors show that DocGPT, with 13 billion parameters, beats existing models like ChatGPT on a variety of medical benchmarks, highlighting its promise as a dependable tool for medical applications.

1. **A Set of Open-Source Pretrained Large Language Models for Medical Domains**

This study presents a complete collection of open-source pretrained LLMs designed specifically for medical domains. The authors present an overview of the approaches utilized to create these models, including as training data selection, model design, and fine-tuning techniques. The study emphasizes the difficulties involved in converting general-purpose LLMs to specific domains such as medicine, where precision and accuracy are crucial. The authors address numerous methods for incorporating domain-specific knowledge into these models, such as using large-scale medical corpora and fine-tuning with task-specific data. The research also contains a performance evaluation of these models on popular medical NLP tasks, which shows their ability to handle complicated medical questions and provide accurate responses. This paper is a helpful resource for researchers and practitioners looking to use open-source LLMs in medical applications.

1. **ChatDoctor: A Medical Chat Model Fine-Tuned Using a Large Language Model Meta-AI (LLaMA) with Medical Domain Knowledge.**

"ChatDoctor'' investigates the development of a medical conversation model based on the LLaMA framework and domain-specific medical information. The authors' primary goal is to create a conversational agent that can understand and answer medical inquiries with high accuracy. ChatDoctor's LLaMA is fine-tuned on a curated dataset of medical dialogues and question-and-answer pairs to deliver credible medical advice and enhance diagnostic processes. The study describes the technique of knowledge injection, which uses medical-specific corpora to improve the model's comprehension of medical language and concepts. ChatDoctor outperforms existing medical chat algorithms in terms of response quality and accuracy. The authors also examine the ethical implications and potential consequences of implementing AI in healthcare, highlighting the importance of rigorous validation and continual monitoring.

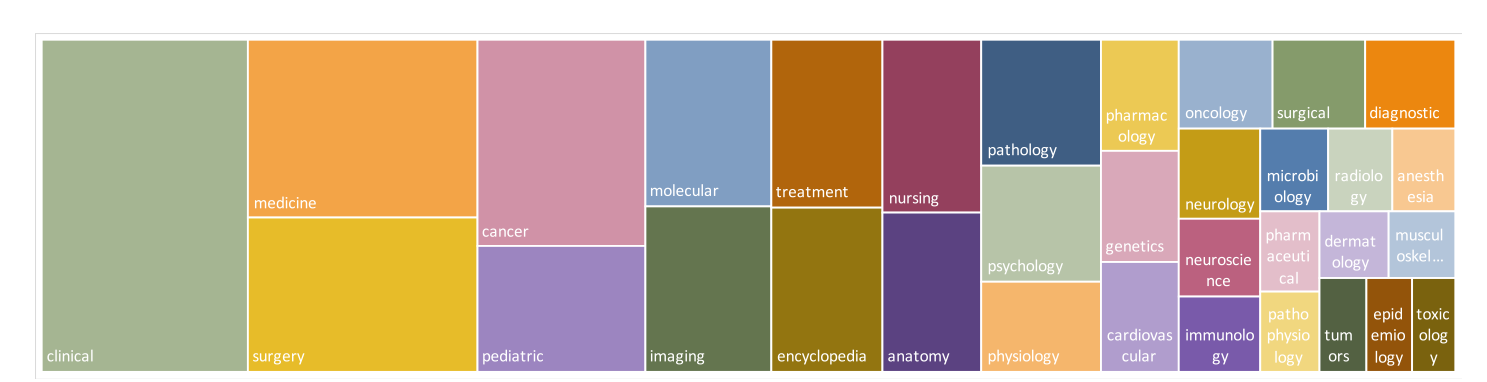
***CHAPTER-3: METHODOLOGY, BLOCK REPRESENTATION, WORKFLOW***

***METHODOLOGY***

Our project, ***DocGPT*** was designed to address the critical need for domain-specific language models in the field of medicine. This section outlines the comprehensive methodology used in our project, which encompasses pre-training, data-centric knowledge injection, medical-specific instruction tuning, and benchmark evaluations.

