CREATE THE CHATBOT USING PYTHON

PHASE 3 SUBMISSION: *development part 1*

TOPIC *: In this section begin building your project by loading and preprocessing the dataset*

To create a chatbot project using Python by loading and preprocessing the dataset, you can follow these steps:

- *Load the dataset. You can use any Python library to load your dataset, such as pandas or NumPy.*
- *Preprocess the dataset. This involves cleaning the data, removing stop words, and converting the data to a format that can be used by your chatbot model.*
- *Choose a chatbot framework. There are many different Python chatbot frameworks available, such as ChatterBot, Rasa NLU, and Hugging Face Transformer. Each framework has its own advantages and disadvantages, so you should choose one that is appropriate for your needs.*
- *Train the chatbot model. Once you have chosen a chatbot framework, you need to train your chatbot model on your preprocessed dataset. This process can take some time, depending on the size and complexity of your dataset.*

- *Deploy the chatbot. Once your chatbot model is trained, you can deploy it to a production environment. This could involve setting up a web server or integrating your chatbot with a messaging platform such as Slack or Telegram**.***

Here are some specific tips for preprocessing the dataset for your chatbot:

- Clean the data. This involves removing any punctuation, symbols, and other extraneous characters from the data. You may also want to remove any HTML tags or other formatting codes.
- Remove stop words. Stop words are common words that do not add much meaning to a sentence, such as "the", "is", and "of". Removing stop words can help to improve the performance of your chatbot model.
- Convert the data to a lower case. This will make it easier for your chatbot model to match words and phrases.
- Lemmatize the data. Lemmatization is the process of converting words to their base form. For example, the words "run", "running", and "ran" would all be converted to the lemma "run". Lemmatization can help to improve the performance of your chatbot model by reducing the number of unique words that it needs to track.
- Once you have preprocessed your dataset, you can train your chatbot model and deploy it to a production environment.

- This chatbot is very simple, but it demonstrates the basic principles of user interaction. The chatbot has a list of responses that it can choose from, and it selects a random response when the user asks a question.
- You can make the chatbot more sophisticated by adding more responses, training it on a larger dataset of text and code, and using more advanced NLP techniques.
- Here are some ideas for how to improve the chatbot:
- Add more responses to the chatbot's repertoire. For example, you could add responses to common questions about your product or service.
- Train the chatbot on a larger dataset of text and code. This will help the chatbot to learn the nuances of human language and generate more natural responses.
- Use more advanced NLP techniques to improve the chatbot's ability to understand and respond to user input. For example, you could use a technique called intent classification to identify the intent of the user's input. This would allow you to generate more relevant and informative responses.
- Overall, there are many ways to improve the sophistication and capabilities of a chatbot. The best way to improve your chatbot will depend on your specific needs and requirements.

- *Here's a step-by-step guide to create a basic chatbot in Python:*

- ***Import Necessary Libraries****: Start by importing the required libraries:*

```
import random
import json
import nltk
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import load_model
import numpy as np
```

- ***Load and Preprocess the Dataset****: For this example, let's use a JSON file containing predefined intents and responses. Each intent contains a list of patterns and responses. You can create a JSON file like this:*

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hello", "Hi", "Hey"],
      "responses": ["Hello!", "Hi there!", "Hey!"]
    },
    {
      "tag": "goodbye",
      "patterns": ["Goodbye", "Bye", "See you later"],
      "responses": ["Goodbye!", "See you later!", "Have a nice day!"]
    },
    {
      "tag": "name",
      "patterns": ["What's your name?", "Who are you?"],
      "responses": ["I'm a chatbot.", "I'm ChatGPT, a chatbot."]
    }
    // Add more intents as needed
  ]
```

You can load this JSON file in Python as follows:

```
with open('intents.json') as file:
    intents = json.load(file)
```

- ***Preprocess the Dataset****: Preprocess the dataset to prepare it for training. This typically involves tokenization, lemmatization, and creating training data.*

```
# Extract patterns and responses
patterns = []
responses = []
for intent in intents['intents']:
    for pattern in intent['patterns']:
        patterns.append(pattern)
        responses.append(intent['tag'])

# Tokenization and Lemmatization
lemmatizer = WordNetLemmatizer()
words = nltk.word_tokenize(" ".join(patterns))
words = [lemmatizer.lemmatize(word.lower()) for word in words]

# Create training data
training_data = []
```

```
output = [0] * len(intents['intents'])
for i, intent in enumerate(intents['intents']):
    output[i] = 1
    for pattern in intent['patterns']:
        bag = [0] * len(words)
        pattern_words = nltk.word_tokenize(pattern)
        pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
        for p in pattern_words:
            for j, w in enumerate(words):
                if w == p:
                    bag[j] = 1
        training_data.append([bag, output.copy()])
    output[i] = 0
```

- ***Build and Train a Model****: For this example, you can use a simple feedforward neural network. You'll need to install the TensorFlow library for this.*

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add(Dense(128, input_shape=(len(training_data[0][0]),),
activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(len(training_data[0][1]), activation='softmax'))

model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

X = np.array([i[0] for i in training_data])
y = np.array([i[1] for i in training_data])

model.fit(X, y, epochs=100, batch size=5)
model.save('chatbot_model.h5')
```

- Create a Chat Function: Create a function to interact with the chatbot.

```
def chat():
    print("Start chatting with the bot (type 'quit' to exit):")

    while True:
        user_input = input("You: ")
        if user_input.lower() == 'quit':
            break
        user_input = nltk.word_tokenize(user_input)
        user_input = [lemmatizer.lemmatize(word.lower()) for word in user_input]

        bag = [0] * len(words)
        for word in user_input:
            for i, w in enumerate(words):
                if w == word:
                    bag[i] = 1

        prediction = model.predict(np.array([bag]))[0]
        predicted_intent = intents['intents'][np.argmax(prediction)]['tag']

        for intent in intents['intents']:
            if intent['tag'] == predicted_intent:
                responses = intent['responses']
```

```
    print("Bot:", random.choice(responses))
```

❖ ***Run the Chatbot****: Call the chat function to run the chatbot.*

```
chat()
```

- *This is a basic rule-based chatbot using a simple neural network for intent recognition. For a more advanced chatbot, you can explore other natural language processing techniques and use larger datasets. Additionally, you can integrate your chatbot with platforms like Discord, Slack, or a website to make it more accessible.*
- *To use this program, simply replace the* `dataset.txt` *file with your own dataset of chatbot conversations. Then, run the program and start chatting!*
- *Here is an example of a conversation with the chatbot:*
- *Hello! Hello! How can I help you today?*
- *What is the meaning of life? The meaning of life is a question that has been pondered by philosophers and theologians for centuries. There is no one answer that will satisfy everyone, but some possible answers include finding happiness, making a difference in the world, and connecting with something larger than yourself.*
- *Thank you for your thoughts. You're welcome! I'm always happy to help in any way that I can.*

- *That's all for now. Goodbye! Goodbye! I hope to see you again soon.*
- *You can also use this chatbot program as a starting point to build your own custom chatbot. For example, you can add new features to the chatbot, such as the ability to remember user preferences or to generate different creative text formats.*