Assignment: Text Classification API with FastAPI, Docker, and Hugging Face Transformers

---

Objective:
Develop a RESTful API using FastAPI that utilizes a pre-trained Hugging Face Transformer model to perform text classification (sentiment analysis, emotion classification, etc). The API will be containerized using Docker.

Deadline: Sunday 21 April, 2024

Requirements:

1. **API Development:**
   - Use FastAPI to develop an API with a POST endpoint `/analyze` that accepts JSON input containing a text field and returns the sentiment/emotion/etc of the text.
   - Integrate the model from Hugging Face Transformers.
2. **Model Integration:**
   - Utilize the `transformers` library from Hugging Face to load the pre-trained model.
   - Ensure the API handles the model loading efficiently, possibly using FastAPI's background tasks or startup events.
3. **Containerization:**
   - Create a Dockerfile to containerize your FastAPI application.
   - Ensure the Docker container can be built and run locally, encapsulating all necessary dependencies.

4. **Testing:**
    - Write at least three API tests using FastAPI's TestClient (examples for sentiment):
        - Test sending a positive sentiment text.
        - Test sending a negative sentiment text.
        - Test sending a neutral sentiment text (if applicable).
5. **Deployment:** Deploy the endpoint on hugging-face spaces
6. **Documentation:**
    - Document your API using FastAPI's automatic documentation features (Swagger UI or ReDoc).
    - Provide a README.md file with instructions on how to build and run the Docker container, and how to interact with the API.

Submission:

- Submit your Huggingface space link, GitHub repository link, containing all code, Dockerfile, tests, and documentation.

Scoring Criteria API Assignment (Total: 15 Marks)

1. API Functionality (5 Marks)

- Correct Implementation (3 Marks): The API must correctly implement the POST endpoint `/analyze` that accurately analyzes the sentiment/emotion/etc of the text using the specified model.
- Error Handling (2 Marks): Proper error handling for edge cases, such as invalid input or server errors.

2.Model Integration and Efficiency (3 Marks)

- Model Loading (1.5 Marks): Efficient loading and integration of the Hugging Face model using FastAPI's advanced features (like startup events).
- Response Time (1.5 Marks): API response times are reasonable, demonstrating efficient use of resources.

3.Containerization (3 Marks)

- Dockerfile Configuration (1.5 Marks): Dockerfile is correctly set up to build and run the API within a Docker container, including all necessary dependencies.

- Container Execution (1.5 Marks): The container can be easily built and run, with no errors and minimal setup required from the user.

4.Testing and Reliability (2 Marks)

- API Tests (2 Marks): At least three meaningful tests using FastAPI's TestClient that cover different scenarios and demonstrate the API's robustness.

5.Documentation and Code Quality (2 Marks)

- Code Clarity and Modularity (1 Mark): Code is well-organized, modular, and includes appropriate comments.
- Documentation (1 Mark): The API is well-documented with clear instructions in the README.md on building, running, and interacting with the container and API, including usage of the Swagger UI or ReDoc.