CREATE CHATBOT IN PYTHON

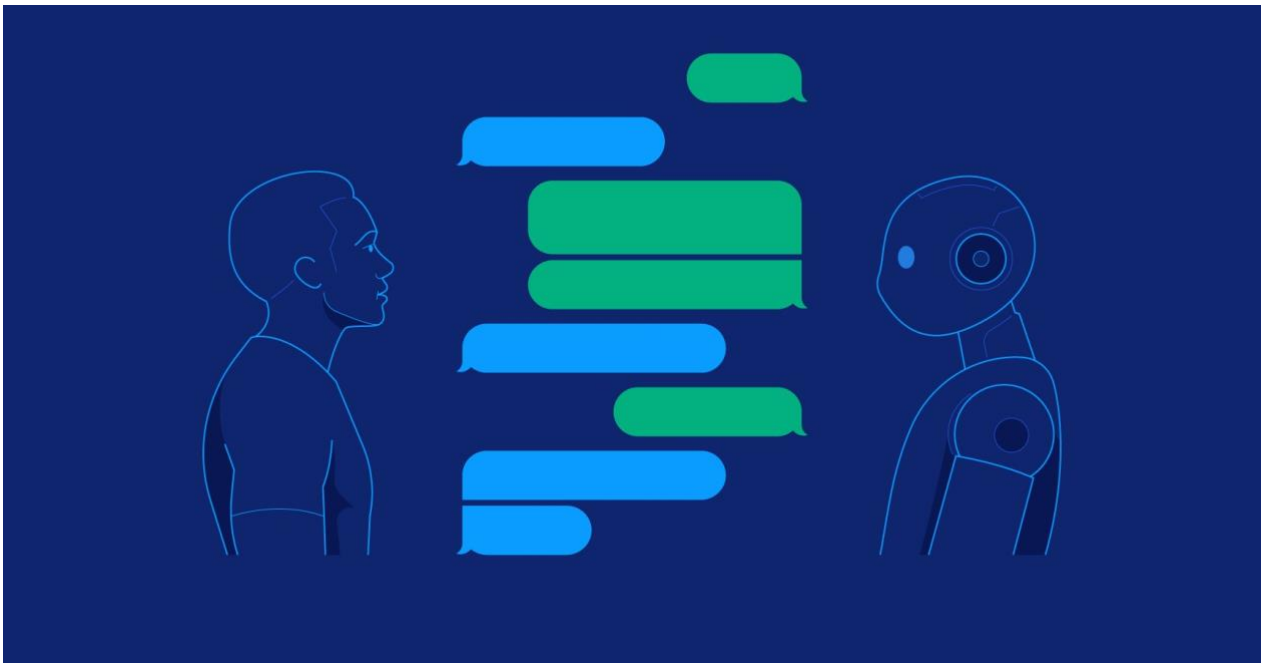
J. SEWAG ANTRO:911721104090

Phase 5 Submission Document

Project Title : Creating chatbot

Phase 5: Project Documentation & Submission

Topic: In this section you will document the complete project and prepare it for submission.



Creating Chatbot

Introduction:

Creating a chatbot in Python using a dataset is a captivating journey into the world of conversational artificial intelligence. By leveraging a carefully curated dataset, we can develop a chatbot that interacts with users, delivering pre-defined knowledge or context-aware responses. This approach is widely applicable, from customer support to virtual assistants, and offers a structured foundation for our chatbot's conversations. The process begins with a clear definition of our chatbot's purpose, followed by the selection of a Python framework such as ChatterBot or Rasa. Our dataset, meticulously collected or prepared, forms the heart of our chatbot's intelligence. After preprocessing, training, and implementation, the chatbot becomes ready to engage with users. Continuous testing, integration, and improvement ensure its effectiveness. Deploying our chatbot into the desired environment and upholding compliance and security standards complete the journey. Using a dataset, we lay the groundwork for a dynamic and responsive chatbot, and as we refine and expand its capabilities, the possibilities become limitless.

