Project Report

Chatbot Using Deep Learning

1. Introduction

Overview

This project focuses on the development of a simple chatbot that can answer predefined questions using deep learning and a rule-based logic approach. The chatbot is implemented in Python using Gradio for the user interface and TensorFlow for the deep learning model. The questions and responses are predefined and stored in a JSON file.

Objective

The primary objective of this project is to build a chatbot capable of interacting with users by providing accurate responses to predefined questions. This project aims to demonstrate the principles of deep learning and natural language processing (NLP) in creating a functional chatbot.

Scope

The scope of this project includes:

- Loading and parsing a JSON file containing predefined questions and responses.
- Developing a deep learning model to predict appropriate responses based on user input.
- Creating a graphical user interface (GUI) using Gradio to facilitate user interaction with the chatbot.
- Ensuring the chatbot can handle a variety of user inputs within the predefined scope.

Importance

This project is significant as it provides a foundational understanding of how chatbots work and can be implemented. It demonstrates the use of deep learning and NLP techniques in building interactive applications and highlights the importance of predefined rule-based systems in scenarios where user interactions are predictable and limited.

2. Algorithm

Description

The algorithm for the rule-based chatbot involves several steps, including loading the dataset, preprocessing the data, building the model, and creating the interface for user interaction.

Steps

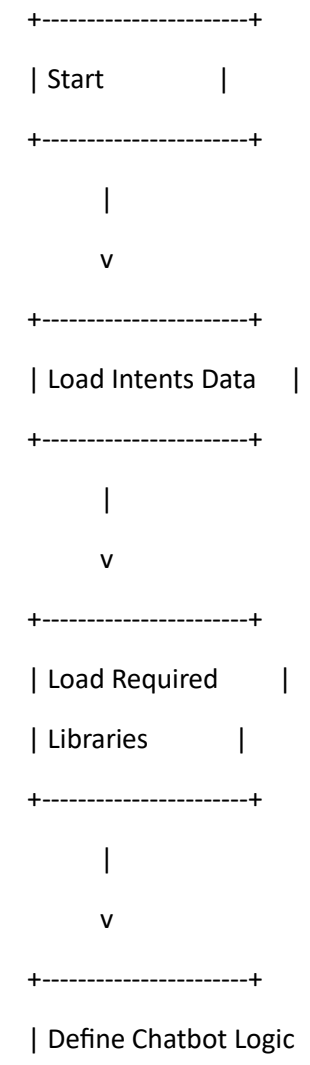
1. **Define Intents and Responses:**
    - Load predefined questions and responses from a JSON file.
2. **Load Required Libraries:**
    - Import necessary libraries including json, numpy, keras, sklearn, random, pickle, and gradio.

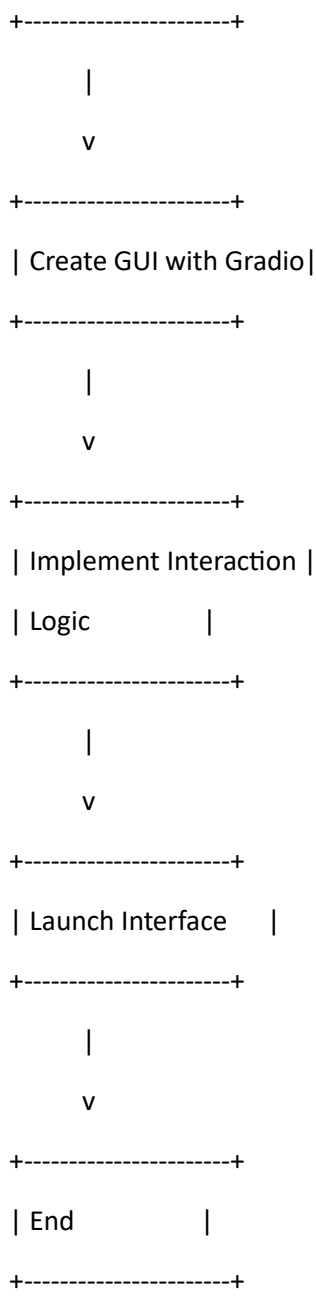
3. **Load Intents Data:**
    - Load the JSON file containing the intents into a data structure.
4. **Define the Chatbot Logic:**
    - Use deep learning to predict the response based on user input.
5. **Create the GUI:**
    - Use Gradio to create a web-based interface for the chatbot.
6. **Implement Interaction Logic:**
    - Map user input to predefined responses based on the loaded intents.
7. **Launch the Interface:**
    - Launch the Gradio app to allow user interaction with the chatbot.