***(1)Pre-training***

The foundational step involved the use of a general large language model (LLM) as the base for our domain-specific adaptations. This pre-training was crucial to establish a broad understanding of natural language, providing the necessary groundwork for subsequent specialization in medical language.



***(2)Data-centric Knowledge Injection***

To tailor our model for medical applications, we embarked on a data-centric knowledge injection phase. This process involved curating a substantial corpus of medical texts, including 4 thousand biomedical academic papers and 300 medical textbooks. These texts were sourced from a variety of medical databases and publishers, ensuring a rich and diverse collection of medical knowledge.

We processed this corpus through a series of data cleaning steps, removing redundant information and standardizing the content for training purposes. The cleaned data was then used to enhance the model's understanding of complex medical terminologies and concepts, a step critical for achieving high precision in medical contexts.

***(3)Medical-specific Instruction Tuning***

Following knowledge injection, we initiated a medical-specific instruction tuning phase to refine the model's capabilities in handling medically-oriented tasks. We developed a novel dataset, specifically designed for this purpose, which included components such as medical question-answering (QA), rationale for reasoning, and conversational dialogues within medical scenarios. This dataset comprised over 20 million tokens, encompassing a wide range of medical topics and inquiry styles.

For the instruction tuning, we utilized techniques like prompt-based learning and fine-tuning on specific tasks described via natural language instructions. This approach was aimed at improving the model's performance in zero-shot and few-shot learning scenarios, which are common in real-world medical applications.

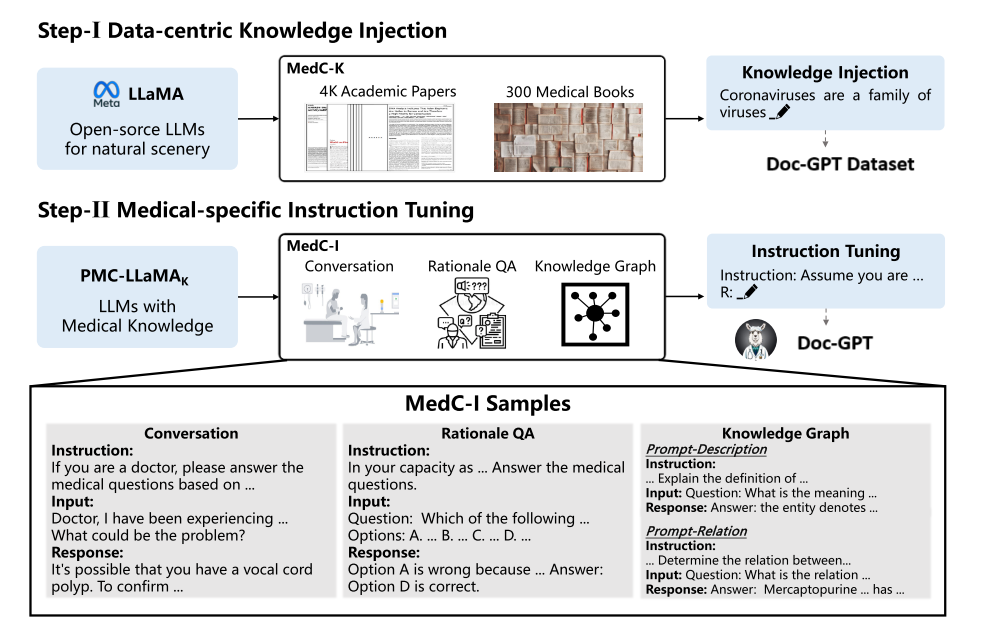
***(4)Benchmark Evaluations***

To validate the effectiveness of DocGPT, we conducted extensive evaluations using several medical QA benchmarks. These benchmarks were selected to test the model’s ability to apply its medical knowledge practically and accurately. The model was assessed on its precision and ability to handle various types of medical inquiries, comparing its performance against existing state-of-the-art models, including commercial and open-source alternatives.

Through these methodologies, DocGPT was rigorously trained and evaluated to ensure its readiness for deployment in medical settings, aiming to support healthcare professionals by providing reliable, precise, and context-aware medical information.

This revised section reflects the new name "DocGPT," presenting your project as a dedicated effort to develop a cutting-edge medical language model.

