Technical Home Task for Senior ML Engineer: Building an LLM-based English Improvement Agent

Task Overview

Develop a Python-based application leveraging the OpenAI API and Transformers library to create an agent that assists non-native English speakers in improving their written English. The application should not involve model training or fine-tuning but should focus on effective prompt engineering using the Retrieval-Augmented Generation (RAG) approach (if/when needed). The agent will provide three primary functionalities:

- **write_properly**: Enhances both grammar and style of the input message.
- **write_the_same_grammar_fixed**: Corrects only the grammatical errors in the input message.
- **summarize**: Provides a concise summary of the input message.

The candidate may choose to include a User Interface (UI), though it is not mandatory.

Technical Questions

- **Prompt Engineering for RAG:** Describe how you would design prompts for the RAG model to differentiate between 'write_properly' and 'write_the_same_grammar_fixed' functions. What considerations will you make to ensure the model understands the distinction between style enhancement and mere grammatical correction?

- **API Utilization Strategy**: Explain your strategy for utilizing the OpenAI API in this task. How will you ensure efficient and effective use of the API for different functions, especially considering the potential variability in the length and complexity of user inputs?

- **Handling Ambiguity in User Inputs**: Given that the users are non-native English speakers, their inputs may contain ambiguities or context-specific nuances. How will you design the system to handle such ambiguities, especially in the 'write_properly' function where style improvement is also considered?

- **Summarization Technique**: Discuss the approach you will take to implement the 'Summarize' function. How will you ensure that the essential points are retained while maintaining brevity, and how does this approach leverage the RAG model?

- **Performance Metrics and Evaluation**: What metrics would you use to evaluate the performance of each function in this application? How would you gather feedback or data to improve the system continuously, and what role does prompt engineering play in this process?

Additional Requirements

- Ensure your code is well-structured and adheres to best practices in Python programming.
- Include a README file with clear instructions on how to set up and run the application.
- Document your code sufficiently to explain the logic behind your implementation decisions.

Submission

Your final submission should include:

- Source code files.
- README documentation.
- Any additional materials (e.g., prompt examples, API usage logs) that support your technical decisions and implementations.

Evaluation Criteria

- Efficiency and effectiveness of prompt engineering for the RAG model.
- Innovative use of the OpenAI API and Transformers library.
- Code quality, including readability, structure, and documentation.
- Ability to handle edge cases and ambiguities in user inputs.
- Overall performance of the application based on the outlined functionalities.
- The use of ChatGPT or any IA to complete this test will invalidate the candidature.