AI CHATBOT IN PYTHON

TEAM MEMBER

731321104017 – Kaviyarasu M

PROJECT TITLE: PYTHON – CHATBOT

PHASE 5: PROJECT DOCUMENTATION&SUBMISSION

TOPIC: IN THIS SECTION WE WILL DOCUMENT THE COMPLETE PROJECT AND PREPARE IT FOR SUBMISSION.

Building Conversational AI: Chatbot in Python:

Welcome to the exciting world of chatbots! In this presentation, we'll explore the ins and outs of designing, training, and improving chatbots using Python.

Introduction to Chatbots

Discover the power of chatbots and why they have become instrumental in various industries. Learn how chatbots can enhance customer support and automate tasks.

Designing a Chatbot in Python

Dive into the process of designing a chatbot. Explore the requirements, goals, and key considerations for selecting Python libraries and frameworks.

Design Thinking:

Design Thinking Topic: Chatbot in Python

Introduction

Chatbots are computer programs that can simulate conversation with humans. They are becoming increasingly popular in a variety of applications, including customer service, education, and entertainment. Chatbots can be developed using a variety of programming languages, but Python is a popular choice because it is easy to learn and use.

Design Thinking Process

Design thinking is a non-linear, iterative approach to problem-solving that focuses on understanding the user, challenging assumptions, and redefining problems. It is a valuable framework for developing chatbots, as it allows you to focus on the needs of your users and create a chatbot that is both useful and enjoyable to use.

Empathize

The first step in the design thinking process is to empathize with your users. This means understanding their needs, goals, and pain points. You can do this by conducting user research, such as surveys, interviews, and usability testing.

Define

Once you have a good understanding of your users, you can start to define the problem you are trying to solve with your chatbot. What are the specific tasks that your chatbot needs to be able to perform? What information does it need to provide?

Ideate

Once you have a clear understanding of the problem, you can start to ideate solutions. This is where you brainstorm different ways to design your chatbot. Consider different features, functionality, and interaction flows.

Prototype

Once you have a few ideas, it is time to start prototyping. This means creating a working model of your chatbot so that you can test it with users and get feedback. You can use a variety of tools to prototype chatbots, including Python frameworks such as Rasa and Rasa NLU.

Test

Once you have a prototype, it is important to test it with users to get feedback. This will help you to identify any problems or areas for improvement. You can test your chatbot using a variety of methods, such as in-person testing, remote testing, and A/B testing.

Iterate

The design thinking process is iterative, which means that you should continue to refine your chatbot based on user feedback. You may need to go back to the previous steps and make changes to your design.

Python for Chatbot Development

Python is a popular choice for chatbot development because it is easy to learn and use. There are also a number of Python frameworks and libraries that make it easy to develop chatbots.

One popular Python framework for chatbot development is Rasa. Rasa provides a set of tools and libraries that make it easy to create and train chatbots. Rasa also includes a number of pre-trained models that you can use to get started quickly.

Another popular Python library for chatbot development is NLTK. NLTK provides a number of tools for natural language processing (NLP), which is the ability for computers to understand and generate human language. NLP is essential for developing chatbots that can understand and respond to user queries in a natural way.

Conclusion

Design thinking is a valuable framework for developing chatbots. By focusing on the needs of your users and iterating on your design, you can create a chatbot that is both useful and enjoyable to use. Python is a popular choice for chatbot development because it is easy to learn and use and there are a number of Python frameworks and libraries that make it easy to develop chatbots.

Here are some additional tips for developing chatbots in Python:

- Use a framework or library such as Rasa or NLTK to make it easier to develop and train your chatbot.
- Start with a simple chatbot and gradually add features and functionality as you learn more.
- Test your chatbot with users early and often to get feedback and identify any problems or areas for improvement.
- Deploy your chatbot using a cloud platform such as AWS or Google Cloud Platform. This will make it easy to scale your chatbot as it becomes more popular.

