**CHATBOT USING PYTHON**

**Phase 4 - Project Development Part 2**

The code provided below is an extension of the previous code shared in Phase 3. It implements a chatbot using Flask, a Python web framework, to create a simple web-based chat interface where users can interact with the chatbot.

**Overview:**

1. **Imports:** The code begins by importing the necessary libraries, including NLTK for natural language processing, Keras for loading the pre-trained model, JSON for reading the intents data, and Flask for creating a web application.

2. **Load Pre-trained Model and Data:**

- The pre-trained model is loaded from 'model.h5', which was trained in the previous code.

- The intents data is loaded from 'data.json', which contains information about the intent categories and their associated responses.

3. **Text Preprocessing:**

- Functions for text preprocessing are defined, including `clean\_up\_sentence` and `bow`, to tokenize and create a bag of words from user input.

4. **Intent Prediction:**

- The `predict\_class` function uses the pre-trained model to predict the intent of a user's input based on the bag of words representation.

5. **Response Generation:**

- The `getResponse` function selects a random response from the intents data based on the predicted intent.

6. **Chatbot Interaction:**

- The `chatbot\_response` function combines the previous functions to generate a response to a user's message.

7. **Flask Web Application:**

- The code defines a Flask web application with two routes:

- The root route ("/") returns an HTML template for the chat interface.

- The "/get" route handles the chat interactions. It takes the user's message as a query parameter and returns the chatbot's response.

8. **Running the Application:**

- The `if \_name\_ == "\_main\_":` block ensures that the Flask app is only started if the script is executed directly (not when it's imported as a module).

- The app is run, and users can access the chatbot interface through a web browser.

--------------------------------------------------------------------------------------------------------------------------------------

**CODE:**

import nltk

nltk.download('popular')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

from keras.models import load\_model

model = load\_model('model.h5')

import json

import random

intents = json.loads(open('data.json').read())

words = pickle.load(open('texts.pkl','rb'))

classes = pickle.load(open('labels.pkl','rb'))

def clean\_up\_sentence(sentence):

# tokenize the pattern - split words into array

sentence\_words = nltk.word\_tokenize(sentence)

# stem each word - create short form for word

sentence\_words = [lemmatizer.lemmatize(word.lower()) for word in sentence\_words]

return sentence\_words

# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence

def bow(sentence, words, show\_details=True):

# tokenize the pattern

sentence\_words = clean\_up\_sentence(sentence)

# bag of words - matrix of N words, vocabulary matrix

bag = [0]\*len(words)

for s in sentence\_words:

for i,w in enumerate(words):

if w == s:

# assign 1 if current word is in the vocabulary position

bag[i] = 1

if show\_details:

print ("found in bag: %s" % w)

return(np.array(bag))

def predict\_class(sentence, model):

# filter out predictions below a threshold

p = bow(sentence, words,show\_details=False)

res = model.predict(np.array([p]))[0]

ERROR\_THRESHOLD = 0.25

results = [[i,r] for i,r in enumerate(res) if r>ERROR\_THRESHOLD]

# sort by strength of probability

results.sort(key=lambda x: x[1], reverse=True)

return\_list = []

for r in results:

return\_list.append({"intent": classes[r[0]], "probability": str(r[1])})

return return\_list

def getResponse(ints, intents\_json):

tag = ints[0]['intent']

list\_of\_intents = intents\_json['intents']

for i in list\_of\_intents:

if(i['tag']== tag):

result = random.choice(i['responses'])

break

return result

def chatbot\_response(msg):

ints = predict\_class(msg, model)

res = getResponse(ints, intents)

return res

from flask import Flask, render\_template, request

app = Flask(\_name\_)

app.static\_folder = 'static'

@app.route("/")

def home():

return render\_template("index.html")

@app.route("/get")

def get\_bot\_response():

userText = request.args.get('msg')

return chatbot\_response(userText)

if \_name\_ == "\_main\_":

app.run()

--------------------------------------------------------------------------------------------------------------------------------------

The chatbot responds to user messages with responses based on the intents defined in 'data.json' and the pre-trained model. Users can interact with the chatbot by entering messages in the web interface.

**TEAM MEMBERS:**

Reshmi M J

Suvetha R

Pavithra A

Mohammed Nawfal M S

Nidhish babu T L S