**CREATE A CHATBOT IN PYTHON**

* **Introduction:**

“A chatbot in Python is a program that uses natural language processing and machine learning techniques to simulate human conversation. It interprets user input, processes it, and generates appropriate responses. Python provides various libraries such as NLTK, spaCy, and TensorFlow that can be used to develop chatbots capable of understanding and generating human-like text-based interactions.”

* **Objectives:**

1. Creating a chatbot in Python can serve various objectives, depending on the specific purpose of the chatbot. Some common objectives for creating a chatbot in Python include:
2. Automating Customer Support: Build a chatbot to handle customer queries, providing instant assistance and information, thereby reducing the burden on human support agents.
3. Enhancing User Engagement: Develop a chatbot to engage users in conversations, provide information, and entertain them, thereby enhancing the overall user experience.
4. Streamlining Tasks: Create a chatbot to automate repetitive tasks, such as data entry, scheduling, or information retrieval, thus improving efficiency and productivity.
5. Gathering User Data: Utilize a chatbot to gather user feedback, preferences, and data, enabling businesses to understand their audience better and tailor their services accordingly.
6. Providing Information and Services: Develop a chatbot to deliver specific information, such as weather updates, news, or product recommendations, thus offering quick and accessible services to users.

By defining clear objectives, you can design a chatbot that effectively fulfills its intended purpose and adds value to the target users or businesses.

* **Dataset:**

Step 1: Install

“ Pip install nltk “

Step 2: Code implementation

“Import nltk”

From nltk.chat.util import Chat, reflections

Pairs = [

[

R”my name is (.\*)”,

[“Hello %1, How are you today?”,]

],

[

R”hi|hey|hello”,

[“Hello”, “Hey there”,]

],

[

R”(.\*) age?”,

[“I’m a computer program”,]

],

[

R”(.\*) (location|city) ?”,

[‘Tokyo, Japan’,]

],

[

R”how is the weather in (.\*)?”,

[“The weather in %1 is amazing like always”,]

],

[

R”quit”,

[“Bye, take care. See you soon ☺ “, “It was nice talking to you. See you later”]

],

]

Def chatbot():

Print(“Hi, I’m your chatbot! How can I help you today?”)

Chat = Chat(pairs, reflections)

Chat.converse()

If \_\_name\_\_ == “\_\_main\_\_”:

Nltk.download(‘punkt’)

Chatbot()

* **Installing libraries**

Sure, to create a simple chatbot in Python, you can use the ChatterBot library. First, you’ll need to install the required libraries. Here is an example of how to create a basic chatbot:

# Install the necessary libraries

!pip install chatterbot

!pip install chatterbot\_corpus

# Import necessary modules

From chatterbot import ChatBot

From chatterbot.trainers import ChatterBotCorpusTrainer

# Create a new chatbot

Bot = ChatBot(‘SimpleBot’)

# Set up the trainer

Trainer = ChatterBotCorpusTrainer(bot)

# Train the chatbot based on the English corpus

Trainer.train(‘chatterbot.corpus.english’)

# Start a conversation with the chatbot

While True:

User\_input = input(“You: “)

If user\_input.lower() == ‘exit’:

Break

Response = bot.get\_response(user\_input)

Print(f”Bot: {response}”)

You can run this code in a Python environment to create a simple chatbot. Make sure to run the pip install commands in your Python environment before running the code.

* **Importing data**

Creating a basic chatbot in Python can be achieved using various libraries. Here is an example using the ChatterBot library. First, you’ll need to install the library using pip:

Pip install chatterbot

Then, you can create a simple chatbot as shown below:

From chatterbot import ChatBot

From chatterbot.trainers import ChatterBotCorpusTrainer

# Create a chatbot instance

Chatbot = ChatBot(‘SimpleBot’)

# Set up a trainer

Trainer = ChatterBotCorpusTrainer(chatbot)

# Train the chatbot on the English language

Trainer.train(‘chatterbot.corpus.english’)

# Start a conversation

While True:

User\_input = input(“You: “)

If user\_input.lower() == ‘exit’:

Break

Response = chatbot.get\_response(user\_input)

Print(f”Bot: {response}”)

This chatbot will respond to the user based on the training data from the ChatterBot library. You can further customize the training data and the bot’s behavior according to your requirements.

* **Missing value analysis:**

Sure, missing value analysis is an important step in data preprocessing. Below is a basic example of how you can perform missing value analysis using Python. Make sure you have the necessary libraries installed by using pip install pandas nump**y.**

Import pandas as pd

Import numpy as np

# Creating a sample dataframe

Data = {‘A’: [1, 2, np.nan, 4, 5],

‘B’: [np.nan, 2, 3, 4, 5],

‘C’: [1, 2, 3, 4, np.nan]}

Df = pd.DataFrame(data)

# Checking for missing values

Missing\_values = df.isnull().sum()

Print(“Missing Values in each column:”)

Print(missing\_values)

# Calculating percentage of missing values

Total\_cells = np.product(df.shape)

Total\_missing = missing\_values.sum()

Percent\_missing = (total\_missing/total\_cells) \* 100

Print(f”Percentage of missing values: {percent\_missing}%”)

This code snippet will help you analyze missing values in your dataset and provide you with the count of missing values for each column along with the percentage of missing values in the dataset.

