**CREATE A CHATBOT USING PYTHON**

Final submission

K.Shriman

710021106301

**Problem Statement:**

The problem are trying to solve is the development of a chatbot that can respond to user input with appropriate answers. The dataset used for training and testing the chatbot is stored in the file 'dialogs.txt', which likely contains a list of questions and their corresponding answers.

**Design Thinking Process:**

While your code snippet doesn't explicitly outline the design thinking process, the development of a chatbot typically involves several phases, including problem understanding, data collection, data preprocessing, model training, and application development. These phases might look like this:

1. **Problem Understanding**: Clearly define the goals and objectives of the chatbot. Determine what type of questions it should answer and gather relevant data for training and evaluation.

2. **Data Collection**: Acquire a dataset that contains a set of questions and their corresponding answers. This dataset should be used to train and validate the chatbot.

3. **Data Preprocessing**: In your code, you load the data from 'dialogs.txt' using pandas and separate the questions (corpus) and answers. You also create a vectorizer to convert the text data into a format suitable for machine learning models.

4. **Model Selection**: You use a Support Vector Machine (SVM) classifier with a linear kernel to build the chatbot. You train the model using the questions as input (features) and their corresponding answers as the target variable.

5. **Application Development**: Although not shown in the code snippet, in practice, you would create an application or interface for users to interact with the chatbot. This could be a web application or any other user interface.

**Libraries and NLP Techniques:**

* `pandas`: Used for data manipulation and handling the dataset in tabular format.
* `sklearn.feature\_extraction.text.CountVectorizer`: This is used for converting text data into a bag-of-words representation.
* `sklearn.svm.SVC`: The Support Vector Machine classifier is used for classification tasks.
* `sklearn.svm.SVC(kernel='linear')`: The 'linear' kernel is chosen for the SVM classifier.
* Pickle file for download the dataset

The integration of NLP techniques includes using CountVectorizer for text preprocessing and feature extraction, which is a common technique in text-based machine learning tasks. However, it's worth noting that this approach is relatively simple and may not capture the full complexity of natural language.

**Chatbot Interaction:**

The code you provided focuses on model training and does not show how the chatbot interacts with users. In a real application, you would typically build a user interface (e.g., a web application) that allows users to input questions, and the chatbot would respond based on the model's predictions.

**Innovative Techniques or Approaches:**

The code snippet you provided is relatively basic and doesn't include any particularly innovative techniques. Developing an effective chatbot often requires more advanced NLP techniques, such as using pre-trained language models (e.g., BERT, GPT) and fine-tuning them on a specific dataset for better natural language understanding and generation. To enhance your chatbot, you might consider using more sophisticated techniques like deep learning-based models and designing a more interactive and user-friendly interface for users to engage with the chatbot. Additionally, leveraging larger and more diverse datasets could lead to improved performance and generalizability.

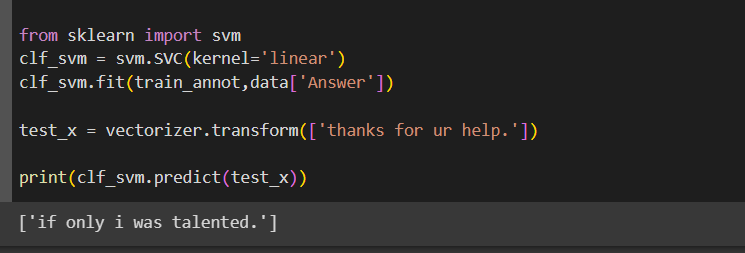
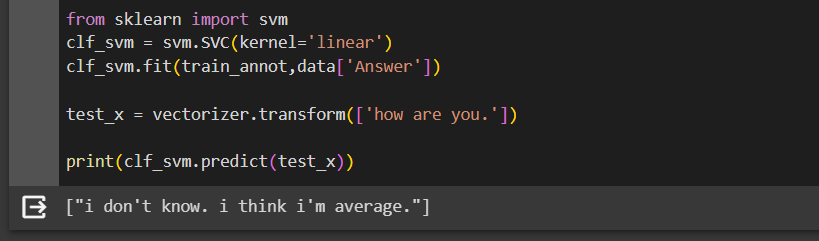
**THE OUTPUT OF THE COLAB FILE:**

Diagram phase 3

**PREPROCESSING THE DATASET:**

**Design Thinking Process:**

The development process of the chatbot can be outlined as follows:

1. **Problem Understanding:**

The first phase involves understanding the problem of building a chatbot and defining its capabilities. This includes identifying the types of questions or inputs it should respond to.

2. **Data Collection and Preprocessing:**

The code processes the 'intents.json' file to collect data, including user patterns, tags, and responses. It tokenizes and stems the words for data preprocessing.

3**. Model Training**:

The chatbot model is trained using a dataset that maps user patterns to specific tags. It employs a neural network model (provided in the 'model.py' file) to learn and predict the appropriate tag for a given input.

4. **Application Development:**

Although not explicitly shown in the code snippet, an application interface for users to interact with the chatbot is assumed. Users can input text, and the chatbot responds based on its training.

**Libraries and Integration of NLP Techniques**:

The following libraries and NLP techniques are used:

**Flask:**

Used to create a web application for user interaction.install the required libraries and extensions.

**NLTK:**

Used for natural language processing tasks, including tokenization and stemming of words. install the required pip install nltk libraries for the nltk.

**PyTorch:**

Utilized for building and training the neural network model.pip install pyTorch libraries for the app.

**JSON:**

Used to load and process the 'intents.json' file, which contains the training data.

