PHASE 3: DEVELOPMENT PART 1

TITLE: CREATE A CHATBOT IN PYTHON

Phase 3: Development Part 1

In this part you will begin building your project by loading and preprocessing the dataset. Start building the chatbot by preparing the environment and implementing basic user interactions. Install required libraries, like transformers for GPT-3 integration and flask for web app development.

DATASET: https://www.kaggle.com/datasets/grafstor/simple-dialogs-for-chatbot

ENVIRONMENT: VISUAL STUDIO

Creating a chatbot in Python from scratch is a complex task that typically involves natural language processing (NLP) and machine learning.

Step 1: Install and Import Required Libraries

Install and import the necessary libraries:

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
```

Step 2: Load the Dataset

Assuming a dataset file named 'chatbotdataset.csv' in the same directory as your Python script, load it using pandas:

```
dataset_file = "chatbotdataset.csv"
# Replace with the path to your CSV file
data = pd.read_csv(dataset_file)
```

Step 3: Explore the Dataset and Processing

Taking a quick look at your dataset to understand its structure and see if there are any missing values:

```
# Display the first few rows of the dataset
print(df.head())
# Check for missing values
print(df.isnull().sum())
```

Step 4: Handle Missing Data

Choose and handle missing data by removing rows containing missing values or imputing them with default values. For example, impute missing values with the mean of the respective column:

```
# Impute missing numerical values with the mean of their column
df.fillna(df.mean(), inplace=True)
# Impute missing numerical values with the mean of their column
df.fillna(df.mean(), inplace=True)
```

```
# Preprocessing and Tokenization
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    # Join tokens back to form a clean sentence
    cleaned_text = " ".join(tokens)
    return cleaned_text
# Apply preprocessing to the 'text' column
data['text'] = data['text'].apply(preprocess_text)
# Now 'data' contains the preprocessed and tokenized text in the 'text' column
```

Step 5: Data Cleaning (Optional)

If dataset contains any specific data cleaning tasks, you can perform them at this stage. This might include removing duplicates, handling outliers, or correcting inconsistent data.

Step 6: Save the Preprocessed Data (Optional)

If you want to save the preprocessed data to a new CSV file, you can use the following code:

```
# Save the preprocessed data to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)
```

PROGRAM:

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
dataset_file = "chatbotdataset.csv"
# Replace with the path to your CSV file
data = pd.read_csv(dataset_file)
# Display the first few rows of the dataset
print(df.head())
# Check for missing values
print(df.isnull().sum())
# Impute missing numerical values with the mean of their column
df.fillna(df.mean(), inplace=True)
```

```
# Impute missing numerical values with the mean of their column
df.fillna(df.mean(), inplace=True)
# Save the preprocessed data to a new CSV file
df.to_csv('preprocessed_dataset.csv', index=False)

# Preprocessing and Tokenization
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove punctuation
    text = text.translate(str.maketrans('', '', string.punctuation))
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    # Join tokens back to form a clean sentence
    cleaned_text = " ".join(tokens)
    return cleaned_text
# Apply preprocessing to the 'text' column
data['text'] = data['text'].apply(preprocess_text)
# Now 'data' contains the preprocessed and tokenized text in the 'text' column
```