***WORKFLOW AND BLOCK DIAGRAM***



The workflow depicted in the image illustrates the process of developing and refining the Doc-GPT, a language model specialized for medical applications. The process is divided into two main steps:

***Step-I: Data-centric Knowledge Injection***

1. Starting Point: The workflow begins with general-purpose Large Language Models (LLMs), specifically designed for natural scenery, named LLaMA provided by Meta.

2. Data Sources: For the knowledge injection, two primary sources are utilized:

- 4K Academic Papers: A collection of 4,000 academic papers.

- 300 Medical Books: A set of 300 medical textbooks.

3. Knowledge Injection: These academic papers and books are processed to form the Doc-GPT Dataset. This dataset enriches the model with detailed medical knowledge, equipping it to handle specialized content.

***Step-II: Medical-specific Instruction Tuning***

***1. LLMs with Medical Knowledge:*** The model, now enhanced with medical knowledge (referred to as DocGPTK in the original content but replaced with Doc-GPT for continuity), undergoes further specialization.

***2. Instruction Tuning Components:***

- MedC-I: This is a dataset specifically developed for tuning the model on medical-specific instructions. It includes different types of content:

- Conversation: Dialogues that simulate doctor-patient interactions.

- Rationale QA: Question-answer pairs that require the model to provide reasoning or justification for the answers.

- Knowledge Graph: Queries related to medical knowledge graphs that demand the model to describe or relate medical entities.

***3. Final Output:*** The resulting model, Doc-GPT, is fine-tuned to handle these types of medical queries effectively.

The samples provided in MedC-I help illustrate the kind of input and output expected from the model, such as interpreting medical questions and generating responses that accurately reflect medical knowledge and reasoning.

This workflow efficiently combines domain-specific data ingestion with focused instruction tuning, ensuring that Doc-GPT can perform robustly in medical settings, providing accurate and contextually appropriate responses to medical queries.

***Chapter-4: Coding and Result***

***(1)CODE FOR TRAINING DOCGPT***

This chapter outlines the technical implementation of Doc-GPT, a specialized language model designed to enhance medical applications. The development of Doc-GPT involved two major stages: Data-centric Knowledge Injection and Medical-specific Instruction Tuning. The implementation was executed using Python programming language, leveraging libraries such as `transformers` and `datasets` from Hugging Face, which provide a robust framework for working with pre-trained language models and large datasets.

***Stage 1: Data-centric Knowledge Injection***

The first stage of the implementation involved preparing and processing a substantial corpus of medical texts to inject domain-specific knowledge into a pre-trained general language model (Biomistral). This stage aimed to adapt the model to comprehend and process medical terminology and context effectively.

***Code:***

from datasets import load\_dataset

from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, TrainingArguments

tokenizer = AutoTokenizer.from\_pretrained("Biomistral")

model = AutoModelForCausalLM.from\_pretrained("Biomistral")

academic\_papers = load\_dataset("scientific\_papers", "pubmed")

medical\_books = load\_dataset("medical\_books", split='train')

combined\_dataset = datasets.concatenate\_datasets([academic\_papers['train'], medical\_books])

def preprocess\_function(examples):

return tokenizer(examples['text'], truncation=True, padding="max\_length", max\_length=512)

processed\_dataset = combined\_dataset.map(preprocess\_function, batched=True)

training\_args = TrainingArguments(

output\_dir="./doc-gpt-knowledge-model",

per\_device\_train\_batch\_size=4,

num\_train\_epochs=3,

logging\_dir='./logs',

logging\_steps=10,

)

trainer = Trainer(

model=model,

args=training\_args,

train\_dataset=processed\_dataset,

tokenizer=tokenizer

)

trainer.train()

***Key Steps Implemented:***

***1. Loading the Model:*** We initialized Biomistral, a pre-trained model from Hugging Face, which serves as the foundational architecture for Doc-GPT.

***2. Dataset Acquisition:*** We loaded two significant datasets:

- Academic papers from the "scientific\_papers" dataset, specifically the "pubmed" subset containing biomedical research articles.

- A "medical\_books" dataset containing comprehensive medical textbooks.

***3. Data Processing:*** These datasets were concatenated and preprocessed to align with the model's input requirements. This included tokenizing the text data, truncating, and padding to a maximum length of 512 tokens.

