**CHATBOT USING PYTHON**

**Phase 5-Project Documentation**

**Problem statement**

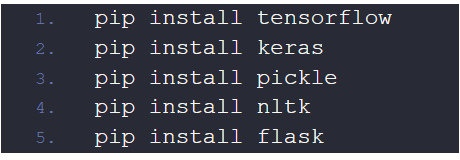
**Creating a chatbot in Python for customer service is a valuable endeavor to enhance user support, engagement, and satisfaction. A customer service chatbot can provide efficient, 24/7 assistance to users, answer common queries, and free up human agents for more complex tasks .In today's digital age, businesses are increasingly turning to chatbots to improve their customer service offerings. These chatbots, powered by Python and advanced Natural Language Processing (NLP) techniques, are designed to understand and respond to user queries in a conversational and helpful manner.**

**Design Thinking process**

1. **Functionality:** Define the scope of the chatbot’s abilities, including answering common questions, providing guidance, and directing users to appropriate resources.
2. **User Interface:** Determine where the chatbot will be integrated(website, app) and design a user-friendly interface for interactions.
3. **Natural Language Processing (NLP):** Implement NLP techniques to understand and process user input in a controversial manner.
4. **Responses:** Plan responses that the chatbot will offer, such as accurate answers, suggestions, and assistance.
5. **Integration:** Decide how the chatbot will be integrated with the website or app.
6. **Testing and Improvement:** Continuously test and refine the chatbot’s performance based on user interactions.

**Prerequisites**

The project requires you to have good knowledge of Python, Keras, and Natural language processing (NLTK). Along with them, we will use some helping modules which you can download using the python-pip command.



Now we are going to build the chatbot using Flask framework but first, let us see the file structure and the type of files we will be creating:

* **data.json –** The data file which has predefined patterns and responses.
* **trainning.py –** In this Python file, we wrote a script to build the model and train our chatbot.
* **Texts.pkl –** This is a pickle file in which we store the words Python object using Nltk that contains a list of our vocabulary.
* **Labels.pkl –** The classes pickle file contains the list of categories(Labels).
* model.h5 – This is the trained model that contains information about the model and has weights of the neurons.
* app.py – This is the flask Python script in which we implemented web-based GUI for our chatbot. Users can easily interact with the bot.

**Dataset used**

The data file is in JSON format so we used the json package to parse the JSON file into Python. This is how our data.json file looks like.

**{**

**"intents":[**

**{**

**"tag": "greeting",**

**"patterns": [**

**"Hi",**

**"How are you?",**

**"Is anyone there?",**

**"Hello",**

**"Good day",**

**"What's up",**

**"how are ya",**

**"heyy",**

**"whatsup",**

**"??? ??? ??"**

**],**

**"responses": [**

**"Hello!",**

**"Good to see you again!",**

**"Hi there, how can I help?"**

**],**

**"context\_set": ""**

**},**

**"What's up",**

**"how are ya",**

**"heyy",**

**"whatsup",**

**"??? ??? ??"**

**],**

**"responses": [**

**"Hello!",**

**"Good to see you again!",**

**"Hi there, how can I help?"**

**],**

**"context\_set": ""**

**},**

**Data Preprocessing**

When working with text data, we need to perform various preprocessing on the data before design an ANN model. Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

Here we iterate through the patterns and tokenize the sentence using nltk.word\_tokenize() function and append each word in the words list. We also create a list of classes for our tags.

Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

**Creating Training Data**

Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn’t understand text so we will convert text into numbers.

**Machine learning algorithm**

The chatbot will be trained on the dataset which contains categories (intents), pattern and responses. We use a special artificial neural network (ANN) to classify which category the user’s message belongs to and then we will give a random response from the list of responses. Stochastic gradient descent with Nesterov accelerated gradient is used train the model

**Model Training**

We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as ‘model.h5’.

