The script utilizes the OpenAI API to process a doctor's appointment transcription and extracts medical information, including diagnosis, procedure, and Australian Medicare codes. The script also enriches this data with additional details from a local Medicare codes dataset.

**Dependencies**

* openai: A library to interact with OpenAI's API.
* dotenv: A library to load environment variables from a .env file.
* os: A library to interact with the operating system.
* json: A library to handle JSON data.

**Setup**

Before running the script, ensure you have the following environment variables set in a .env file:

* API\_KEY: Your Azure OpenAI API key.
* API\_VERSION: The version of the API you are using.
* AZURE\_ENDPOINT: The endpoint for the Azure OpenAI service.

**Functions**

**provide\_australian\_medicare\_codes(json\_transcription)**

This is the main function that processes the input JSON transcription and returns enriched medical information.

**Parameters:**

* json\_transcription: A JSON object containing the conversation transcription.

**Steps:**

1. **Load Environment Variables:**

load\_dotenv()

This loads the environment variables from the .env file.

1. **Initialize AzureOpenAI Client:**

client = AzureOpenAI(

api\_key=os.getenv("API\_KEY"),

api\_version=os.getenv("API\_VERSION"),

azure\_endpoint=os.getenv("AZURE\_ENDPOINT")

)

This initializes the AzureOpenAI client with the API key, version, and endpoint.

1. **Prepare the Prompt:**

prompt = f"""

...

"""

This defines the prompt for the OpenAI model to extract relevant information from the doctor's notes.

1. **Request Summary from Azure OpenAI:**

python

Copy code

response = client.chat.completions.create(

model="gmdevgpt4model",

...

messages=[

{"role": "system", "content": "Assistant is a conversation summarizer between the doctor and patient."},

{"role": "user", "content": prompt}

]

)

This sends the request to the Azure OpenAI service to generate a summary based on the prompt.

1. **Parse the Response:**

output = response.choices[0].message.content.strip()

summaries = []

for item in output.split("},"):

...

summaries.append(summary\_data)

This parses the JSON response from the API into individual summaries.

1. **Transform the Summaries:**

python

Copy code

transformed\_data = []

for item in summaries:

...

transformed\_data.append({

'diagnosis': diagnosis,

'procedure': procedure,

'australian\_medicarecode': code

})

This transforms the summaries into a more structured format.

1. **Load Medicare Codes Data:**

with open("australianmedicarecodes.json", "r") as f:

medicare\_codes\_data = json.load(f)

This loads the Medicare codes data from a local JSON file.

1. **Create a Dictionary for Quick Lookup:**

python

Copy code

mbs\_dict = {item["ItemNum"]: item for item in medicare\_codes\_data["MBS\_XML"]["Data"]}

This creates a dictionary for quick lookup of Medicare codes.

1. **Filter and Enrich Data:**

python

Copy code

filtered\_list\_data = []

for item in transformed\_data:

...

filtered\_list\_data.append(item)

This filters and enriches the transformed data with additional details from the Medicare codes dataset.

1. **Return the Enriched Data:**

python

Copy code

return filtered\_list\_data

**Main Execution**

When the script is executed directly, it reads an input JSON file, processes it using the provide\_australian\_medicare\_codes function, and prints the response.

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

with open("input\_file.json", "r") as f:

input\_json = json.load(f)

response = provide\_australian\_medicare\_codes(input\_json)

print(response)

**Example**

To run the script, ensure you have an input file named input\_file.json with the transcription data and a australianmedicarecodes.json file with the Medicare codes. Then execute the script to get the enriched medical information.