CHATBOT USING PYTHON

PHASE 5 – PROJECT DOCUMENTATION

PROBLEM STATEMENT

In a world of increasing automation and AI, there's a growing need for natural language interfaces to interact with machines. Our project aims to address this by developing an AI chatbot capable of recognizing user intent and generating contextually relevant responses. The problem statement revolves around creating a conversational AI system that can effectively understand and respond to user queries and commands.

DESIGN THINKING PROCESS

Phase 1: Project Planning and Research

- **Objectives**: Our objectives were to design a user-friendly chatbot interface that can assist users by understanding their intent and providing meaningful responses.
- **Target Audience**: We aimed to create a chatbot that could be utilized by individuals across various domains, such as customer support, education, and general information retrieval.
- **Research**: We conducted extensive research into Natural Language Processing (NLP), chatbot development, and the use of machine learning for intent recognition. We gathered resources, tutorials, and libraries relevant to our project.

Phase 2: Data Collection and Preparation

- **Dataset**: We utilized a dataset consisting of user questions and corresponding answers. The dataset was a collection of dialogues covering a wide range of topics.
- **Data Acquisition Challenges**: The dataset had formatting issues and required extensive preprocessing, including removing unnecessary symbols and rearranging the data into a structured format.

Phase 3: Model Design and Development

- **Machine Learning Algorithm**: We opted for a Multinomial Naive Bayes classifier due to its effectiveness in text classification tasks.
- **Model Training**: Our model was trained using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization to convert text data into numerical form.
- **Development**: We built a Flask-based API to enable interaction with the chatbot.

Phase 4: Development Part

- **User Interface**: We designed a simple user interface for the chatbot using HTML and CSS. The interface featured a user input box, a send button, and a chat log display.
- **Flask Application**: We developed a Flask-based web application to host the chatbot and facilitate user interactions.

Phase 5: Project Documentation & Submission

- We understood the importance of proper documentation and structured submission to make our project accessible to others.

DATASET DESCRIPTION

Our dataset is a compilation of dialogue pairs, each consisting of a user's question and the corresponding response. The data is tab-separated and comes from various sources, covering a wide array of topics.

DATA PREPROCESSING

- Data was loaded into a Pandas DataFrame.
- Columns were renamed for clarity.
- We removed special characters and symbols to ensure data quality.

MODEL TRAINING

- We employed a Multinomial Naive Bayes classifier to classify user intent.
- We utilized the TF-IDF vectorization technique to transform text data into numerical features.
- The model was trained on the prepared dataset.

EVALUATION METRICS

- To assess the model's performance, we used accuracy as the primary metric to measure the proportion of correctly classified intents.
- We generated classification reports to provide more insights into the model's performance.

USER INTERFACE

We developed a user-friendly interface for the chatbot using HTML and CSS. The interface includes:

- An input box for users to type messages.
- A send button to submit user queries.
- A chat log display to view the conversation.

The Flask-based web application hosts the chatbot and connects the interface to the AI model.

INSTRUCTIONS FOR RUNNING THE CODE

To run the project and interact with the chatbot, follow these steps:

1. **Install Dependencies**: Ensure you have Python, Flask, Transformers, Pandas, scikit-learn, and other required libraries installed. If not, use **pip install** to install them.
2. **Set Up OpenAI GPT-3 API Key**: Set up your OpenAI GPT-3 API key:
    a. If you don't have one, sign up for the OpenAI API and obtain an API key.
    b. Save your API key in a file named **config.py** as **API_KEY = 'your_api_key'**.
3. **Execute the Code**: Run 'main.py' using Python. This starts the Flask server hosting the chatbot.
4. **Access the Chatbot**: Open a web browser and navigate to the provided URL, usually 'http://127.0.0.1:5000'. You will be greeted by the chatbot interface.

5. **Interact with the Chatbot**: Type your message in the input box and click the send button. The chatbot will recognize your intent and provide responses.

FILES IN THE PROJECT

In this section, we will provide explanations for the various files that constitute the project. These files are crucial to the functioning of the chatbot system and include both code files and supporting assets.

1. **main.py**
   - **Description**: This is the main Python script that initializes and runs the chatbot system. It includes code for setting up the Flask web application, handling user messages, and generating chatbot responses.
   - **Purpose**: **main.py** is the entry point of the chatbot system, integrating the chatbot model, intent recognition, and the web interface.

2. **config.py**
   - **Description:** This Python file contains a single line of code for storing the OpenAI GPT-3 API key. It is essential for secure access to the OpenAI API.
   - **Purpose:** config.py enhances security by storing the API key outside of the main script (main.py), making it easier for users to use their own API keys without exposing sensitive information.
   - **Usage Instructions:** When you access this project, you should open config.py and replace 'YOUR_API_KEY' with your own valid OpenAI GPT-3 API key. This step is essential for proper functioning and API access.

3. **dialogs.txt**
   - **Description**: This is the dataset file in tab-separated values (TSV) format that contains a collection of questions and corresponding answers used for training the intent recognition model.
   - **Purpose**: **dialogs.txt** serves as the training data for the machine learning model that recognizes user intents.

4. **script.js**
   - **Description**: This JavaScript file is responsible for handling user interactions on the web interface. It captures user input, sends it to the chatbot server, and displays responses.
   - **Purpose**: **script.js** provides interactivity to the web interface, enabling users to communicate with the chatbot.

5. **styles.css**
   - **Description**: This Cascading Style Sheets (CSS) file contains styles and formatting rules for the HTML elements on the web interface, ensuring a visually appealing user experience.
   - **Purpose**: **styles.css** enhances the presentation and user interface design.

6. **index.html**

- **Description**: This HTML file defines the structure and layout of the web interface. It includes input fields, chat log display, and a button for sending messages.
- **Purpose**: The HTML file creates the user interface where users interact with the chatbot.

SUBMISSION

For submission, we have compiled all the code files and project assets, including data preprocessing, model training, and evaluation steps.

- We provided a well-structured README file that explains how to run the code and lists all necessary dependencies.
- We included the dataset source and provide a brief description of the data used.
- The entire project, including code, documentation, and assets, will be shared on GitHub for others to access, review, and potentially contribute to.

This project is a step towards creating more intelligent and conversational AI systems that can assist users across various domains, from customer support to educational applications. We believe that well-documented and accessible projects like this can inspire further advancements in AI

BY:

SHWETHA V

ROOBINI M

GOVINDALALITHA T M

HARINI PRIYAA S