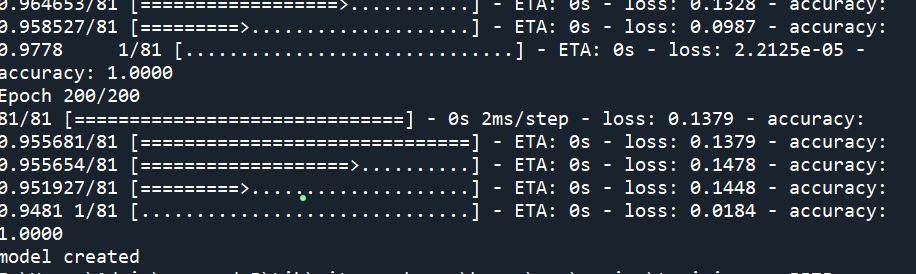
**Flask based GUI**

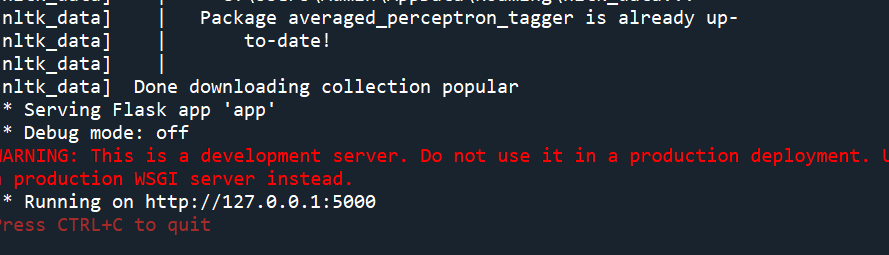
We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve a random response from the list of responses.

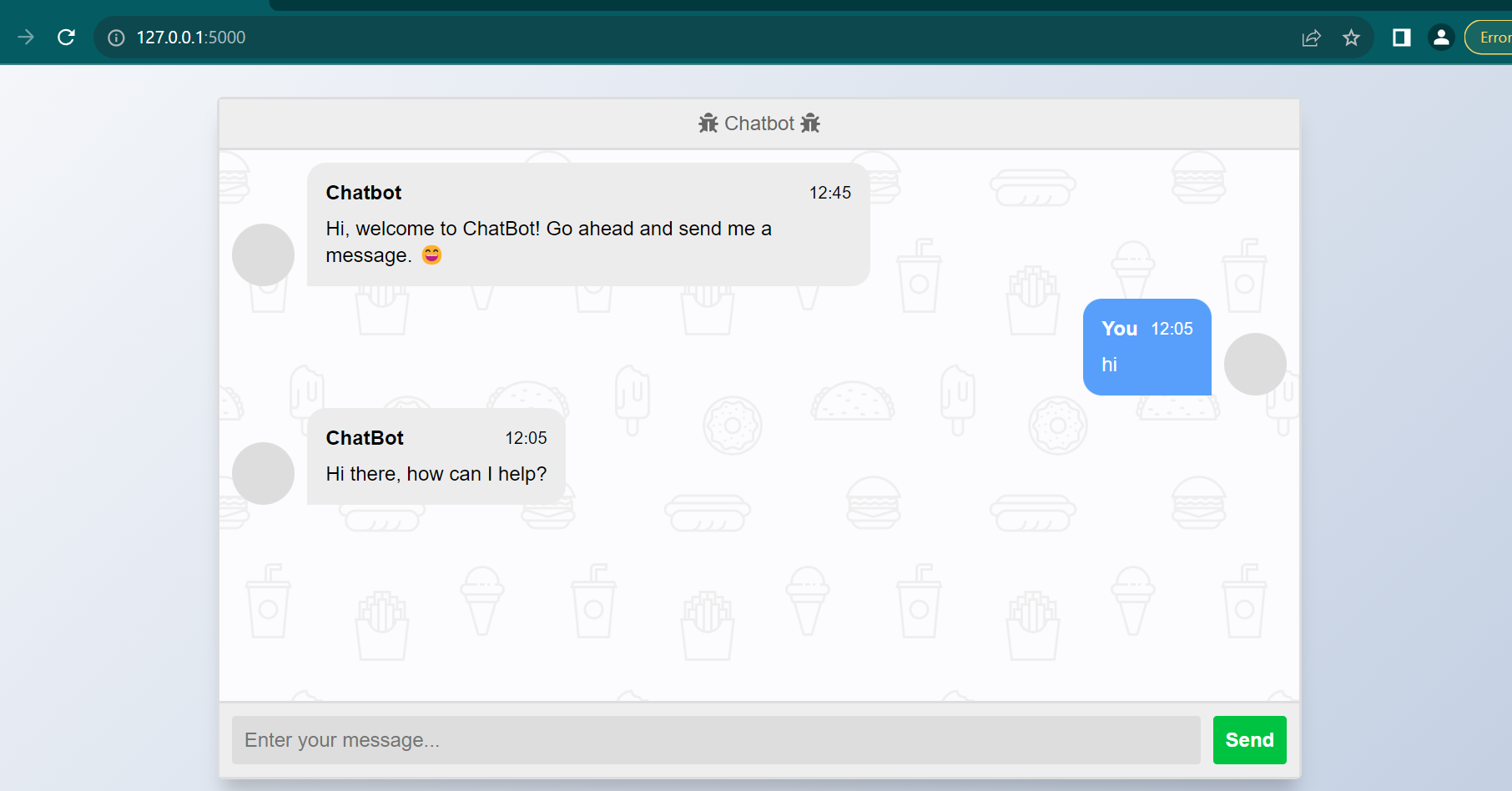
Again we import the necessary packages and load the ‘texts.pkl’ and ‘labels.pkl’ pickle files which we have created when we trained our model:

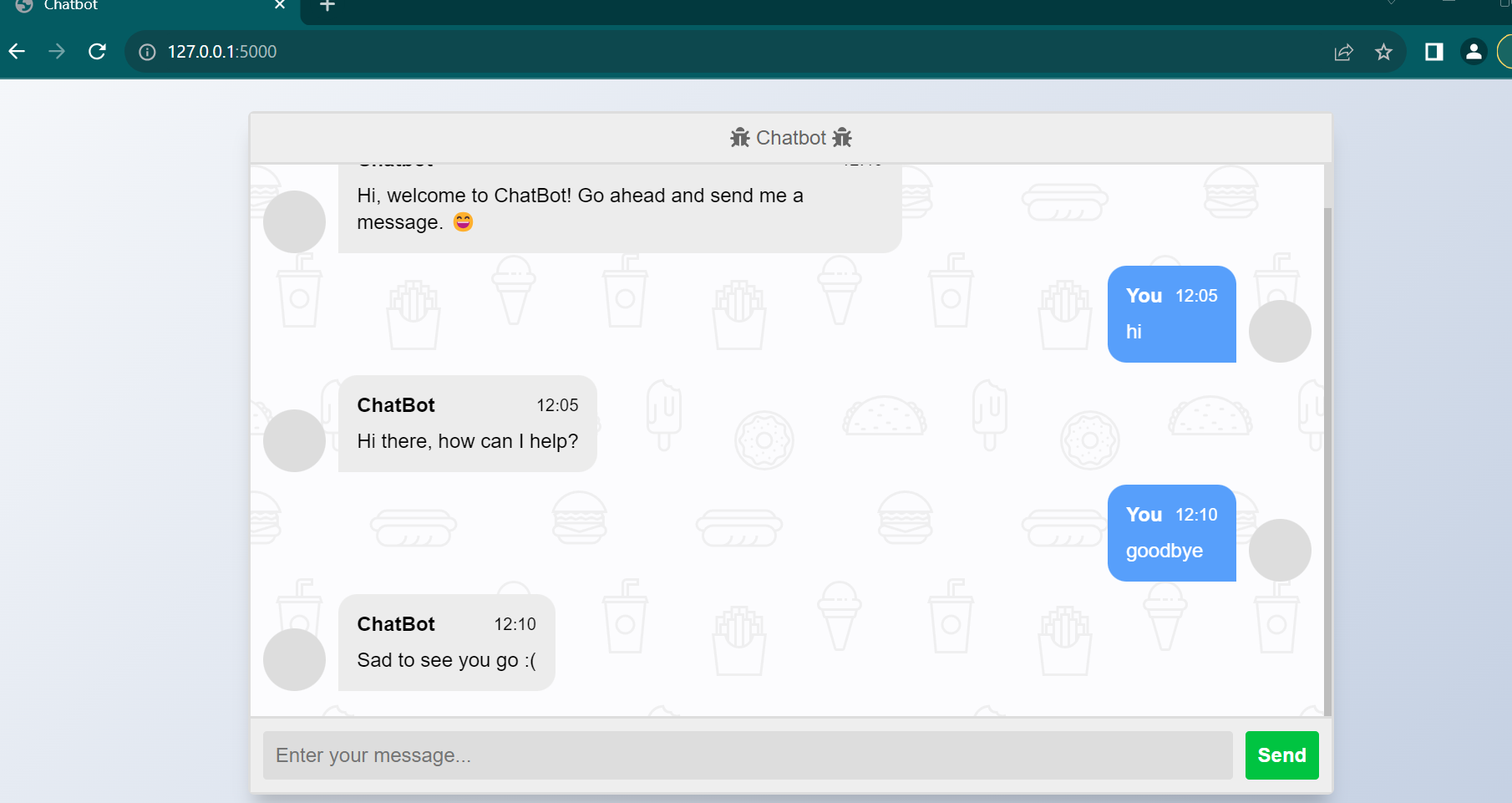
To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class. After predicting the class, we will get a random response from the list of intents.