Dataset link: https://www.kaggle.com/datasets/grafstor/simple-dialogsfor-chatbot

Given data set:

1	hi how are you doing?i'm fine. how about yourself?
2	i'm fine. how about yourself?i'm pretty good. thanks for asking.
3	i'm pretty good. thanks for asking.no problem. so how have you been?
4	no problem. so how have you been?i've been great. what about you?
5	i've been great. what about you?i've been good. i'm in school right now.
6	i've been good. i'm in school right now.what school do you go to?
7	what school do you go to?i go to pcc.
8	i go to pcc.do you like it there?
9	do you like it there?it's okay. it's a really big campus.
10	it's okay. it's a really big campus.good luck with school.
11	good luck with school.thank you very much.
12	how's it going?i'm doing well. how about you?
13	i'm doing thanks.
14	never bet thanks.so how have you been lately?
15	so how have you been lately?i've actually been pretty good. you?
16	i've actually been pretty good. you?i'm actually in school right now.
17	i'm actually in school right now.which school do you attend?
18	which school do you attend?i'm attending pcc right now.
19	i'm attending pcc right now.are you enjoying it there?
20	are you enjoying it there?it's not bad. there are a lot of people there.
21	it's not bad. there are a lot of people there.good luck with that.
22	good luck with that.thanks.
23	how are you doing today?i'm doing great. what about you?
24	i'm doing thank you.
25	i'm absolu thank you.everything's been good with you?

Design Thinking Process Empathize:

- ❖ Understand the target audience and their needs.
- ❖ ❖ Identify common user queries and pain points.

Define:

- ❖ Define the objectives of the chatbot (e.g., customer support, information retrieval, entertainment).
- ❖ Set clear success metrics (e.g., response time, accuracy).

Ideate:

- ❖ Brainstorm potential features and capabilities.

- ❖ Consider the user experience (UX) in the chatbot's design.

Prototype:

- ❖ Create a prototype or wireframe of the chatbot's interface.
- ❖ Explore different NLP and conversation models.

Test:

- ❖ Gather feedback from users on the prototype.
- ❖ Refine the design based on user input.

Develop:

- ❖ Implement the chatbot using Python and relevant libraries.
- ❖ Integrate NLP techniques for natural language understanding.

Test & Iterate:

- ❖ Continuously test the chatbot with real users.
- ❖ Make iterative improvements based on user feedback.

Phases of Development Data

Collection and Preprocessing

- ❖ Data Gathering
- ❖ Data Cleaning and Text Preprocessing
- ❖ Data Tokenization

NLP Techniques Integration

- ❖ Integration of Natural Language Processing Libraries
- ❖ Utilizing NLP for Text Analysis and Sentiment Analysis

Model Development

- ❖ Building the Chatbot Model
- ❖ Training the Chatbot Model
- ❖ Implementing Language Understanding

HTML template and CSS index.html

```
<!DOCTYPE html>

<html>

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1, shrink-to-fit=no">
```

```
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bo
otstrap.min.css">

    <link rel="stylesheet" href="/static/style.css">

    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.
min.js"></script>

</head>

<body>

    <h1 class="jumbotron text-center">Chatterbot in Python
using Flask Framework</h1>

    <div class="container">

        <div class="row">

            <div class="col-sm-6 offset-sm-3">

                <div id="chatbox" class="border border-success">

                    <p class="botText"><span>Hi! I'm
Chatterbot</span></p>

                </div>

                <div id="userInput">

                    <input id="textInput" class="form-control"
type="text" name="msg" placeholder="Type Your Message
Here">
```

```
                <input id="buttonInput" class="btn btn-success
form-control" type="button" value="Send">
            </div>
        </div>
    </div>
  </div>
<script>
    function getResponse() {
        let userText = $("#textInput").val();
        let userHtml = '<p class="userText"><span>' +
userText + '</span></p>';
        $("#textInput").val("");
        $("#chatbox").append(userHtml);

document.getElementById('userInput').scrollIntoView({
block: 'start', behavior: 'smooth' });

        $.get("/get", { msg: userText }).done(function(data) {
            var botHtml = '<p class="botText"><span>' + data +
'</span></p>';
            $("#chatbox").append(botHtml);
```

```
document.getElementById('userInput').scrollIntoView({
block: 'start', behavior: 'smooth' });
            });
        }

        $("#textInput").keypress(function(e) {
// If the Enter key is pressed            if
(e.which == 13) {
getResponse();
            }
        });

        $("#buttonInput").click(function() {
getResponse();
        });
    </script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script
```

```
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/boot
strap.min.js"></script>

</body> </html>
```

