Chatbot Project Documentation

1. Project Overview

This chatbot project is built using Python and Flask for the backend, along with a machine learning model trained on a dataset of questions and answers. The frontend is implemented using HTML, CSS, and JavaScript to create an interactive chat interface.

Features:

- **Natural Language Processing (NLP):** Uses SpaCy for text preprocessing.
- **Machine Learning Model:** Trained with Support Vector Classification (SVC) using TF-IDF vectorization.
- **REST API:** Built with Flask to handle chatbot interactions.
- **User-Friendly Interface:** Simple and modern UI for chat interactions.

2. System Requirements

Prerequisites:

- Python 3.x
- Flask
- Scikit-learn
- Pandas
- SpaCy
- Pickle
- A browser (for frontend testing)

3. Installation & Setup

Step 1: Install Required Software

Install Python:

Download and install Python 3.9+ from python.org.
Verify installation:

```
python –version
```

Install an IDE

Recommended: Install PyCharm or VS Code for Python development.

Install Dependencies

Open a terminal and run:

```
pip install flask pandas scikit-learn spacy gunicorn
```

Download the spaCy NLP model:

```
python -m spacy download en_core_web_sm
```

Step 2: Prepare Training Data

Create a CSV file (training_data.csv) in your project folder.

Add chatbot training data (question-answer pairs):

```
question,answer
Hello,Hi there! How can I help you?
How are you?,I'm doing great! How about you?
What is AI?,Artificial Intelligence is the simulation of human intelligence in machines.
Goodbye,See you later! Take care.
```

Step 3: Train the Chatbot Model

Create train_chatbot.py

```
import pandas as pd
import spacy
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
# Load dataset
df = pd.read_csv("training_data.csv")
# Load NLP model
nlp = spacy.load("en_core_web_sm")
# Preprocess text
def preprocess(text):
    doc = nlp(text.lower())
    return " ".join([token.lemma_ for token in doc if not token.is_punct])
df["processed_question"] = df["question"].apply(preprocess)
# Convert text to TF-IDF features
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df["processed_question"])
y = df["answer"]
# Train model
model = SVC(kernel="linear")
model.fit(X, y)
# Save model
pickle.dump(model, open("chatbot_model.pkl", "wb"))
pickle.dump(vectorizer, open("vectorizer.pkl", "wb"))
print(" Model training complete!")
```

📌 **Run the training script:**

```
python train_chatbot.py
```

This generates:

chatbot_model.pkl (trained model)
vectorizer.pkl (TF-IDF processor)

Step 4: Create the Chatbot API

Create chatbot_api.py

```
import pickle
import spacy
from flask import Flask, request, jsonify
# Load trained model & vectorizer
model = pickle.load(open("chatbot_model.pkl", "rb"))
vectorizer = pickle.load(open("vectorizer.pkl", "rb"))
nlp = spacy.load("en_core_web_sm")
# Flask API
app = Flask(__name__)
@app.route("/chat", methods=["POST"])
def chat():
    data = request.json
    user_message = data.get("message", "")
    # Preprocess input
    doc = nlp(user_message.lower())
    processed_input = " ".join([token.lemma_ for token in doc if not token.is_punct])
    # Convert input to TF-IDF features
    input_vector = vectorizer.transform([processed_input])
    # Get chatbot response
    response = model.predict(input_vector)[0]
        return jsonify({"response": response})
if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

📌 **Run the API server:**

```
python chatbot_api.py
```

The chatbot API is now running at http://localhost:5000/chat.

Step 5: Create a Simple Frontend

Create index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Chatbot</title>
</head>
<body>
    <h2>Chatbot</h2>
    <input type="text" id="userInput" placeholder="Type your message...">
    <button onclick="sendMessage()">Send</button>
    <p id="botResponse"></p>

    <script>
        function sendMessage() {
            let userMessage = document.getElementById("userInput").value;

            fetch("http://localhost:5000/chat", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                body: JSON.stringify({ message: userMessage })
            })
            .then(response => response.json())
            .then(data => {
                document.getElementById("botResponse").innerText = "Bot: " + data.response;
            });
        }
    </script>
</body>
</html>
```

📌 Open index.html in a browser and chat with the bot.

4. Training the Model

To train the chatbot, run the following command:

```
python train_chatbot.py
```

This script will:

- Load and preprocess the training dataset (training_data.csv).
- Train a **Support Vector Classification model**.
- Save the trained model (chatbot_model.pkl) and vectorizer (vectorizer.pkl).

5. Running the Chatbot API

Start the Flask server:

```
python chatbot_api.py
```

This will host the chatbot API on http://127.0.0.1:5000/.

6. API Endpoints

POST /chat

Request:

```
{
  "message": "Hello, chatbot!"
}
```

Response:

```
{
  "response": "Hi there! How can I help you?"
}
```

7. Frontend Integration

The frontend (index.html) allows users to chat with the bot via a simple UI.

Steps to Run:

1. Open index.html in a browser.
2. Type a message and send it.
3. The chatbot API will respond accordingly.

8. Troubleshooting & FAQs

Q: I get an error when Installing the library.

- Error occurs when I am importing the flask pandas scikit-learn spacy gunicorn.

Q: I get an error when running train_chatbot.py.

- Ensure training_data.csv exists and is correctly formatted.
- Install missing dependencies using `pip install -r requirements.txt`.

Q: My chatbot is not responding.

- Check if the Flask server is running.
- Open browser console (F12) and look for any errors.

9. Summary

- Step 1: Install Python, IDE, dependencies.
- Step 2: Prepare CSV training data.
- Step 3: Train the NLP model.
- Step 4: Create Flask API.
- Step 5: Develop a simple frontend.