**Output**

****

****

****

****

**Improving the Chatbot**

There are many ways to improve a chatbot, and I’ll share some ideas below:

1. Use a more advanced language model: One way to improve the chatbot is to use a more advanced language model, such as GPT-3 or BERT, which have been pre-trained on massive amounts of text data and can generate human-like responses. You can use a pre-trained model and fine-tune it on your specific chatbot task.
2. Add more training data: Another way to improve the chatbot is to use more training data, ideally with a wide range of conversational topics and styles. You can scrape data from social media or forums, or use existing chatbot datasets such as Cornell Movie Dialogs Corpus or Persona-Chat.
3. Use a more complex model architecture: You can also improve the chatbot’s performance by using a more complex model architecture, such as a transformer-based model, which can capture longer-term dependencies in the input sequence.
4. Incorporate user feedback: You can incorporate user feedback into the chatbot to improve its responses over time. For example, you can ask users to rate the quality of the chatbot’s responses or suggest alternative responses, and use this feedback to retrain the model.
5. Add multi-turn conversation capability: The current chatbot can only handle one question-answer exchange at a time. You can improve the chatbot by adding multi-turn conversation capability, allowing the chatbot to remember previous conversation context and generate more meaningful responses.
6. Implement personality and emotional intelligence: You can also make the chatbot more engaging and human-like by implementing personality and emotional intelligence. For example, you can give the chatbot a specific personality trait, such as being funny or sarcastic, or use sentiment analysis to detect and respond to the user’s emotional state.

**Conclusion**

In this project, we have built a simple chatbot using Python and TensorFlow. We started by gathering and preprocessing data, then we built a neural network model using the Keras Sequential API. We then created a simple command-line interface for the chatbot and tested it with some example conversations.

**TEAM MEMBERS:**

Nidhish babu T L S

Mohammed Nawfal M S

Reshmi M J

Suvetha R

Pavithra A