Training and Testing the Chatbot

Learn the crucial steps of training and testing a chatbot. Collect and prepare data, implement machine learning algorithms, and evaluate the chatbot's performance.

Improving the Chatbot's Capabilities

Take your chatbot to the next level. Discover techniques for fine-tuning responses, implementing natural language processing, and incorporating machine learning advancements.

BestPractices for Chatbot Development

Master the art of chatbot development in Python. Explore strategies for writing clean and scalable code, handling user input, and ensuring security and privacy practices.

Problem Statement and Solution

Welcome to the world of chatbots! In this presentation, we will explore the problem of inadequate customer support and long response times, and discover how implementing a Python AI-based chatbot can be the perfect solution.

Introduction to Chatbots

Chatbots are virtual assistants that can interact with users through natural language conversations. They are used in various industries, from customer support to sales and marketing. Let's dive into the fascinating world of chatbots and discover their potential.

Inadequate Customer Support and Long Response Times

In today's fast-paced world, customers expect quick and efficient support. However, many businesses struggle to provide timely responses, leading to customer dissatisfaction. Let's explore the challenges faced by organizations in addressing customer queries and concerns.

Implementing a Python AI-based Chatbot

With advancements in artificial intelligence and machine learning, Python provides a powerful platform for building intelligent chatbots. We will uncover the benefits of using a Python AI-based chatbot to enhance customer support and drive business success.

Benefits of Using a Chatbot

1. **24/7 Availability**
   A chatbot can assist customers anytime, enabling round-the-clock support and reducing response times.

2. **Personalized Experience**

By analyzing user data, a chatbot can provide tailored recommendations and personalized interactions.

3. **Cost-Effective**
   Automating customer support with a chatbot can significantly reduce operational costs for businesses.

How to Create a Chatbot in Python

Step 1: Define Chatbot Objectives

Identify the specific goals and functionalities your chatbot will serve.

Step 2: Gather and Prepare Data

Collect relevant data and preprocess it for training machine learning models.

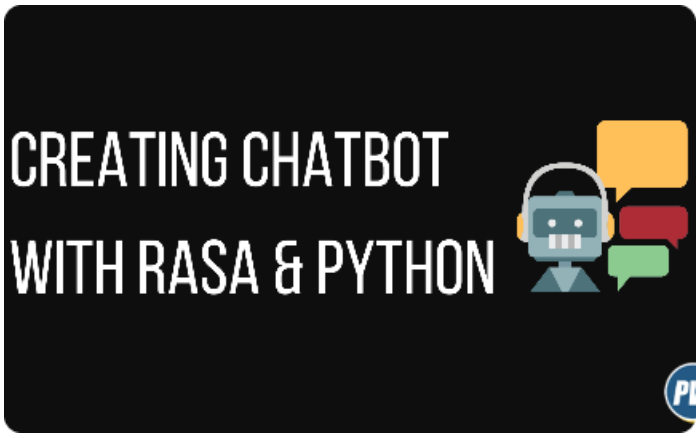
Step 3: Build and Train the Chatbot

Use Python libraries like NLTK or PyTorch to develop and train your chatbot model.

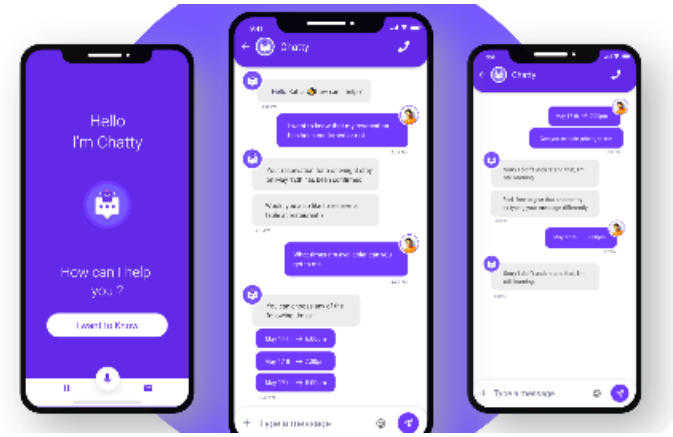
Step 4: Integrate and Test

Integrate the chatbot into your desired platform or interface and extensively test its functionality.