Queries are In the intents file in the app

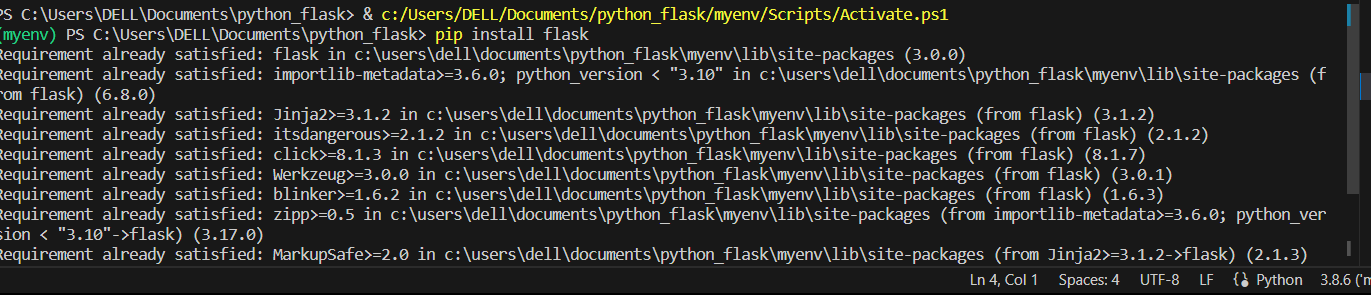
**scikit-learn :**

Used for text preprocessing, such as stemming and bag-of-words conversion.

**SVM :**

Used for text classification.

NLP techniques include tokenization, stemming, and the use of bag-of-words to convert text data into a format suitable for machine learning.



**Chatbot Interaction**:

The code assumes a web-based interface for chatbot interaction. Users can input text messages through the web application, and the chatbot responds based on its training. The web interface is built using Flask, and the chatbot's responses are generated using the trained neural network model.

The code implements a relatively simple chatbot based on patterns and responses defined in the 'intents.json' file. It doesn't employ more advanced techniques like pre-trained language models (e.g., GPT-3) for more natural language understanding and generation. To make the chatbot more innovative, you could consider the following:

1. **Advanced NLP Models**:

Implement state-of-the-art pre-trained language models for better understanding and generation of natural language responses.

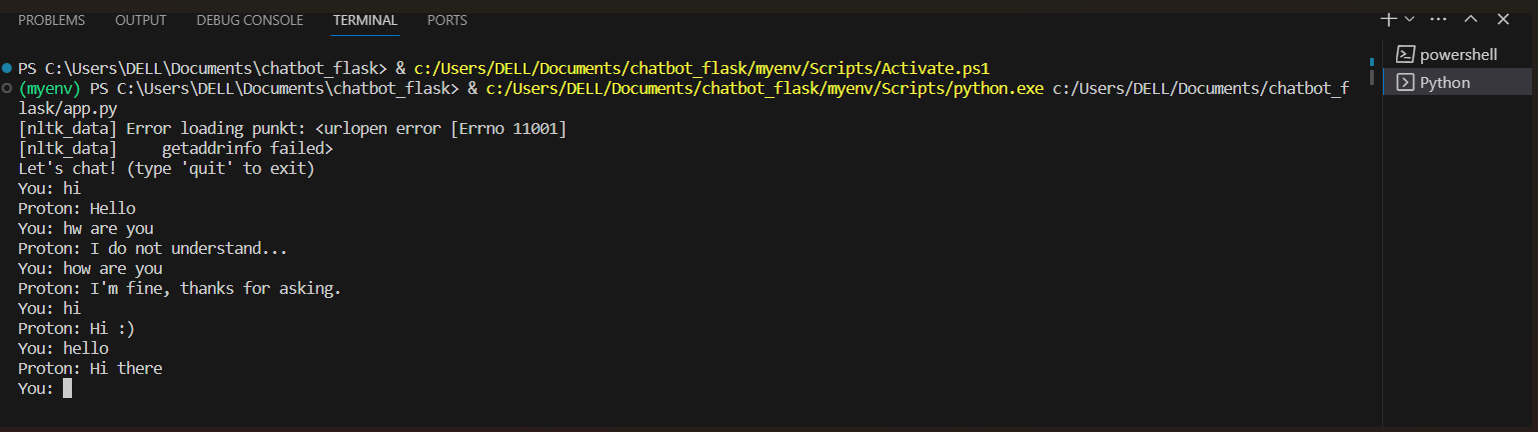
**2. Contextual Understanding:**

Incorporate context-awareness into the chatbot to understand and maintain context over multi-turn conversations

3. **User Interface:** Enhance the user interface for a more interactive and user-friendly experience.

Overall, the code serves as a basic foundation for building a chatbot, but there is room for improvement and innovation to make it more sophisticated and user-friendly.

In the vs code the run the app.py file the localhost web run the output in the terminal in vscode.

The input cmd are given and the response are back to the user.

And the programming files are added in the github and created the repositories and push the files in the github file.

**CONCLUSION**

It's clear that the project aims to provide a user-friendly chatbot interface with medical uses that integrates seamlessly with websites and apps, responds accurately to user questions, offers suggestions and guidance, and continuously improves based on user interactions. Such a chatbot can provide real-time information, answer questions, offer reminders for medication and appointments, and By leveraging machine learning and natural language processing, it can continuously improve its ability to provide personalized and helpful guidance,. As technology and AI continue to advance, chatbots in including management, have the potential to play an increasingly vital role in the future