* **Exploratory data analysis:**

Certainly! Exploratory Data Analysis (EDA) is an essential part of any data analysis process. Here’s a simple chatbot example in Python that can guide you through an EDA process using the popular libraries pandas and matplotlib.

Firstly, ensure you have these libraries installed. If not, install them using pip:

Pip install pandas matplotlib

Import pandas as pd

Import matplotlib.pyplot as plt

Class EDAChatbot:

Def \_\_init\_\_(self, data\_path):

Self.data = pd.read\_csv(data\_path) # Assuming you have a CSV file

Def summary(self):

Return self.data.describe()

Def show\_head(self, n=5):

Return self.data.head(n)

Def show\_tail(self, n=5):

Return self.data.tail(n)

Def plot\_histogram(self, column\_name):

Self.data[column\_name].plot(kind=’hist’)

Plt.title(f’Histogram for {column\_name}’)

Plt.show()

This script provides a basic structure for a Python-based EDA chatbot. You can extend it further with more features and analysis functions depending on your specific use case.

* **Outlier detection:**

# Import necessary libraries

Import numpy as np

From sklearn.ensemble import IsolationForest

# Generate random data for demonstration purposes

Np.random.seed(1)

X = 0.3 \* np.random.randn(100, 2)

X\_outliers = np.random.uniform(low=-4, high=4, size=(20, 2))

X = np.r\_[X + 2, X – 2, X\_outliers]

# Fit the Isolation Forest model

Clf = IsolationForest(max\_samples=100, random\_state=42)

Clf.fit(X)

# Define a function to detect outliers

Def detect\_outliers(chat\_input):

Try:

Values = list(map(float, chat\_input.split()))

Prediction = clf.predict([values])

If prediction[0] == 1:

Return “The data point does not seem to be an outlier.”

Else:

Return “The data point seems to be an outlier.”

Except ValueError:

Return “Please enter valid numerical values.”

# Test the outlier detection function

Print(detect\_outliers(“1.5 2.5”))

Print(detect\_outliers(“10 10”))

* **Feature engineering:**

Certainly! Creating a chatbot involves several key steps, including natural language processing (NLP) techniques and feature engineering. Here’s a simple example of how to set up a basic chatbot in Python with feature engineering:

Import nltk

Import numpy as np

Import random

Import string

From sklearn.feature\_extraction.text import TfidfVectorizer

From sklearn.metrics.pairwise import cosine\_similarity

# Sample corpus for the chatbot

Corpus = [

‘Hello!’,

‘How are you?’,

‘What is your name?’,

‘What is the weather like today?’,

‘Tell me a joke.’,

‘Bye!’

]

# Tokenization

Nltk.download(‘punkt’)

Nltk.download(‘wordnet’)

Sent\_tokens = nltk.sent\_tokenize(‘. ‘.join(corpus))

Word\_tokens = nltk.word\_tokenize(‘. ‘.join(corpus))

# Preprocessing and normalization

Lemmer = nltk.stem.WordNetLemmatizer()

Def LemTokens(tokens):

Return [lemmer.lemmatize(token) for token in tokens]

Remove\_punct\_dict = dict((ord(punct), None) for punct in string.punctuation)

Def LemNormalize(text):

Return LemTokens(nltk.word\_tokenize(text.lower().translate(remove\_punct\_dict)))

# Generating response

Def response(user\_response):

Robo\_response = ‘’

Sent\_tokens.append(user\_response)

# TF-IDF vectorization

TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop\_words=’english’)

Tfidf = TfidfVec.fit\_transform(sent\_tokens)

# Calculate similarity

Vals = cosine\_similarity(tfidf[-1], tfidf)

Idx = vals.argsort()[0][-2]

Flat = vals.flatten()

Flat.sort()

Req\_tfidf = flat[-2]

If req\_tfidf == 0:

Robo\_response = “I’m sorry! I don’t understand you”

Return robo\_response

Else:

Robo\_response = sent\_tokens[idx]

Return robo\_response

# Chat function

Def chat():

Print(“Hello! I am your chatbot. You can start chatting with me. If you want to exit, just type ‘Bye!’”)

While True:

User\_response = input()

User\_response = user\_response.lower()

If user\_response != ‘bye’:

If user\_response == ‘thanks’ or user\_response == ‘thank you’:

Print(“You’re welcome!”)

Else:

If response(user\_response) != “I’m sorry! I don’t understand you”:

Print(response(user\_response))

Else:

Print(“I’m sorry! I don’t understand you”)

Else:

Print(“Bye! Have a great day!”)

Break

# Start the chat

Chat()

This is a basic framework for a chatbot using simple feature engineering with the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm. You can expand this code by integrating more complex NLP models or adding more sophisticated features.