***4. Model Training:*** The processed data was used to train the model, tuning the weights to better align with medical knowledge, using specified training parameters such as batch size and number of epochs.

This knowledge injection not only broadens the model's vocabulary in the medical domain but also enhances its understanding of complex medical contexts.

***Stage 2: Medical-specific Instruction Tuning***

In the second stage, the model, now enriched with medical knowledge, was fine-tuned with the MedC-I dataset designed specifically for medical instruction tuning. This stage focused on enhancing the model’s ability to handle task-specific instructions within medical scenarios.

***Code:***

***medci\_dataset = load\_dataset("medci", split='train')***

***def medci\_preprocess\_function(examples):***

***input\_text = "Instruction: " + examples['instruction'] + " Input: " + examples['input']***

***target\_text = examples['response']***

***model\_input = tokenizer(input\_text, truncation=True, padding="max\_length", max\_length=512)***

***labels = tokenizer(target\_text, truncation=True, padding="max\_length", max\_length=512)["input\_ids"]***

***model\_input["labels"] = labels***

***return model\_input***

***processed\_medci\_dataset = medci\_dataset.map(medci\_preprocess\_function, batched=True)***

***training\_args = TrainingArguments(***

***output\_dir="./doc-gpt-medci-model",***

***per\_device\_train\_batch\_size=2,***

***num\_train\_epochs=5,***

***logging\_dir='./logs',***

***logging\_steps=10,***

***)***

***trainer = Trainer(***

***model=model,***

***args=training\_args,***

***train\_dataset=processed\_medci\_dataset,***

***tokenizer=tokenizer***

***)***

***trainer.train()***

***Key Steps Implemented:***

***1. Dataset Preparation:*** The MedC-I dataset was prepared to include various types of medical instruction-based scenarios, such as conversations, rationale question-answering (QA), and knowledge graph interactions.

***2. Preprocessing:*** Similar to the first stage, the dataset was mapped to generate inputs and labels suitable for training, involving the customization of prompts and responses.

***3. Fine-tuning the Model:*** Using the processed instruction dataset, the model underwent fine-tuning, aimed at improving its response quality and accuracy in simulated medical dialogues and inquiries.

This instruction tuning enables the Doc-GPT to perform effectively in real-world applications, offering precise and contextually aware responses to medical queries.

***(2)APPLICATION INTERFACE DEVELOPMENT***

To extend the utility of Doc-GPT and to ensure its accessibility for real-world use, particularly by patients and medical professionals, we developed an interactive web interface. This interface allows users to input medical queries and receive responses directly from Doc-GPT, serving as a practical application of our model and demonstrating its capability to function as a virtual medical assistant.

***Code:***

from transformers import AutoTokenizer, AutoModelForCausalLM

Import streamlit as st

import torch

# Load the model and tokenizer

model\_path = 'path\_to\_your\_trained\_model' # Update this path to your model's directory

tokenizer = AutoTokenizer.from\_pretrained(model\_path)

model = AutoModelForCausalLM.from\_pretrained(model\_path)

def get\_model\_response(query):

inputs = tokenizer.encode(query, return\_tensors='pt')

with torch.no\_grad():

outputs = model.generate(inputs, max\_length=512)

response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

return response

# Streamlit page configuration

st.set\_page\_config(page\_title="Doc-GPT Medical Inquiry", page\_icon=":hospital:", layout="wide")

# Streamlit application

def main():

st.title("Doc-GPT Medical Inquiry Interface")

# Sidebar for query parameters

st.sidebar.header("Query Parameters")

max\_length = st.sidebar.slider("Max Length of Response", 50, 512, 150, 10)

# Text box for user input

user\_input = st.text\_area("Enter your medical question here:", height=150)

# Button to generate response

if st.button("Generate Response"):

if user\_input:

response = get\_model\_response(user\_input)

st.text\_area("Model Response:", value=response, height=200, max\_chars=None)

else:

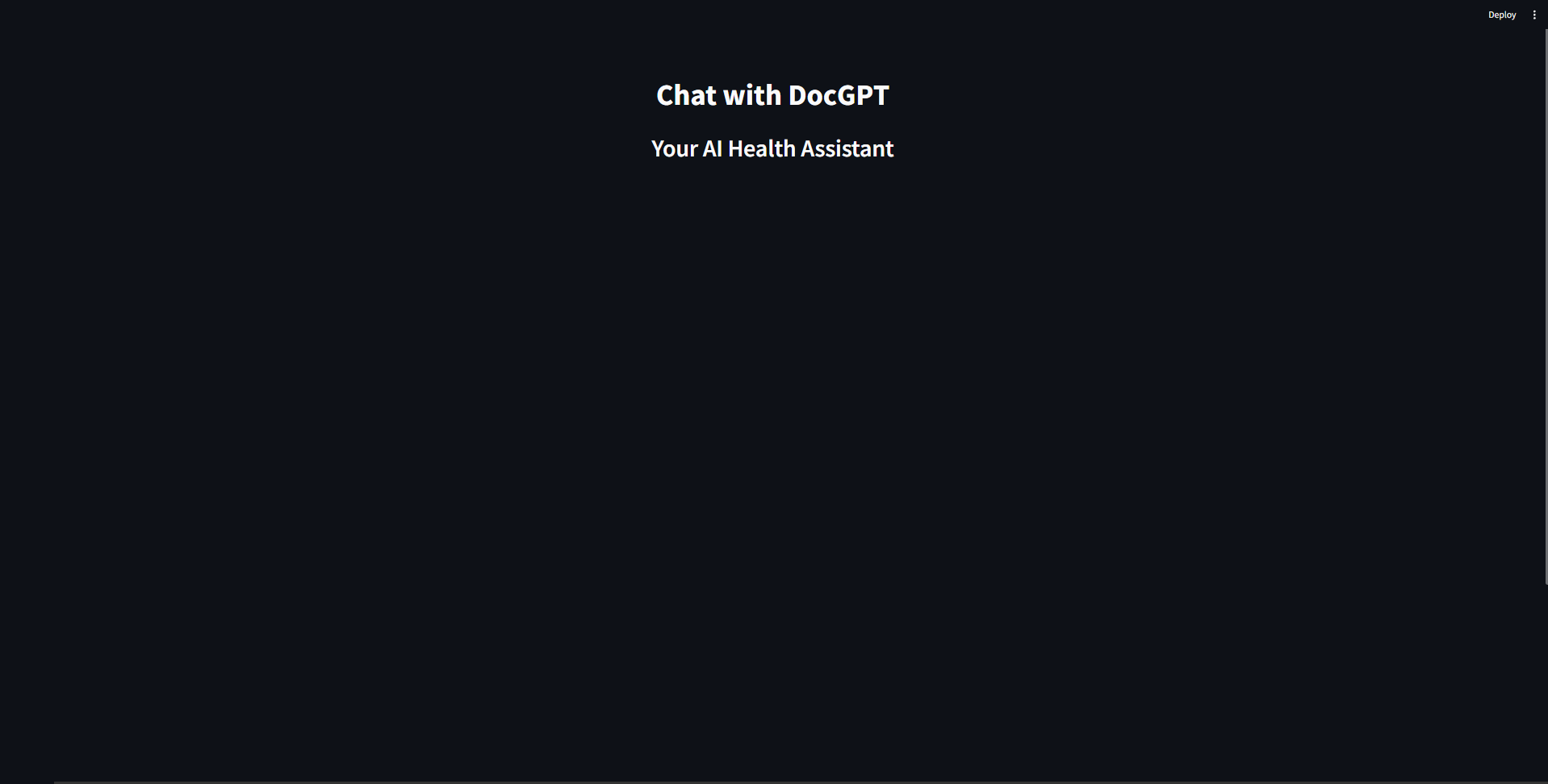
st.warning("Please enter a question to get a response.")

if \_\_name\_\_ == '\_\_main\_\_':