Flowchart

plaintext

Copy code





Examples

- User Input: "Hello"
- Predicted Response: "Hello! How can I help you?"

3. Source Code

Python Code

python

Copy code

```
import json
import numpy as np
```

```
from tensorflow import keras
from sklearn.preprocessing import LabelEncoder
import random
import pickle
import gradio as gr

# Load the intents file
with open("intents.json") as file:
    data = json.load(file)

# Load the trained model
model = keras.models.load_model('chat_model')

# Load the tokenizer object
with open('tokenizer.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)

# Load the label encoder object
with open('label_encoder.pickle', 'rb') as enc:
    lbl_encoder = pickle.load(enc)

# Parameters
max_len = 20

def predict_response(user_input, history):
    result =
model.predict(keras.preprocessing.sequence.pad_sequences(tokenizer.texts_to_sequences([user_in
put]),
                                          truncating='post', maxlen=max_len))
    tag = lbl_encoder.inverse_transform([np.argmax(result)])

    # Prepare the response
```

```
    bot_response = "I don't understand."
    for i in data['intents']:
        if i['tag'] == tag:
            bot_response = random.choice(i['responses'])

    # Append the new chat messages to history
    history.append((f"You: {user_input}", f"ChatBot: {bot_response}"))
    return history

# Create Gradio interface
with gr.Blocks() as demo:
    # Add custom styling with Markdown
    gr.Markdown("""
    <style>
    .container {
        background-color: #f5f5f5; /* Light gray background color */
        padding: 10px; /* Smaller padding */
        border-radius: 8px; /* Slightly smaller border radius */
        max-width: 400px; /* Narrower width of the container */
        margin: 0 auto; /* Center horizontally */
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        min-height: 100vh; /* Center vertically */
        box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1); /* Optional: Add shadow */
    }
    .input-box {
        width: 100%;
        margin: 5px 0; /* Smaller margin */
    }
```

```
    .send-button {
        width: 100%;
        margin: 5px 0; /* Smaller margin */
        display: block;
    }
    </style>
    <div class="container">
        <h2 style="color: #333; font-size: 20px;">Chat with the bot</h2>
    </div>
    """)

    # Chat history display
    chat_history = gr.Chatbot()

    # Input field for user message
    user_input = gr.Textbox(placeholder="Type your message here...", label="Your Message",
elem_id="input-box")

    # Send button
    send_button = gr.Button("Send", elem_id="send-button")

    # Define the callback function
    def respond(message, history):
        return predict_response(message, history)

    # Set up interaction
    send_button.click(respond, inputs=[user_input, chat_history], outputs=chat_history)

# Launch the Gradio app
demo.launch(share=True)
```

Libraries Used

- json
- numpy
- tensorflow.keras
- sklearn
- random
- pickle
- gradio

Code Explanation

The code consists of several main parts:

1. **Loading Data and Model:**
    - The intents are loaded from a JSON file.
    - The trained deep learning model, tokenizer, and label encoder are loaded from disk.
2. **Defining the Chatbot Logic:**
    - The predict_response function takes user input and predicts the appropriate response using the trained model.
3. **Creating the GUI:**
    - A Gradio interface is created with custom styling to provide a user-friendly chat environment.
4. **Interaction Logic:**
    - The respond function processes user input, updates chat history, and displays the chatbot's response.
5. **Launching the Interface:**
    - The Gradio app is launched to enable user interaction.

Examples

- **User Input:** "Hi"
- **Chatbot Response:** "Hello! How can I help you?"

4. Documentation and PPT

Project Report

- **Introduction:** Overview and objective of the project.
- **Methodology:** Detailed explanation of the algorithm, dataset, and implementation.
- **Results:** Description of the chatbot's performance and examples of interactions.
- **Conclusion:** Summary of findings and future work.

Presentation Slides

1. **Title Slide:** Project title and team members.
2. **Introduction:** Brief overview and objective.
3. **Methodology:** Steps and flowchart of the algorithm.
4. **Implementation:** Key parts of the code and GUI screenshots.
5. **Results:** Examples of chatbot interactions.
6. **Conclusion:** Summary and future work.

References

- Gradio Documentation
- TensorFlow Documentation
- Keras Documentation

Appendices

- **Appendix A:** Full source code.
- **Appendix B:** Detailed explanation of the JSON intents file.
- **Appendix C:** Installation and setup instructions.