* **Transforming data:**

Certainly! To create a simple chatbot in Python, you can use the transformers library to handle the natural language processing. First, make sure you have the necessary libraries installed. You can install the required packages using pip:

Pip install transformers

Pip install torch

Here’s a basic example of how you can create a simple chatbot using the Hugging Face transformers library:

From transformers import pipeline

# Load the pre-trained model

Chatbot = pipeline(“conversational”)

# Chat with the bot

Print(“You can start chatting with the bot. Type ‘exit’ to end the conversation.”)

While True:

User\_input = input(“User: “)

If user\_input.lower() == ‘exit’:

Break

Bot\_response = chatbot(user\_input)[0][‘generated\_text’]

Print(f”ChatBot: {bot\_response}”)

This code snippet creates a simple chatbot that you can interact with in the command line. Make sure you run this script in a Python environment with the required libraries installed.

Remember that this is a simple demonstration. For more complex chatbot scenarios, you might need to fine-tune a pre-trained model or use a more sophisticated architecture.

* **Data split:**

Import nltk

From nltk.chat.util import Chat, reflections

Pairs = [

[

R”my name is (.\*)”,

[“Hello %1, how are you today?”,]

],

[

R”what is your name?”,

[“My name is ChatGPT, nice to meet you ☺”,]

],

[

R”how are you?”,

[“I’m doing good, how about you?”,]

],

[

R”sorry (.\*)”,

[“No need to apologize, it’s all good.”,]

],

[

R”(.\*) (hungry|thirsty|tired)”,

[“I can understand. Please have some snacks.”,]

],

]

Def chatbot():

Print(“Hi! I’m a simple chatbot. Ask me anything or just say hi ☺”)

Chat = Chat(pairs, reflections)

Chat.converse()

If \_\_name\_\_ == “\_\_main\_\_”:

Chatbot()

Make sure you have the NLTK library installed. You can install it using pip:

Pip install nltk

* **Cross validate models:**

Sure, here’s an example of how you can create a simple chatbot in Python that cross-validates models. For this example, I’ll use the scikit-learn library for model building and cross-validation:

From sklearn import datasets

From sklearn.model\_selection import train\_test\_split

From sklearn.linear\_model import LogisticRegression

From sklearn.tree import DecisionTreeClassifier

From sklearn.model\_selection import cross\_val\_score

Def load\_data():

Iris = datasets.load\_iris()

X = iris.data

Y = iris.target

Return X, y

Def build\_models():

Models = [

(‘Logistic Regression’, LogisticRegression()),

(‘Decision Tree’, DecisionTreeClassifier())

]

Return models

Def cross\_validate(models, X, y):

For name, model in models:

Scores = cross\_val\_score(model, X, y, cv=5)

Print(f’{name} cross-validation accuracy: {scores.mean():.2f}’)

If \_\_name\_\_ == ‘\_\_main\_\_’:

X, y = load\_data()

Models = build\_models()

Cross\_validate(models, X, y)

You can run this script to cross-validate the performance of the logistic regression and decision tree models on the Iris dataset. You can also extend this script by including more models and datasets to perform cross-validation on a wide variety of machine learning models.

* **Hyper parameter tuning:**

Sure, I can guide you through the process of creating a simple chatbot in Python and how you can perform hyperparameter tuning. First, you’ll need to install the necessary libraries. Run the following commands in your Python environment:

Pip install numpy

Pip install tensorflow

Pip install tflearn

Here’s a simple example of how you can create a basic chatbot and perform hyperparameter tuning:

Import numpy as np

Import tensorflow as tf

Import tflearn

Import random

# Data representation

Training\_data = […] # Your training data here

Training\_labels = […] # Your training labels here

# Building the neural network

Net = tflearn.input\_data(shape=[None, len(training\_data[0])])

Net = tflearn.fully\_connected(net, 8)

Net = tflearn.fully\_connected(net, 8)

Net = tflearn.fully\_connected(net, len(training\_labels[0]), activation=’softmax’)

Net = tflearn.regression(net)

# Define model and setup tensorboard

Model = tflearn.DNN(net, tensorboard\_dir=’tflearn\_logs’)

# Start training (apply hyperparameter tuning here)

Model.fit(training\_data, training\_labels, n\_epoch=1000, batch\_size=8, show\_metric=True)

# Save the model

Model.save(‘model.tflearn’)

To perform hyperparameter tuning, you can tweak the values of the n\_epoch (number of epochs) and batch\_size parameters in the model.fit function. You can also experiment with different values for the fully\_connected layers and other parameters in the neural network architecture.

* **Prediction:**

To create a simple chatbot in Python, you can use the ChatterBot library. Here is a simple example of how you can create a basic chatbot in Python:

First, you will need to install the ChatterBot library if you haven’t already. You can do this using pip:

From chatterbot import ChatBot

From chatterbot.trainers import ChatterBotCorpusTrainer

# Create a new chat bot named prediction\_bot

Prediction\_bot = ChatBot(‘PredictionBot’)

# Create a new trainer for the chat bot

Trainer = ChatterBotCorpusTrainer(prediction\_bot)

# Train the chat bot based on the English corpus

Trainer.train(‘chatterbot.corpus.english’)

# Get a response from the chat bot

Response = prediction\_bot.get\_response(‘Hello, how are you?’)

# Print the response

Print(response)