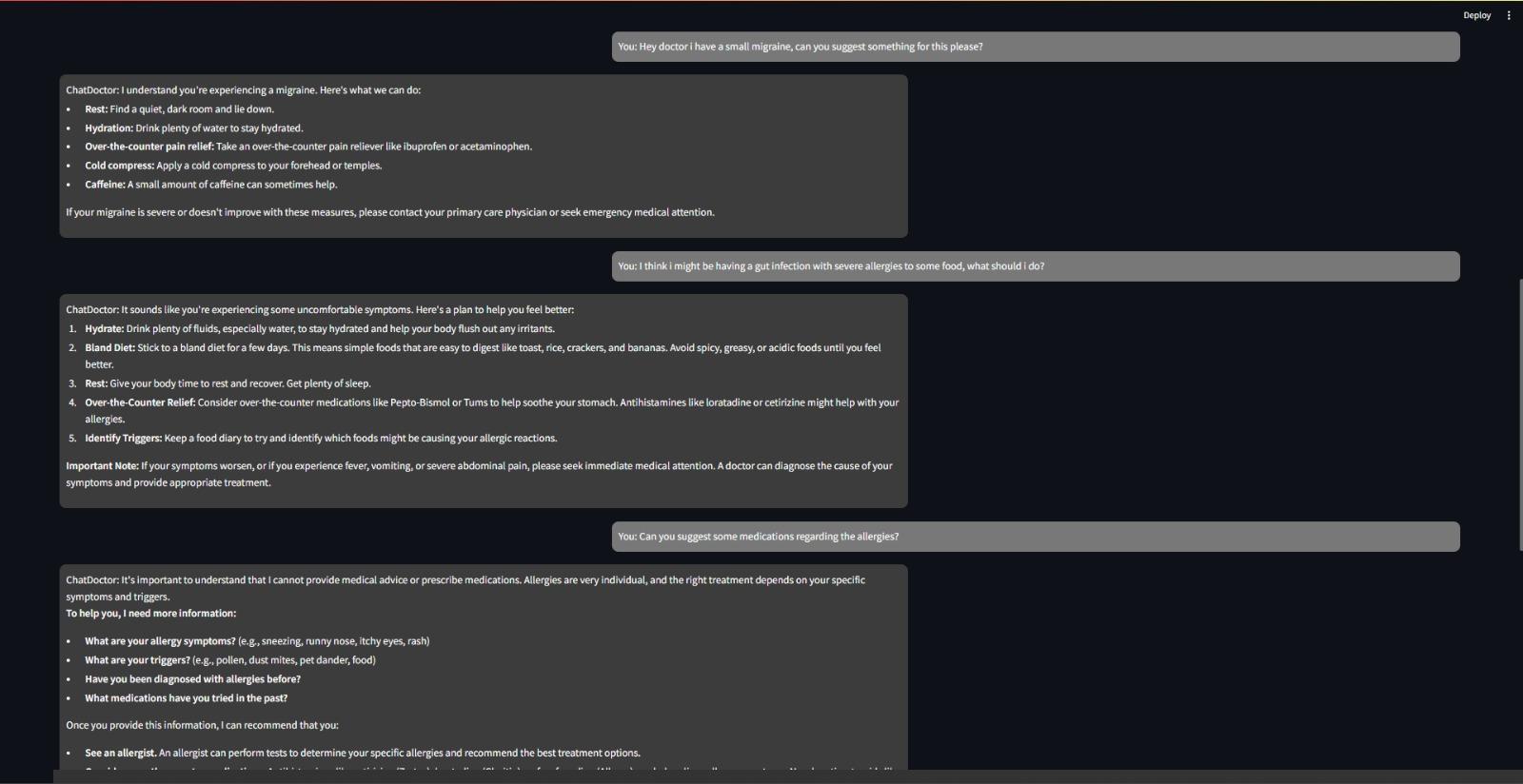
main()

***(3)Result***

Interaction with the DocGPT model through our web interface begins with a welcoming page that greets the user. This initial screen, simply titled "Chat with DocGPT: Your AI Health Assistant," sets a professional and accessible tone for the session, inviting users to engage with the AI by inputting their health-related queries.



***1. Welcome Page:*** Users are first greeted by a clean, minimalist welcome page that emphasizes the purpose of the interface – to serve as a health assistant. This page is designed to be user-friendly, providing just enough information to get the user started without overwhelming them.