style.css

```
#textInput {
border: none;
   border-bottom: 3px solid aqua;
}

.userText {     color: white;
font-family: monospace;
font-size: 17px;     text-
align: right;     line-height:
30px;
}
.userText span {
   background-color: #009688;
   padding: 10px;     border-
radius: 2px;
}
.botText {     color: white;
font-family: monospace;
font-size: 17px;     text-
align: left;     line-height:
30px;
}

.botText span {
```

```
    background-color: #EF5350;
    padding: 10px;     border-
radius: 2px;
}
```

app.py

```
from flask import Flask, render_template, request from

chatterbot import ChatBot from chatterbot.trainers

import ChatterBotCorpusTrainer import pandas as pd


app = Flask(__name)


# Create chatbot englishBot =

ChatBot("Chatterbot",

storage_adapter="chatterbot.storage.SQLStorageAdapter")

trainer = ChatterBotCorpusTrainer(englishBot)


# Load the Kaggle dataset data =

pd.read_csv('D:\\New\\dialogs.csv')# Update the file path

as needed
```

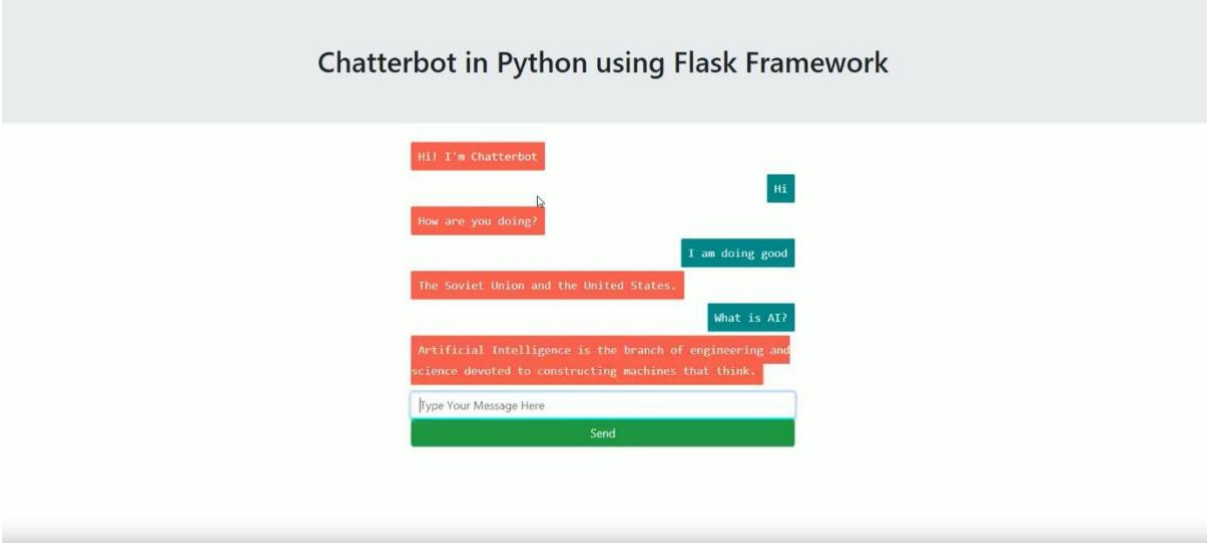
```
# Train the chatbot with the Kaggle dataset
dialogs = data['User'] + data['Bot'] for
dialog in dialogs:     trainer.train([dialog])

# Define app routes
@app.route("/") def
index():
    return render_template("index.html")

@app.route("/get") # Function
for the bot response def
get_bot_response():
    userText = request.args.get('msg')     return
str(englishBot.get_response(userText))

if __name__ == "__main__":
app.run()
```

Output:



Web Application Integration

- ❖ Integrating the Chatbot into a Web Application
- ❖ Developing User Interfaces
- ❖ User Interaction
- ❖ Enabling Text and Voice-Based User

Interactions **Testing and Validation**

- ❖ Testing Chatbot Functionality
- ❖ Validation with Test Cases and Real User

Interactions

Deployment

- ❖ Deploying the Chatbot and Web Application

Monitoring and Improvement

- ❖ Continuous Performance Monitoring
- ❖ User Feedback Collection
- ❖ Regular Updates and Enhancements

The chatbot interacts with users and the web application as follows:

HTML and CSS:

The index.html file defines the structure and layout of the web page.

