Create a chatbot in Python

**Phase 4: Development Part 2**

**Introduction:**

Building a chatbot and integrating it into a web app using Flask is a common and practical application of chatbot technology. Flask is a lightweight Python web framework that's well-suited for creating web applications, and you can easily integrate a chatbot into a Flask-based web app.

**1. Install Required Libraries:**

Make sure you have Flask and any other necessary libraries installed. You may want to use a chatbot framework or library, like ChatterBot or Rasa, to simplify chatbot development. Install them using pip:

bash

pip install Flask

pip install chatterbot

**2. Create a Flask Web App:**

Create a Flask web app by creating a Python file, e.g., `app.py`. Here's a basic example:

python

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

**3. Create HTML Template:**

Create an HTML template for your chat interface. You can use this template to collect user input and display chatbot responses. Save this template as `index.html` in a `templates` directory in your project folder.

html

<!DOCTYPE html>

<html>

<head>

<title>Chatbot Example</title>

</head>

<body>

<h1>Chatbot Example</h1>

<div id="chatbox">

<div id="chatlog">

<!-- Chat messages will appear here -->

</div>

<input type="text" id="user\_input" placeholder="Type your message...">

<button id="send">Send</button>

</div>

</body>

</html>

**4. Implement Chatbot Logic:**

Implement your chatbot logic in Python. You can use ChatterBot, Rasa, or any other chatbot framework of your choice. Define a function in your Flask app that handles the chatbot interaction, taking user input and returning bot responses.

**5. Handle User Input:**

Add JavaScript to your HTML template to handle user input and display chatbot responses. You can use AJAX or WebSocket to communicate with your Flask app. Here's a simple example using jQuery:

html

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script>

$(document).ready(function () {

$("#send").click(function () {

var user\_input = $("#user\_input").val();

$("#chatlog").append("<p>User: " + user\_input + "</p>");

$("#user\_input").val("");

$.ajax({

type: "POST",

url: "/get\_response",

data: JSON.stringify({ user\_input: user\_input }),

contentType: "application/json; charset=utf-8",

dataType: "json",

success: function (data) {

$("#chatlog").append("<p>Bot: " + data.bot\_response + "</p>");

}

});

});

});

</script>

**6. Create a Flask Route for Chatbot Interaction:**

In your Flask app, create a route to handle chatbot interactions. This route should take user input, process it, and return the chatbot's response.

python

from flask import request, jsonify

@app.route('/get\_response', methods=['POST'])

def get\_response():

user\_input = request.json['user\_input']

# Process user\_input and get the chatbot's response

bot\_response = get\_chatbot\_response(user\_input)

return jsonify({'bot\_response': bot\_response})

**7. Run the Flask App:**

Run your Flask app using the command `python app.py`. You can access your chatbot web app at `http://localhost:5000` in your web browser.

**8. Test and Refine:**

Test your chatbot and make any necessary refinements to improve its functionality and user experience.

These are the basic steps to integrate a chatbot into a Flask web app. Depending on your chatbot's complexity, you may need to add features like natural language processing, user authentication, and more to create a fully functional and secure chat application.

**Program:**

import pickle

from flask import Flask, request, jsonify

import pandas as pd

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from spellchecker import SpellChecker # You may need to install this library

app = Flask(\_\_name)

# Load your dialog dataset

dataset = pd.read\_csv('dialog.csv')

# Preprocess the dataset

tokenizer = Tokenizer()

tokenizer.fit\_on\_texts(dataset['a'])

total\_words = len(tokenizer.word\_index) + 1

# Tokenize and pad the sequences

input\_sequences = []

for line in dataset['a']:

token\_list = tokenizer.texts\_to\_sequences([line])[0]

for i in range(1, len(token\_list)):

n\_gram\_sequence = token\_list[:i + 1]

input\_sequences.append(n\_gram\_sequence)

max\_sequence\_length = max([len(x) for x in input\_sequences])

input\_sequences = pad\_sequences(input\_sequences, maxlen=max\_sequence\_length, padding='pre')

# Separate input and target sequences

X = input\_sequences[:, :-1]

y = input\_sequences[:, -1]

# Create and compile the model

model = Sequential()

model.add(Embedding(total\_words, 100, input\_length=max\_sequence\_length - 1))

model.add(LSTM(150))

model.add(Dense(total\_words, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam')

# Train the model with a specified number of epochs

model.fit(X, y, epochs=10) # You can specify the number of epochs here

# Save the trained model to a pkl file

with open('model.pkl', 'wb') as model\_file:

pickle.dump(model, model\_file)

# Initialize a spell checker

spell = SpellChecker()

@app.route('/chatbot', methods=['POST'])

def chatbot():

user\_input = request.json['user\_input'] # Assuming you're receiving JSON input

# Load the trained model from the pkl file

with open('model.pkl', 'rb') as model\_file:

model = pickle.load(model\_file)

# Preprocess user input

user\_input = user\_input.lower() # Convert to lowercase

user\_input = spell.correction(user\_input) # Correct spelling

input\_sequence = tokenizer.texts\_to\_sequences([user\_input])[0]

input\_sequence = pad\_sequences([input\_sequence], maxlen=max\_sequence\_length - 1, padding='pre')

# Generate a response using the trained model

response\_sequence = []

for \_ in range(max\_sequence\_length - 1):

predicted\_word\_index = model.predict\_classes(input\_sequence, verbose=0)

predicted\_word = ""

for word, index in tokenizer.word\_index.items():

if index == predicted\_word\_index:

predicted\_word = word

break

input\_sequence = pad\_sequences([input\_sequence.tolist() + [predicted\_word\_index]], maxlen=max\_sequence\_length - 1, padding='pre')

response\_sequence.append(predicted\_word)

response = ' '.join(response\_sequence)

return jsonify({'response': response})

if \_name\_ == '\_main\_':

app.run(debug=True)

**Project By:**

**NAME:** M. Deepak

**DEPT:** CSE III YEAR

**COLLEGE:** AKT MEMORIAL COLLEGE OF ENGINEERING

**GROUP:** AI GROUP-5

**REG NO:** 420121104017