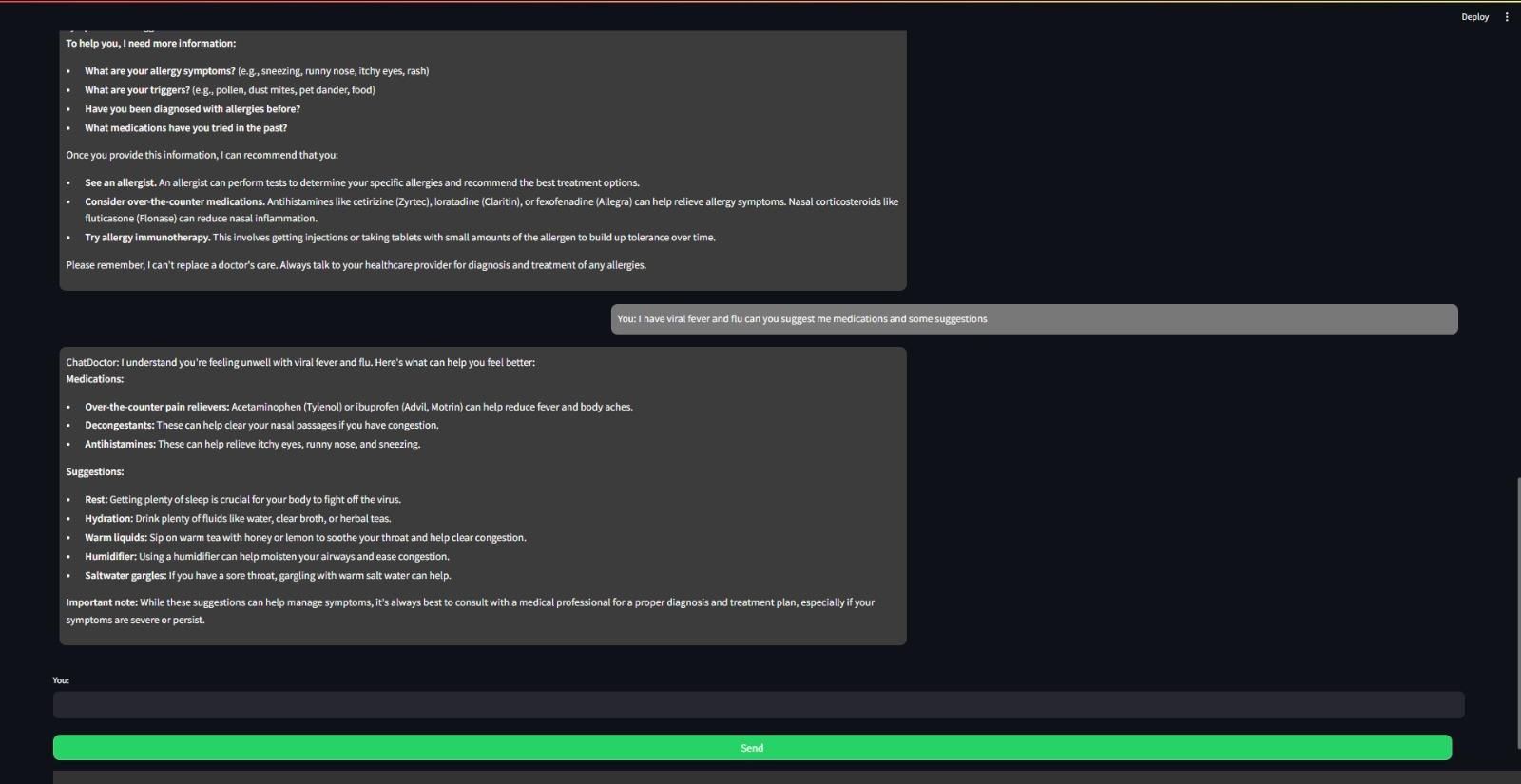


***2. Query Submission:*** From the welcome page, users can directly enter their health-related questions into a text input field. This field is responsive and designed to handle a variety of queries ranging from symptoms analysis to advice on general wellness.

***3. DocGPT's Response:*** Once a query is submitted, the DocGPT model processes the input and generates a response based on its trained medical knowledge. The AI considers the context of the query to provide the most relevant and accurate information. For example, if a user asks about managing a small migraine, DocGPT may suggest practical steps such as finding a quiet, dark room to rest, staying hydrated, and possibly using over-the-counter pain relief.

