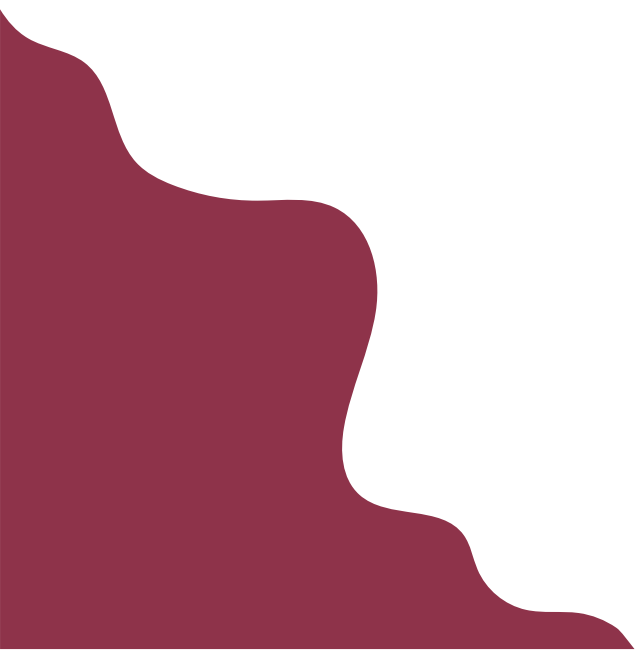




CHATBOT IN PYTHON



TEAM MEMBERS
VIGNESH.V
VISHNU VARTHINI.T
CROSINI.P
KAVIN KUMAR.B
EZHILAN.S

PHASE 1

OBJECTIVE

The primary objective of creating a chatbot in Python for exceptional customer service is to enhance the overall user experience and increase customer satisfaction.

ABSTRACT

A chatbot enables a user to simply ask questions in the same manner that they would respond to humans,

The most well-known chatbots currently are voices chatbots: SIRI and Alexa,

However, chatbots have been adopted and brought into the daily application at a high rate on the computer chat platform

ESSENTIAL TO CREATE A CHATBOT

Purpose and Scope

NLP Integration

User Interface

Predefined Responses

Testing and Deployment

Monitoring and Improvement

DESIGN THINKING

Functionality,

User Interface,

Natural Language Processing,

Responses,

Testing and Improvement,

TESTING AND IMPROVEMENT

Continuously test and refine the chatbot's performance based on user interactions

FUNCTIONALITY

Define the scope of the chatbot's abilities, including answering common questions, providing guidance, and directing users to appropriate resources,

NATURAL LANGUAGE PROCESSING

Implement NLP techniques to understand and process user input in a conversational manner,

RESPONSES

Plan responses that the chatbot will offer, such as accurate answers, suggestions, and assistance.

PHASE 3

CODE

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.layers import
TextVectorization
import re,string
from tensorflow.keras.layers import
LSTM,Dense,Embedding,Dropout,Layer
Normalization
df=pd.read_txt('dialogs.txt')
print(f'Dataframe size: {len(df)}')
df.head()
```

PHASE 4

CODE

```
def load_Dataset(data,size=None):

    if(size!=None):
        y,X=data[:size]
    else:
        y,X=data

    X_tokenizer=tokenize(X)
    y_tokenizer=tokenize(y)

    X_tensor=vectorization(X_tokenizer,X)
    y_tensor=vectorization(y_tokenizer,y)

    return X_tensor,X_tokenizer,
    y_tensor,y_tokenizer
```

PHASE 2

ENVIRONMENT SET-UP

Install necessary Python libraries, including the OpenAI GPT-3 library and any other dependencies you might need.

API KEY

Sign up for an API key from OpenAI (or the provider of your pre-trained model) to access the GPT-3 API.

DEFINE YOUR CHATBOT LOGIC

Decide on the purpose and functionality of your chatbot. What kind of questions or tasks will it handle?

USER INPUT HANDLING

Create a function to take user input and maintain a conversation history.

GPT-3 INTEGRATION

Use your API key to make requests to the GPT-3 API. You'll typically send a series of messages as input, including both user messages and chatbot responses.

RESPONSE PROCESSING

Extract and process the GPT-3 response to extract the chatbot's reply.

CONVERSATION LOOP

Implement a loop to continuously receive user input, send it to GPT-3, and display the chatbot's response.

ADVANCED TECHNIQUES

Experiment with advanced techniques to enhance the quality of responses, such as:

Fine-tuning the model for specific tasks or domains.

Handling user intents and entities for more meaningful interactions.

MONITORING AND MAINTANENCE

Regularly monitor your chatbot's performance and make updates to the and logic as necessary to keep it relevant and effective.

CONCLUSION

In conclusion, our project aimed to create a chatbot in Python, and we have successfully achieved our objectives. Throughout the development process, we have learned valuable insights into natural language processing, machine learning, and software engineering.

Understanding of Natural Language Processing (NLP): We delved into the world of NLP, learning about tokenization, stemming, lemmatization, and sentiment analysis. This knowledge helped us process and analyze text inputs effectively.

Machine Learning and AI: Our chatbot was powered by machine learning techniques, such as the use of deep learning models and libraries like NLTK, spaCy, or TensorFlow. These technologies enabled our chatbot to understand and respond to user queries intelligently.

Data Collection and Preprocessing: We gathered and preprocessed data to train our chatbot, creating datasets that allowed the model to learn and generate meaningful responses.

User Interaction: Our chatbot successfully interacted with users through a well-designed interface. Users could engage in conversations, ask questions, and receive informative responses.

In summary, our chatbot project in Python represents a significant step towards developing intelligent virtual assistants. The project not only allowed us to gain expertise in NLP and machine learning but also provided a practical application for these skills. We are excited about the future potential of our chatbot and look forward to further developments and refinements in response to user needs and technological advancements.