Leveraging Generative AI for Medical Data Analysis and Simplification: A Step-by-Step Implementation

Maziyar Mirzaei

UT Austin - Fall 24

GitHub Link

Dataset Overview

Dataset: Synthea Synthetic Dataset

- Patients: Demographics and medical history
- Conditions: Diagnosed conditions
- Encounters: Medical interactions and costs

```python
import pandas as pd

# Load datasets
patients = pd.read_csv('patients.csv')
encounters = pd.read_csv('encounters.csv')
conditions = pd.read_csv('conditions.csv')
```

Data Merging

Create a unified dataset for analysis by merging datasets.

Steps:

- Merge `conditions` with `encounters` using `ENCOUNTER` as the key.
- Merge the resulting dataset with `patients` using `PATIENT_x` as the key.

```python
# Merge datasets to create unified patient records
# Merge conditions with encounters
conditions_encounters = pd.merge(
    conditions, encounters, left_on="ENCOUNTER", right_on="Id", how="left"
)

# Merge the result with patients
patient_records = pd.merge(
    conditions_encounters, patients, left_on="PATIENT_x", right_on="Id", how="left"
)
```

Data Cleaning

Select and rename relevant fields for summarization.

- **Output**: A cleaned DataFrame ready for analysis.

```
# Correct the column names and re-select key fields for summariz
summary_data = patient_records[
    [
        "PATIENT_x",
        "BIRTHDATE",
        "DESCRIPTION_x",  # Condition description
        "DESCRIPTION_y",  # Encounter description
        "START_x",  # Condition start
        "STOP_x",   # Condition end
        "CITY",
        "STATE",
        "HEALTHCARE_EXPENSES",
        "HEALTHCARE_COVERAGE",
    ]
]

# Rename columns for clarity
summary_data.columns = [
    "Patient_ID",
    "Birthdate",
    "Condition_Description",
    "Encounter_Description",
    "Condition_Start",
    "Condition_End",
    "City",
    "State",
    "Healthcare_Expenses",
    "Healthcare_Coverage",
]

# Display the cleaned and merged dataset
summary_data_cleaned = summary_data.head(
summary_data_cleaned
```

Selecting a Sample Patient Record for Summarization

To create a structured input format from a patient's record for use in a natural language processing (NLP) model like GPT, enabling summarization in simple terms.

- Standardizes the data format for model processing.
- Prepares the record for high-quality summarization output.

```python
# Select a sample patient record for summarization
sample_record = summary_data.iloc[0]

# Create a structured summary for GPT input
record_for_prompt = f"""
Patient ID: {sample_record['Patient_ID']}
Birthdate: {sample_record['Birthdate']}
Condition: {sample_record['Condition_Description']}
Encounter: {sample_record['Encounter_Description']}
Condition Start: {sample_record['Condition_Start']}
Condition End: {sample_record['Condition_End']}
City: {sample_record['City']}
State: {sample_record['State']}
Healthcare Expenses: ${sample_record['Healthcare_Expenses']}
Healthcare Coverage: ${sample_record['Healthcare_Coverage']}
"""

# Display the structured input for the GPT prompt
record_for_prompt
```

'\nPatient ID: 30a6452c-4297-a1ac-977a-6a23237c7b46\nBirthdate: 1994-02-06\nCondition: Housing unsatisfactory (finding)\nEncounter: General examination of patient (procedure)\nCondition Start: 2012-04-01\nCondition End: nan\nCity: Braintree\nState: Massachusetts\nHealthcare Expenses: $56904.96\nHealthcare Coverage: $18019.99\n'

Record Summarization

Generate layperson-friendly summaries of patient records using GPT.

Steps:

- Create a structured input for GPT.
- Use GPT to summarize the record.

```python
import openai

# Set OpenAI API key
openai.api_key = "sk-proj-...-zGMJbadOXW7Q8NW128k24eF08n-S22IsqyumBRh4A"


# Create a GPT prompt for summarization
def generate_summary(record):
    prompt = f"""
    You are a medical assistant. Summarize the following medical record in simple
    Provide explanations for any medical terms or conditions if necessary.
    Record: {record}
    """
    try:
        response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo",  # Use a model available to your API key
            messages=[
                {"role": "system", "content": "You are a helpful assistant."},
                {"role": "user", "content": prompt}
            ],
            temperature=0.7
        )
        return response["choices"][0]["message"]["content"]
    except Exception as e:
        return f"Error generating summary: {e}"

# Test the summarization
summary = generate_summary(record_for_prompt)
```

Record Summarization

Example Prompt:

- You are a medical assistant. Summarize the following medical record in simple terms. Record: {structured_record}

```
Summary:

- Patient ID: 30a6452c-4297-a1ac-977a-6a23237c7b46

- Birthdate: February 6, 1994

- Location: Braintree, Massachusetts

- Condition: Housing unsatisfactory (finding)

- Encounter: General examination of patient

- Condition Start Date: April 1, 2012

- Healthcare Expenses: $56,904.96

- Healthcare Coverage: $18,019.99


Explanation:

- The medical record indicates that the patient's living situation is found to be unsatisfactory.

- The patient underwent a general examination during the medical encounter.

- The condition of unsatisfactory housing was
```

Summarizing Multiple Patient Records

To process and summarize multiple patient records, generating layperson-friendly explanations for each record, and save the results to a CSV file.