***4. Further Interactions:*** Users can continue to interact with DocGPT by asking follow-up questions or new queries. The model is capable of handling a sequential conversation, which allows for a dynamic interaction as if one is conversing with a human medical advisor.

***Example Interaction***



- User Query: "Hey doctor, I have a small migraine, can you suggest something for this please?"

- DocGPT Response: "Rest in a quiet, dark room and lie down. Hydration: Drink plenty of water to stay hydrated. Over-the-counter pain relief: Take an over-the-counter pain reliever like ibuprofen or acetaminophen."

The interaction concludes once the user's queries are adequately addressed by DocGPT. The AI’s responses guide users on how to manage their symptoms or direct them to seek professional medical advice if the situation warrants it. This ensures that the queries are resolved to the satisfaction of the users, making the AI a valuable tool in providing first-line medical guidance.

This dynamic interface effectively demonstrates the practical application of DocGPT in everyday health management, highlighting its potential as a reliable AI health assistant.

***Chapter-5: Conclusion***

#### **Effective Pest Identification**

The deep learning-based pest classification model developed in this project leverages advanced algorithms to accurately identify various agricultural pests. By analyzing images or sensor data, the model can distinguish between different types of pests, such as insects, fungi, or diseases, with high precision. This capability is crucial for farmers and agronomists to swiftly detect and address pest infestations before they cause significant damage to crops.

#### **Improved Decision-Making**

The model's accurate and timely pest detection capabilities revolutionize decision-making in agriculture. By providing real-time insights into pest presence and severity, farmers and agronomists can make informed choices regarding pest management strategies. Whether it's implementing targeted interventions, adjusting pesticide application schedules, or deploying natural predators, having reliable data at their fingertips empowers agricultural stakeholders to mitigate risks effectively and optimize crop health and productivity.

#### **Real-World Applicability**

One of the key strengths of this model lies in its adaptability and applicability to real-world agricultural scenarios. Through rigorous training on diverse datasets encompassing various pest species and crop types, the model has demonstrated robust performance and generalization capabilities. This means that it can effectively operate across different geographical regions, climates, and farming practices, making it a versatile tool for farmers worldwide. Whether in large-scale commercial farms or smallholder operations, the model's reliability and scalability make it a valuable asset for sustainable agriculture practices and food security efforts globally.

# 

#### **Multispectral Imaging**

By incorporating multispectral and hyperspectral imaging technologies, the pest classification model can access a broader range of spectral information beyond the visible spectrum. This enhancement allows the model to detect subtle differences in pest signatures that may not be visible to the human eye alone. Multispectral imaging captures data across multiple bands of the electromagnetic spectrum, providing valuable insights into plant health, stress levels, and pest presence. By leveraging this additional information, the model's accuracy and reliability in identifying pests are further enhanced, enabling more proactive and targeted pest management strategies.

#### **Automated Intervention**

Integrating the pest classification model with autonomous robotic systems and precision spraying technologies revolutionizes pest management practices. By connecting the model's pest detection capabilities with automated intervention tools, such as robotic drones or smart sprayers, farmers can deploy targeted pest control measures with unprecedented precision and efficiency. These autonomous systems can navigate fields, identify pest hotspots identified by the model, and administer precise doses of pesticides or biological control agents only where necessary. This not only reduces the reliance on broad-spectrum chemical treatments but also minimizes pesticide usage, environmental impact, and labor costs while maximizing crop protection.

#### 

#### **Cloud-Based Deployment**

Developing a cloud-based platform for deploying the pest classification model democratizes access to advanced agricultural technology. By hosting the model on a cloud infrastructure, it becomes accessible to a wider range of users, including small-scale farmers with limited resources. The platform offers real-time monitoring and decision support capabilities, allowing farmers to remotely access pest detection results, receive actionable insights, and make informed decisions from anywhere with internet connectivity. This democratization of technology empowers farmers of all scales to implement proactive pest management strategies, optimize resource allocation, and improve overall crop yields and profitability. Additionally, cloud-based deployment enables seamless updates and scalability, ensuring the model remains cutting-edge and adaptable to evolving agricultural challenges.