Text-Text Generator Chatbot ■

Welcome to **Text-Text Generator Chatbot**, an all-in-one application for handling various text-processing tasks like paraphrasing, grammar checking, AI detection, and more! This project combines the capabilities of tools like Grammarly and ChatGPT to streamline text-related tasks.

Demo video

https://github.com/user-attachments/assets/514dac38-7a68-48cc-a826-d0fa895a8553

Project Overview

This Flask-based chatbot application allows users to perform various text-generation and text-manipulation tasks with ease. Simply enter your text, select a task, and let the application do the rest!

Features

- **Paraphrasing**: Reword text to enhance clarity and originality.
- **Grammar Checking**: Detect and correct grammar and spelling errors.
- **AI Detection**: Identify if text is AI-generated or human-written.
- **Plagiarism Checking**: Ensure originality by scanning for copied content.
- **Summarization**: Condense lengthy text into a concise summary.
- **Short Answer Generation**: Generate brief answers for quick information retrieval.
- **Basic Chat**: Interact with a chatbot for general queries.

Application Benefits

- ■ **Save Time** on basic text tasks.
- ■ **Text Generation & Manipulation** with ease.
- ■ **User-Friendly Interface** for quick and intuitive navigation.

How to Access the Application

1. ■ Visit NVIDIA's website.
2. ■ Create an NVIDIA account.
3. ■■ Generate an API Key for any text-to-text language model.
4. ■ Paste the API Key in the "Access Key" page, then enter the chatbot page.
5. ■ Start using the text generation tools!

Installation & Setup

1. Clone the repository: `bash git clone https://github.com/vijaytakbhate2002/Text-Text-Generator.git`
2. Navigate to the project directory: `bash cd Text-Text-Generator`
3. Install the required dependencies: `bash pip install -r requirements.txt`
4. Run the application: `bash flask run`

File Structure

- **app.py**: The main file containing the Flask application logic.
- **templates/**: Contains HTML templates using Jinja for dynamic content rendering.
- **static/**: Contains static assets (CSS, JavaScript, images) for styling and functionality.

Technologies Used

- **Flask**: Backend framework for the application.
- **HTML/CSS**: Frontend layout and styling.
- **JavaScript**: Enhancing interactivity on the frontend.
- **NVIDIA API**: Provides access to text-to-text language models.

Future Plans

- Enhance the UI/UX with more interactive elements.
- Add additional text-processing tools and options.
- Refine performance and error-handling capabilities.

Contributing

Feel free to fork this repository, create a new branch, and submit a pull request with any improvements or fixes!

License