Function Definition:

- A function, `summarize_multiple_records`, is defined to loop through a specified number of patient records (`n_records`) in the dataset and generate summaries for each.
- Each record is structured into a prompt, similar to the single-record approach, and sent to GPT for summarization.

Generating Summaries:

The `summarize_multiple_records` function is called with the `summary_data` dataset, processing the first 10 records.

- Each summary contains:
- A simplified explanation of the patient's medical condition and encounter.
- Details like healthcare expenses and coverage.

Example Output:

```
Summary:

- Patient: Born on February 6, 1994, in Braintree, Massachusetts.

- Condition: Identified as having unsatisfactory housing during a general examination on April 1, 2012.

- Healthcare Expenses: $56,904.96.
```

The generated summaries are saved to a CSV file (patient_summaries.csv) for future reference.

Code:

```
summaries = summarize_multiple_records(summary_data,
n_records=10)

summaries_df = pd.DataFrame(summaries)

summaries_df.to_csv("patient_summaries.csv", index=False)

print("Summaries saved to patient_summaries.csv")
```

CSV Content Example:

The CSV file contains two columns:

Patient_ID: The unique ID of the patient.
Summary: The generated summary for the patient.

Example Rows:

Patient_ID	Summary
30a6452c-4297-a1ac-977a-6a23237c7b46	Summary: The patient had a general examination and was identified with unsatisfactory housing. Healthcare expenses were $56,904.96, with $18,019.99 covered.

Simplifying Medical Notes for Different Patient Groups

To tailor medical notes for specific audience groups (e.g., children, adults, seniors) by simplifying the language to suit their comprehension levels.

Function Definition:

- A function, `simplify_medical_notes`, takes two inputs:
- **`record`**: The medical note to be simplified.
- **`target_group`**: The intended audience (e.g., child, adult, senior).
- A GPT prompt is constructed to guide the model in simplifying the note for the specified audience.

Sample Record Selection:

A sample condition description is selected from the dataset (`summary_data.iloc[0]['Condition_Description']`).

Example:

```
Condition: Housing unsatisfactory (finding)
```

Simplification for Different Groups:

The function is called three times, each with a different target group:

- **Children**
- **Adults**
- **Seniors**

```
simplifications = { "Children":
simplify_medical_notes(sample_note, "child"), "Adults":
simplify_medical_notes(sample_note, "adult"), "Seniors":
simplify_medical_notes(sample_note, "senior") }
```

Example Simplified Outputs:

- **Children**:

  Simplified for basic comprehension.

  Output: *"Your home needs some improvements."*

- **Adults**:

  More neutral and factual.

  Output: *"Not happy with living situation."*

- **Seniors**:

  Focus on clarity and respect.

  Output: *"Your living situation is not good (finding)."*

Tailored Communication: Ensures medical information is understandable for different age groups and literacy levels.

Improved Patient Engagement: Simplifying notes fosters better understanding and compliance with medical advice.

Tree-of-Thought Reasoning for Medical Records

To apply a hierarchical reasoning approach to analyze medical records, generating a structured summary, detailed explanation, and actionable recommendations.

Function Definition:

The `tree_of_thought_reasoning` function processes a medical record in three hierarchical steps:

1. **Summarize the Record**: Extract key information from the medical record.
2. **Explain the Condition**: Provide a detailed explanation of the condition(s) mentioned in the summary.
3. **Generate Recommendations**: Offer actionable recommendations based on the explained condition.

Testing the Function:

- A sample medical record is selected as input.
- The function processes this record and returns three outputs: summary, explanation, and recommendations.

```
record_for_thought = summary_data.iloc[0]['Condition_Description']

tree_of_thought_result = tree_of_thought_reasoning(record_for_prompt)

print(tree_of_thought_result)
```

Example Output:

Summary:

- *"The patient with ID 30a6452c-4297-a1ac-977a-6a23237c7b46, born on February 6, 1994, has a finding of unsatisfactory housing noted during a general examination on April 1, 2012, in Braintree, Massachusetts. The patient has incurred healthcare expenses totaling $56,904.96, with $18,019.99 covered by healthcare coverage. The end date of the housing condition is not specified in the record."*

Explanation:

- *"The patient's living conditions were found to be inadequate or substandard during a general examination. The record notes healthcare expenses of $56,904.96 and insurance coverage of $18,019.99, suggesting potential financial challenges."*

Recommendations:

Assess Housing: Verify if the housing issue persists.

Explore Assistance: Research local housing programs or financial counseling services.

Provide Emotional Support: Address stress from health and housing issues.

Follow-Up Care: Ensure regular health monitoring.

Insights and Future Direct

Key Insights:

Generative AI Effectiveness:

- Successfully simplifies complex medical data for various audiences (children, adults, seniors).
- Breaks down medical records into structured summaries, explanations, and actionable recommendations.

Adaptability:

- Tree-of-Thought Reasoning showcases AI's ability to provide layered insights, enhancing the decision-making process.

Improved Patient Communication:

- Tailored medical notes promote better understanding and engagement across diverse patient groups.

Scalability:

- The pipeline handles multiple records efficiently, with results easily exportable for integration into healthcare systems.

Ending Note

This project demonstrates how generative AI can revolutionize healthcare by making data more accessible and actionable for all stakeholders.