🏛️Indian Penal Code Chatbot: Your Legal Assistant 🗣️

📜 In this assignment, we will build a chatbot for the Indian Penal Code. We'll start by downloading the official Indian Penal Code document, and then we'll create a chatbot that can interact with it. Users will be able to ask questions about the Indian Penal Code and have a conversation with it. 🤖🗣️ This chatbot will help users understand the legal provisions easily. 🌐 It's a great tool for students, lawyers, and anyone interested in Indian law. 📘 Get ready to explore the IPC like never before! 🚀

Step-1:Document Loading

https://raw.githubusercontent.com/venkatareddykonasani/Datasets/master/IPC/THE_INDIAN_PENAL_CODE.pdf

Instructions

1. **Download a PDF file using wget:**
    - Import the `wget` module and use it to download the PDF file from the specified URL.

```
# Download the THE_INDIAN_PENAL_CODE.pdf file
!wget
https://raw.githubusercontent.com/venkatareddykonasani/Datasets/master/I
```

2. **Load the PDF file:**
    - Use the `PyPDFLoader` from `PyPDFLoader` library to load the downloaded PDF file.
3. **Extract text from the PDF:**
    - Loop through the pages of the PDF document and concatenate the text content of each page into a single string.
4. **Print the first 1000 characters of the extracted text:**
    - Use string slicing to print the first 1000 characters of the concatenated text.
5. **Print the number of lines, words, and characters in the**

extracted text:

- Use string methods to split the text by newline characters to count the number of lines.
- Use string methods to split the text by spaces to count the number of words.
- Use the `len` function to count the total number of characters in the text.

In [ ]:
```
#Write your code here
```

In [ ]:
```

```

Step-2: Split the data into Chunks

Instructions

1. **Install and Import Necessary Libraries:**
   - Ensure you have the `langchain` library installed. If not, install it using `pip`.
   - Import `RecursiveCharacterTextSplitter` from the `langchain.text_splitter` module.
2. **Initialize the Text Splitter:**
   - Create an instance of `RecursiveCharacterTextSplitter` with a specified `chunk_size` and `chunk_overlap`.
3. **Split the Text into Chunks:**
   - Use the `split_text` method of the text splitter instance to divide the full text into chunks.
4. **Print the Number of Chunks and the First Chunk:**
   - Print the total number of chunks created.
   - Print the content of the first chunk.

In [ ]:
```
#Write your code here
```

In [ ]:
```

```

Step-3: Creating embeddings and Storing in Vector Stores

Instructions

1. Install the Required Library

First, you need to install the `ChromaDB` library, which is essential for creating and managing vector stores. Use the following command to install it:

2. Import Necessary Modules

Next, import the necessary modules from `langchain`. You will need `OpenAIEmbeddings` for creating embeddings and `Chroma` for managing vector stores.

3. Create Embeddings

Instantiate the `OpenAIEmbeddings` class to create embeddings. This class will be used to generate embeddings for your text data.

4. Create and Populate the Vector Store

Use the `Chroma` class to create a vector store from your text data. The `from_texts` method is used to convert your text data (`chunks`) into embeddings and store them in a persistent directory (`IPC_db`).

5. Persist the Vector Store

Finally, ensure that the vector store is saved by calling the `persist` method. This will save the data to the specified directory, making it available for future use.

In [ ]:
```
#Write your code here
```

In [ ]:
```

```

Step-4: Conversation and Retrieval Chain

Instructions

1. Set Up the Retriever

First, create a retriever from the `IPC_db` vector store. This retriever will be used to fetch relevant documents based on user queries.

2. Define the Query

Create a query string that you want to search in the vector store. This query should be specific to the information you are looking for.

```
query = """
What is the section related to take part in an unlawful assembly or
riot.
"""
```

3. Retrieve Relevant Documents

Use the retriever to get documents that are relevant to your query. The `get_relevant_documents` method will return a list of documents matching the query.

```
result = retriever.get_relevant_documents(query)
```

4. Display the Results

Loop through the retrieved documents and print their content. This will display the relevant information from the documents.

5. Set Up the Conversational AI

Instantiate the `ChatOpenAI` class to create a language model for conversation. Set the temperature to 0 for deterministic responses.

6. Create a Memory Buffer

Create a memory buffer using `ConversationBufferMemory`. This buffer will store the conversation history.

7. Set Up the Conversational Retrieval Chain

Use the `ConversationalRetrievalChain` class to create a retrieval-augmented generation (RAG) system. This system will handle the conversation and retrieval tasks.

In [ ]:
```
#Write your code here
```

In [ ]:
```

```

Step-5 : Conversation

Instructions

1. **Import the Necessary Libraries**
    - Import any libraries required for conversational AI. In this case, we are assuming a function `conversational_RAG` is available for answering legal questions. If the function is part of an external library, ensure that the library is installed and imported.
2. **Create the User Input Loop**
    - Initialize a variable `user_input` to store the message input by the user.
    - Use a `while` loop to continuously prompt the user for input until they type "quit".
3. **Define the Conversational Function**
    - Define or import the function `conversational_RAG` which will take a dictionary with the key "question" and return a dictionary with the key "answer".
4. **Process User Input**
    - Inside the loop, capture user input using `input()`.
    - Pass the user input to the `conversational_RAG` function.
    - Print the response received from the function.
5. **Exit Condition**
    - The loop should terminate when the user types "quit".
6. **Sample Interactions**

```
# What is the section related to take part in an unlawful assembly or
riot.
# What is the punishment for that
# What is the Punishment for using a false property mark.
# Which section deals with that?
# Which section deals with Counterfeiting currency-notes or bank-
notes?
# tell me more about the section and the punishment for
Counterfeiting currency-notes
```

In [ ]:

In [ ]: