NAAN MUDHALVAN

ASSIGNMENT

PHASE - 3

NAME:SYED ALI FARZANA

ROLL NO : 2021105553

Building CHATBOT USING PYTHON by loading and preprocessing the dataset.

To load and preprocess the data set for building a chatbot using Python, you can follow these steps: 1. Import the necessary libraries: python import pandas as pd import numpy as np import re import nltk from nltk.tokenize import word_tokenize from nltk.corpus import stopwords 2. Load the data set into a Pandas DataFrame. The data set can be in various formats such as CSV, Excel, or JSON: python df = pd.read_csv('data.csv') 3. Clean the data by removing any unnecessary characters, converting the text to lowercase, and removing stopwords. You can define a function for text preprocessing: python def preprocess_text(text): # Remove special characters and digits text = re.sub(r'[^a-zA-Z\s]', '', text, re.I|re.A) # Convert to lowercase text = text.lower() # Tokenize the text tokens = word_tokenize(text) # Remove stopwords tokens = [w for w in tokens if not w in stopwords.words('english')] return ' '.join(tokens) 4. Apply the preprocessing function to the text column in the DataFrame: python df['processed_text'] = df['text'].apply(preprocess_text) 5. Split the data into training and testing sets. You can use scikit-learn's train_test_split to perform the split: python from sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test = train_test_split(df['processed_text'], df['label'], test_size=0.2, random_state=42) 6. Convert the text data into numerical features using vectorization techniques such as CountVectorizer or TF-IDF. Here is an example using CountVectorizer: python from sklearn.feature_extraction.text import CountVectorizer vectorizer = CountVectorizer() X_train_vectorized = vectorizer.fit_transform(X_train) X_test_vectorized = vectorizer.transform(X_test) 7. You can now use the preprocessed and vectorized data for training your chatbot model. Note that the above steps are just a basic example of data loading and preprocessing for a chatbot. Depending on your specific requirements and the type

of model you are building, you may need to perform additional steps such as handling missing values, handling imbalanced classes, or using more advanced text preprocessing techniques.

To build a simple chatbot using Python, you can follow the steps below: 1. Install the necessary libraries: - Install nltk library: pip install nltk - Install numpy library: pip install numpy - Install scikit-learn library: pip install scikit-learn 2. Import the required libraries: python import nltk import numpy as np import random import string from sklearn.feature_extraction.text import TfidfVectorizer from sklearn.metrics.pairwise import cosine_similarity 3. Download the necessary NLTK data: python nltk.download('punkt') nltk.download('wordnet') 4. Provide some sample text for the chatbot to train on: python text = """ Hello! How are you doing? What is your name? What are you up to? Goodbye! """ sentences = nltk.sent_tokenize(text) 5. Preprocess the text data: python lemmer = nltk.stem.WordNetLemmatizer() def preprocess(text): remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation) tokens = nltk.word_tokenize(text.lower().translate(remove_punct_dict)) return [lemmer.lemmatize(token) for token in tokens] preprocessed_sentences = [preprocess(sentence) for sentence in sentences] 6. Create a TF-IDF vectorizer and extract features: python vectorizer = TfidfVectorizer(tokenizer=preprocess, stop_words='english') tfidf_matrix = vectorizer.fit_transform(sentences) 7. Implement the chatbot logic: python def get_response(user_response): preprocessed_user_response = preprocess(user_response) tfidf_user_response = vectorizer.transform([user_response]) similarity_scores = cosine_similarity(tfidf_user_response, tfidf_matrix)[0] index = np.argmax(similarity_scores) return sentences[index] def chat(): print("Chatbot: Welcome! How can I assist you today?") while True: user_response = input("User: ") if user_response.lower() ==

'goodbye': print("Chatbot: Goodbye! Have a great day!") break else: print("Chatbot:", get_response(user_response)) chat() Upon running the code, you can interact with the chatbot by typing your inputs. The chatbot will provide responses based on the similarity of your input to the pre-defined sentences. The more training data and the more varied the sentences are, the better the chatbot's understanding and response will be.

Building the chatbot using python by preparing the environment and implementing basic user interactions

To build a chatbot using Python, you'll need to prepare your environment and implement basic user interactions. Here's a step-by-step guide: Step 1: Set up Python Environment 1. Install Python: Download and install the latest version of Python from the official website (python.org). Step 2: Install Required Libraries 1. Open a command prompt or terminal. 2. Install the necessary libraries using pip, Python's package manager. - Install nltk: Run the command pip install nltk to install the Natural Language Toolkit (NLTK). - Install numpy: Run the command pip install numpy to install the Numerical Python (NumPy) library. Step 3: Import Required Libraries and Data 1. Open a Python editor or IDE (e.g., PyCharm, Atom, Jupyter Notebook). 2. Import the required libraries:] python import nltk from nltk.chat.util import Chat, reflections 3. Download necessary data from NLTK: python nltk.download('punkt') nltk.download('averaged_perceptron_tagger') nltk.download('wordnet') Step 4: Define Chatbot Responses 1. Define a list of patterns and responses that the chatbot will recognize: python pairs = [[r"my name is (.*)", ["Hello %1, How are you today?"]], [r"hi|hey|hello", ["Hello", "Hey there"]], [r"quit", ["Bye!", "Nice talking to you. Take care!"]] # Add more patterns and responses as needed] Step 5:

Implement Chatbot Interaction 1. Create a new instance of the Chat class using the patterns and responses defined:

```
python
chatbot = Chat(pairs, reflections)
```

2. Implement a loop to interact with the user and chatbot:

```
python while True:
user_input = input("User: ") if user_input.lower() == "quit":
print("Chatbot: Bye!")
break else:
print("Chatbot:", chatbot.respond(user_input))
```

Step 6: Run the Chatbot 1. Save the Python file and run it. 2. Start interacting with the chatbot by typing messages in the terminal. That's it! You've built a basic chatbot using Python. Feel free to add more patterns and responses to enhance the chatbot's capabilities.