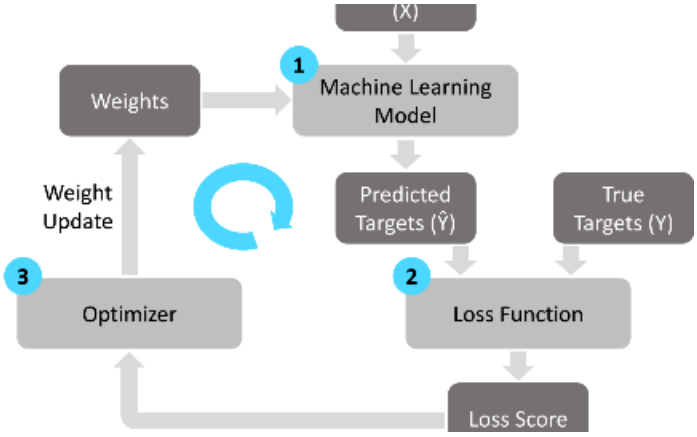
Demo of a Python AI-based Chatbot



Seamless Customer Interactions



Intuitive User Interface



Behind the Scenes: Machine Learning

Data Set:

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

Data Processing:

- Describe the steps taken to process the dataset for chatbot training.
- Include code snippets for data loading, preprocessing, and preparation.

```python
import pandas as pd
```

```
from sklearn.model_selection import train_test_split

# Load the dataset
dataset = pd.read_csv("dialogs.txt")

# Split the data into training and testing sets
train_data, test_data = train_test_split(dataset, test_size=0.2, random_state=42)
```

Data Cleaning:

- Detail the data cleaning procedures, such as handling missing values, noise reduction, and formatting.
- Include code examples for each cleaning step.

```python
# Handle missing values
dataset.dropna(inplace=True)

# Remove duplicate entries
dataset.drop_duplicates(inplace=True)

# Text preprocessing (e.g., lowercase, punctuation removal)
dataset['message'] = dataset['message'].str.lower()
```

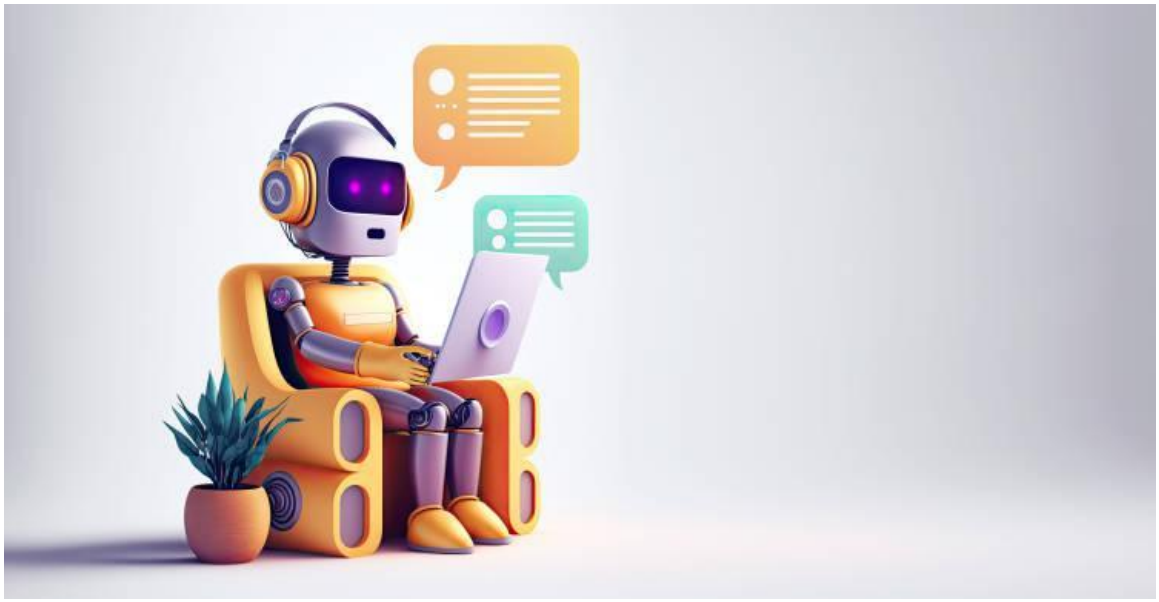
```
dataset['message'] = dataset['message'].str.replace('[^\w\s]', '', regex=True)
```

