Report on Patient Record Query Chatbot System

---

1. Introduction

This report presents a simple AI-powered chatbot system that answers queries related to patient records. The chatbot processes questions regarding patient data, such as vital signs, and provides answers by leveraging OpenAI's GPT-3 model. The system reads a CSV file containing patient information, converts the data to a JSON format, and responds to user queries based on the dataset. This approach enables users to interact with the data naturally, asking questions like "What is the max age of the patient?" or "On which date was the blood pressure high?"

2. Objective

The goal of the system is to use natural language processing (NLP) to answer user queries based on patient data stored in a CSV file. The system is designed to:

- Read and process CSV data in memory.
- Answer a variety of queries about patient vital signs, such as age, blood pressure readings, and more.
- Provide a user-friendly interface where users can ask questions in natural language.

3. System Overview

The core components of the system include:

- **CSV to JSON Conversion**: The CSV file is read and converted into a JSON format. This allows for easier processing and querying of the data in-memory without the need to save it to disk.
- **User Query Processing**: When a user asks a question, the system sends the query to OpenAI's GPT-3 model along with relevant data from the patient records. GPT-3 processes the information and generates a response.
- **User Interaction**: The chatbot continuously accepts user input, processes each query, and provides answers until the user exits.

4. Key Functions

CSV to JSON Conversion: The `csv_to_json_in_memory` function converts the CSV data into a list of dictionaries, where each dictionary represents a patient record.

```
def csv_to_json_in_memory(csv_file_path):
    with open(csv_file_path, mode='r', encoding='utf-8') as csv_file:
        csv_reader = csv.DictReader(csv_file)
        data = [row for row in csv_reader]
    return data
```

Processing User Queries: The `process_patient_query` function constructs a prompt for GPT-3, sends it the query, and retrieves a detailed answer.

```
def process_patient_query(question, json_data):
    prompt = f"Here is a dataset of patient records:\n\n{json_data}\n\nUser question:
{question}\nProvide a detailed answer based on the patient data above."
    response = openai.Completion.create(
        engine="gpt-3.5-turbo-instruct",
        prompt=prompt,
        temperature=0.7,
        max_tokens=150,
        n=1,
        stop=None
    )
    return response.choices[0].text.strip()
```

Interactive Query System: The system continuously accepts queries from users and provides responses until the user types "exit".

```
def handle_user_query():
    while True:
        user_query = input("Ask a question related to the patient records (or type
'exit' to quit): ").strip()
        if user_query.lower() == "exit":
            break
        answer = process_patient_query(user_query, json_data)
        print("Answer:", answer)
```

5. Example Use Cases

1. **Max Age and Patient Name**:
    - **Query**: "What is the max age of the patient from the record, and what is his/her name?"
    - **Response**: The system processes the data to find the oldest patient and returns the name and age.
2. **High Blood Pressure Date**:
    - **Query**: "On which date was the blood pressure high for a patient?"
    - **Response**: The system identifies the highest blood pressure reading and provides the date it occurred.
3. **Sort by Blood Pressure**:
    - **Query**: "Provide a list of patients based on blood pressure in descending order."
    - **Response**: The system sorts the patient records by blood pressure and returns the names from highest to lowest.

6. Limitations

- **Data Quality**: The accuracy of responses depends on the quality and structure of the CSV file. Missing or incorrect data can lead to wrong answers.
- **Contextual Understanding**: GPT-3 may not always provide precise answers without detailed prompts or context, especially for more specific questions.
- **Performance**: As the dataset grows, in-memory processing may become less efficient, especially for large CSV files.

7. Conclusion

The chatbot system demonstrates a straightforward way to interact with structured patient data using natural language queries. It leverages OpenAI's GPT-3 model to answer questions based on a CSV file converted to JSON. This solution is effective for providing detailed insights from structured datasets, making it useful for various data exploration and reporting scenarios.

Future improvements could include better data processing for large datasets, handling more complex queries, and improving the accuracy of responses with further training on medical data.