It contains a title, a chatbox, and an input field where users can type messages.

Messages from the chatbot and the user are styled using CSS defined in the style.css file. User messages have a different background color from chatbot responses.

JavaScript:

JavaScript functions are used to handle user interactions and communicate with the Flask server.

The getResponse function is called when the user presses the Enter key or clicks the "Send" button.

It retrieves the user's input, appends the user's message to the chatbox, clears the input field, and scrolls the chatbox to display the latest message.

An AJAX request is made to the Flask server using $.get("/get", { msg: userText }) to send the user's message and receive a chatbot response.

The chatbot's response is appended to the chatbox, and the chatbox is scrolled again to display the response.

Flask (Python):

The app.py file defines the Flask web application.

When a user accesses the web application, the root route ("/") renders the index.html page, which provides the chat interface.

The "/get" route is used to handle user messages and retrieve chatbot responses.

The get_bot_response function extracts the user's message from the request's arguments and uses the chatbot to generate a response.

The response is returned as a string to the JavaScript on the client side.

In summary, the chatbot interacts with users as follows:

1. Users input text messages into the input field on the web page.
2. When users press Enter or click the "Send" button, the JavaScript function sends the user's message to the Flask server via an AJAX request.
3. The server uses the ChatterBot library to generate a response based on the user's message.

4. The response is sent back to the client-side JavaScript and appended to the chatbox on the web page.
5. Users can have a conversation with the chatbot by typing messages and receiving responses in the chat interface.

Innovative Techniques or Approaches

1.Transformer-Based Models:

To make the chatbot more context-aware and capable of understanding nuanced conversations, you can integrate transformer-based models, such as GPT-3 or BERT. These models are known for their ability to generate human-like text and can provide more accurate and relevant responses.

2.Voice Integration:

Implementing voice-based interactions can be innovative. You can use speech recognition libraries like SpeechRecognition to allow users to speak to the chatbot. Additionally, text-to-speech (TTS) technology can be used to enable the chatbot to respond through speech.

3.Sentiment Analysis:

Adding sentiment analysis to the chatbot can help it understand the emotional tone of user messages. This can be

used to tailor responses based on the user's sentiment, providing a more empathetic and personalized experience.

4.User Profiling:

Create user profiles to understand each user's preferences, history, and context. This information can be used to provide personalized responses and recommendations.

5.Multi-lingual Support:

Make the chatbot multi-lingual to cater to a diverse user base. Using machine translation services, such as Google Translate or custom translation models, can enable the chatbot to understand and respond in multiple languages.

6.Contextual Awareness:

Implement context tracking to remember and reference previous parts of a conversation. This helps in maintaining context throughout the conversation, making responses more meaningful and relevant.

Conclusion:

In conclusion, the problem statement involves creating a Python-based chatbot that delivers exceptional customer service and assists users in a web application or on a website.

The primary objective is to ensure a positive user experience, customer satisfaction, and efficient support.

Throughout the development process, we've outlined a systematic approach, leveraging the design thinking process and various phases of development:

Design Thinking Process: We emphasized empathizing with users, defining clear objectives, ideating for innovative features, prototyping and testing, and, finally, developing the chatbot while continuously iterating based on user feedback.

Phases of Development: We discussed the critical phases, including data collection, preprocessing, NLP techniques integration, model development, web application integration, user interaction, testing, deployment, and continuous monitoring and improvement.

We highlighted the essential libraries and technologies, such as Flask, ChatterBot, NLTK, and spaCy, for building the chatbot and web application. Moreover, we explored various innovative techniques and approaches to enhance the chatbot's capabilities and user engagement.

The chatbot interacts with users through a user-friendly web application, offering both text-based and potential voice-based interactions. It processes user queries, applies NLP techniques to understand intent, and generates responses. Continuous testing, feedback collection, and model updates are pivotal for maintaining and improving the chatbot's performance over time.