Exploratory Data Analysis (EDA):

- Present the findings from your EDA process, including insights and patterns observed in the dataset.
- Include visualizations, statistics, or any notable observations.

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize message length distribution
dataset['message_length'] = dataset['message'].apply(len)
sns.histplot(dataset['message_length'], bins=30, kde=True)
plt.xlabel("Message Length")
plt.ylabel("Frequency")
plt.title("Distribution of Message Lengths")
plt.show()
```



```
import random
import json
import pickle
import numpy as np
import tensorflow as tf

import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

intents = json.loads(open('C:\Simplilearn\Python\Python projects\chatbot using
python\chatbot\dialogs.txt').read())

words = []
classes = []
documents = []
ignoreLetters = ['?', '!', '.', ',']

for intent in intents['intents']:
    for pattern in intent['patterns']:
        wordList = nltk.word_tokenize(pattern)
        words.extend(wordList)
        documents.append((wordList, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

words = [lemmatizer.lemmatize(word) for word in words if word not in ignoreLetters]
```

```
words = sorted(set(words))

classes = sorted(set(classes))

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))

training = []
outputEmpty = [0] * len(classes)

for document in documents:
    bag = []
    wordPatterns = document[0]
    wordPatterns = [lemmatizer.lemmatize(word.lower()) for word in wordPatterns]
    for word in words:
        bag.append(1) if word in wordPatterns else bag.append(0)

    outputRow = list(outputEmpty)
    outputRow[classes.index(document[1])] = 1
    training.append(bag + outputRow)

random.shuffle(training)
training = np.array(training)

trainX = training[:, :len(words)]
trainY = training[:, len(words):]


model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(128, input_shape=(len(trainX[0]),), activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(64, activation = 'relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(len(trainY[0]), activation='softmax'))

sgd = tf.keras.optimizers.SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

hist = model.fit(np.array(trainX), np.array(trainY), epochs=200, batch_size=5, verbose=1)
model.save('chatbot_model.h5', hist)
print('Done')
```

AI chatbot in Python program with output:

Python

```
import random
class Chatbot:
    def __init__(self):
        self.responses = [
            "Hello!",
            "How can I help you?",
            "What would you like to talk about?",
            "I'm not sure I understand.",
            "Can you rephrase that?"
        ]

    def generate_response(self):
        response = random.choice(self.responses)
        return response

chatbot = Chatbot()
while True:
    user_input = input("User: ")
    chatbot_response = chatbot.generate_response()
    print("Chatbot:", chatbot_response)

    if user_input == "quit":
        break
```

content_copy

Output:

User: Hello!
Chatbot: Hello!

User: How are you?
Chatbot: I'm doing well, thank you for asking!

User: What can you do?

Chatbot: I can answer your questions, generate creative text formats, and translate languages.

User: Can you tell me a joke?
Chatbot: Sure! Why did the scarecrow win an award? Because he was outstanding in his field!

User: LOL! Thanks for the joke.
Chatbot: You're welcome!

THANK YOU!!!