**Task: SAMPLE TRANSCRIPT OF A DOCTOR AND A PATIENT**

**1. Importing Necessary Libraries:**

The code imports several libraries required for natural language processing (NLP) tasks, model inference, and web app development:

* **json**: For handling JSON files.
* **re**: For regular expressions, used in text processing.
* **streamlit**: For creating a web-based interface.
* **transformers**: Specifically, the pipeline for question-answering using pre-trained models.
* **sentence\_transformers**: For sentence embeddings, allowing for semantic similarity comparisons.
* **numpy**: For numerical operations, particularly for working with arrays.
* **spacy**: For natural language processing tasks such as Named Entity Recognition (NER).

**2. Loading the spaCy Model:**

python

Copy code

nlp = spacy.load("en\_core\_web\_sm")

* Loads the spaCy English model, which can be used for various NLP tasks like NER, tokenization, and more.

**3. Function Definitions:**

**a. load\_transcript(json\_file):**

* **Purpose**: Loads and parses a JSON file containing the doctor-patient conversation transcript.
* **Input**: A file path to the JSON file.
* **Output**: A Python dictionary representing the JSON content.

**b. extract\_dialogues(transcript):**

* **Purpose**: Extracts individual sentences or dialogues from the transcript.
* **Process**:
  + Iterates over each entry in the transcript.
  + Splits the text into sentences using regular expressions to detect sentence-ending punctuation.
  + Collects all sentences in a list.
* **Output**: A list of individual dialogue sentences.

**c. clean\_and\_preprocess\_text(text):**

* **Purpose**: Cleans and preprocesses text for further processing.
* **Process**:
  + Converts text to lowercase.
  + Removes punctuation and non-word characters.
  + Tokenizes the text by splitting on spaces.
* **Output**: A cleaned and preprocessed string.

**d. find\_relevant\_section(query, sections, embedding\_model, section\_embeddings):**

* **Purpose**: Finds the most relevant section in the transcript that matches the user's query.
* **Process**:
  + Encodes the user's query into an embedding vector.
  + Computes cosine similarities between the query embedding and precomputed embeddings of the transcript sections.
  + Identifies the section with the highest similarity score.
* **Output**: The most relevant section of the transcript.

**e. get\_answer\_from\_section(question, context, qa\_model):**

* **Purpose**: Generates an answer to the user's question based on the most relevant transcript section.
* **Process**:
  + Uses a pre-trained question-answering model to generate an answer.
  + Returns the model's answer, unless it's too short, in which case it returns an error message.
* **Output**: A string containing the answer.

**4. Main Program Flow:**

**a. Loading and Preprocessing Transcript:**

* **Transcript Loading**: The transcript is loaded from a JSON file using load\_transcript.
* **Dialogue Extraction**: Extracts dialogues from the transcript using extract\_dialogues.
* **Error Handling**: Checks if dialogues are found; if not, it displays an error using Streamlit.

**b. Text Preprocessing:**

* Cleans and preprocesses each dialogue using clean\_and\_preprocess\_text.

**c. Model Loading:**

* **qa\_model**: Loads a pre-trained question-answering model from the Hugging Face library (bert-large-uncased-whole-word-masking-finetuned-squad).
* **embedding\_model**: Loads a sentence embedding model from the Sentence Transformers library (multi-qa-mpnet-base-dot-v1).

**d. Embedding Computation:**

* Computes embeddings for each cleaned dialogue in the transcript.

**e. User Input Handling:**

* **Question Input**: Prompts the user to enter a question via a text input field in the Streamlit app.
* **Finding the Relevant Section**: Uses the find\_relevant\_section function to locate the most relevant dialogue based on the user's question.
* **Answer Extraction**: Extracts the answer from the identified relevant section using the get\_answer\_from\_section function.
* **Displaying the Answer**: Outputs the answer on the Streamlit interface.

**5. Summary:**

This code creates a Streamlit web app that allows users to ask questions based on a doctor-patient conversation transcript. The app:

* Loads and preprocesses the transcript.
* Embeds the transcript's dialogues for semantic similarity matching.
* Uses a question-answering model to find and display answers to user queries. The app provides a straightforward interface where users can input questions and receive answers based on the most contextually relevant section of the transcript.