Project Description: Argument Quality Analysis Application

Overview

In this project, you will develop an application that uses a large language model to evaluate the quality of arguments presented in text. The application will include a web-based user interface that allows users to submit text arguments and receive a quality score. The score will reflect the coherence and persuasiveness of the argument.

The **dataset** you can use for this is the "IBM Debater® - IBM-ArgQ-Rank-30kArgs" that you can find at
https://research.ibm.com/haifa/dept/vst/debating_data.shtml#Argument_Quality

The paper associated with the dataset is here: https://arxiv.org/pdf/1911.11408

Objectives

- Implement a backend using Python and an LLM to assess argument quality.
- Develop a web-based user interface where users can interact with the model.
- Enhance the application with advanced features, including detailed feedback on argument quality and a comparison tool for multiple arguments.

Requirements

Part 1: Basic Argument Quality Assessment

- Goal: Build the core functionality to score the quality of arguments using a pre-trained language model.
- Resources:
    1. Python packages: `transformers`, `torch`, and `flask`.
    2. Pre-trained model from the Hugging Face library that is suitable for analyzing text quality, or "Ollama" library or even just OpenAI apis.
- Develop a function to process text input and return a quality score.
- Test the function with a set of example argumentative texts.

Part 2: Data Management and UI Development

- Goal: Create a simple, functional web interface for users to submit and evaluate arguments.
- Resources
    1. Flask, FastAPI

    2. Various Javascript frameworks, Reflex API for python, etc.
- Build front-end forms for users to submit their arguments.
- Display the argument quality score returned by the model.
- Ensure robust integration of the front-end with the backend model processing.
- Ensure speed and visual aesthetics

Part 3: Advanced Features and Enhancements

- Goal: Extend the application with additional functionalities to enhance user experience and utility.
- Tasks:
    1. Implement a feature to provide detailed feedback on the argument, highlighting strengths and areas for improvement. (using LLMs)
    2. Optionally, create a comparison feature that allows users to evaluate multiple arguments side by side.
    3. Refine the UI to accommodate new features and ensure intuitive navigation.

Deliverables

- A fully functional web application (github link)
- Source code for both the backend and frontend, including any scripts used for model training or data processing.
- Documentation covering setup, usage, and a detailed explanation of how the application works. Include comments in your code to clarify the purpose of each function and section.
- Basically, I should be able to clone the repo and initialize a python virtual environment, pip install your requirements file and run the code.

Evaluation Criteria

You will be evaluated on the following criteria:

- Functionality: The application meets all functional requirements, handling data correctly and displaying the expected results.
- Usability: The UI is user-friendly, aesthetically pleasing, and functional.
- Code Quality: Your code should be clean, well-organized, and well-documented.
- Innovation: Extra credit for implementing additional features that provide more significant insights and utility to the user.
- Process: Demonstration of good development practices, including version control, testing, and documentation.

Additional Notes

- You are encouraged to use Git for version control. Regular commits with clear messages will be part of your evaluation.
- Consider edge cases for text input, such as handling very short or very long texts, or texts with unusual formatting.
- Your project will be run on a standard CPU configuration; ensure that the model's performance is optimized for such environments.