Task 1: AdhamGPT

This project is a lightweight AI chatbot system designed to interact with users and answer questions. It consists of a **Flask backend** that serves the model and a **Streamlit frontend**, which provides a user interface. The chatbot was designed to run efficiently on limited resources without huge hallucinations, and it is tailored for a specific dataset

Its structure:

LLM-CHATBOT/

- ── backend/
    - app.py # Flask API backend
    - churn.csv # Tabular dataset
    - .env # Hugging Face token
    - requirements.txt # Backend dependencies

- ── frontend/
    - app.py #Streamlit UI
    - requirements.txt # Frontend dependencies

- ── venv/ # Python virtual environment

Technologies that I used:

- In backend: Flask, Transformers, dotenv
- In frontend: Streamlit
- Model: https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0
- tokenizer Hugging Face autotokenizer

project functionality summary:

- Accepts user input and queries a dataset-aware LLM.
- Provides Streamlit interface with chat history.
- Displays execution time.
- Uses local model with efficient memory footprint.
- Disables randomness using temperature=0 to reduce hallucinations.

History:

I used 5 models

1- mistralai/Mistral-7B-Instruct via OpenRouter: fast, but produced inaccurate or unrelated answers (hallucinations). Its answers are way better than GP2 and it is newer than GPT2, but I chose GP2 as it is from Hugging face, as it is lighter, and I can run it easily on my laptop

2- meta-llama/Llama-2-7b-chat-hf via hugging face: Too heavy for my system. Tried 4-bit quantization, but it failed due to the unavailability of CPU quantization

3- tiiuae/falcon-rw-1 via Hugging Face: Lightweight, not as fast as GPT2. The response time was more than 1 minute, and gave non-promising responses

4- GPT2: very lightweight, can run easily on the PC, very fast, it is not as accurate as it requires training and tuning. Perfect for CPU-only environments with limited RAM. However, it hallucinates too much as it is too old and needs much training again

5- Tiny llama: is lightweight, perfect for chatbots as it is a decoder-only model designed for these tasks, it is compatible, and gives really promising responses, it has low memory footprint, fast interface, strong response quality with minimal hallucinations

Backend:

1- I imported the libraries Flask, request, jsonify, pandas, time, torch, os, load_dotenv, AutoTokenizer, and AutoModelForCausalLM

2- I loaded the Hugging Face access token securely from the .env file.

3- I loaded our model and tokenizer from Hugging Face and the Health checked endpoint to confirm the backend is working.

4- Fed the model with in-context learning and small hyperparameter tuning making temp = 0 to reduce randomness

5- I extracted the model reply and returned it with the response time. This app is running locally on port 8500 with no reloading

Frontend:

1- I imported streamlit, requests, and made the basic layout setup, backend URL configuration

2- Then, I initialized the session state to store the chat history and display it

3- I handled user input through a form and sent the question to the Flask backend, waited for the model reply, and displayed it with elapsed time.

To run the project

1- First, activate the environment that I installed, in which the libraries I will use are with specific versions to ensure not crash

```
python -m venv venv
```

```
.venv\Scripts\activate
```

2- Then, open the terminal in the backend folder and run:

Note: I added my token in env like:

```
# Add to .env:
```

```
HF_TOKEN=your_huggingface_token
```

3- And install the requirements

```
pip install -r requirements.txt
```

4- Launch the app backend

```
python app.py
```

then we need to open another terminal in the frontend folder and install the requirements and then launch the app

```
pip install -r requirements.txt
```

```
streamlit run app.py
```

some technical decisions

1- Model quantization disabled: due to limited GPU support on windows and local constraints
2- We chose TinyLlama as best performance-to-size ratio
3- We made the Temperature =0 forcing deterministic outputs for stability
4- The token kept in .env keeps credentials secure and manageable
5- Added some in-context learning to help the model (we could not train the model due to lack of resources)

Adham Tarek Ragab Ali
MSc Researcher in Data Science HU| Cairo, Egypt