Evaluation Project : Collaborative Sentiment Analysis Pipeline

Overview

In this project, pairs of students will work together to build a sentiment analysis pipeline using a BERT model. The pipeline is broken into three major components:

- **Data Extraction:** Load and prepare raw text data.
- **Data Processing:** Clean and tokenize the text, converting it into the format required for BERT.
- **Model Training & Inference:** Fine-tune a pretrained BERT model for sentiment classification and create an inference script.

Collaboration will be achieved through branching, pull requests, code reviews, and a shared continuous integration (CI) setup.

Repository Structure

```
.
├── src/
│   ├── data_extraction.py
│   ├── data_processing.py
│   ├── model.py
│   └── inference.py
├── tests/
│   └── unit/
│   ├── test_data_extraction.py
│   ├── test_data_processing.py
│   ├── test_model.py
│   └── test_inference.py
├── requirements.txt
└── README.md
```

Key Details & Tasks

1. Data Extraction (Student 1)

Task:

- Write functions in data_extraction.py to load the raw data provided.
- Ensure the function handles errors (missing files, wrong formats) gracefully.
- Testing:

- Create unit tests (tests/unit/test_data_extraction.py) that verify the data is loaded correctly, the DataFrame has expected columns, and edge cases are handled.

2. Data Processing (Student 1 & Student 2)

Task:

- Implement text cleaning and preprocessing in data_processing.py similar to the Kaggle notebook's approach:
- Remove unnecessary characters, lower-case conversion, and normalization.
- Tokenize text using the Hugging Face AutoTokenizer (e.g., for "bert-base-uncased").
- Include splitting of data into training and validation sets.
- Testing**:**
  - Develop tests in tests/unit/test_data_processing.py to ensure the tokenization produces the expected token IDs, and that text cleaning works as intended.

3. Model Training & Inference (Student 2 or Joint Effort)

Task:

- In model.py, load a pretrained BERT model for sequence classification (for example, using AutoModelForSequenceClassification from Hugging Face).
- Fine-tune the model on your sentiment dataset. Use a simple training loop or the Hugging Face Trainer API.
- Create inference.py to allow users to pass in new text and see sentiment predictions.
- Testing**:**
  - Write tests in tests/unit/test_model.py that at least instantiate the model and run a dummy batch through it to ensure it outputs logits with the expected shape.
  - (Bonus Point) Test the end-to-end inference process in tests/unit/test_inference.py.

Collaboration & Git Workflow

- Branching:

    - Student A works on feature-data-extraction and part of feature-data-processing.
    - Student B focuses on feature-model-training (and possibly an inference branch).
- Pull Requests:
    - Each feature branch must be merged into main/master via a pull request.
    - Code reviews are mandatory: each student must review and comment on their partner's PR.
- Commit Messages:
    - Use descriptive messages that clearly reference the task (e.g., "Implement CSV data loader for sentiment data", "Add tokenization function using AutoTokenizer").

Deliverables

Public GitHub Repository - The repository should reflect the structure above with evidence of collaboration (branching, PRs, commit history).

Documentation - A comprehensive README.md that includes setup instructions, usage examples, and a brief description of each component.

Project Report (1-2 pages) - Overview of the chosen approach (inspired by the Kaggle notebook), division of labor, challenges faced, and future improvements. Do not forget to add your names.

Resources

Sentiment Analysis using BERT - https://www.kaggle.com/code/prakharrathi25/sentiment-analysis-using-bert
Github CheatSheet - https://education.github.com/git-cheat-sheet-education.pdf