**Documentation for Sentiment Analysis Using Large Language Models (LLM)**

**Overview**

The objective of this project is to perform sentiment analysis on customer reviews using a Large Language Model (LLM). Initially, the code was designed to utilize OpenAI’s GPT-3.5 API, but due to rate-limiting issues, an alternative approach using Hugging Face's Transformers was explored.

**Approach and Workflow**

1. **Dataset Preparation**:
   * The [dataset](https://www.kaggle.com/datasets/niraliivaghani/flipkart-product-customer-reviews-dataset) containing customer reviews and their corresponding sentiment labels was loaded using Pandas.
   * The reviews were filtered to retain only two columns: "Review" and "Sentiment". The cleaned data was then saved as a new CSV file (review.csv) for further processing.
2. **Data Cleaning**:
   * The reviews underwent text cleaning to improve their quality before feeding them into the sentiment analysis model. The following cleaning steps were applied:
     + Removal of special characters, punctuation, and HTML tags.
     + Conversion of all text to lowercase.
     + Removal of single characters and extra whitespaces.
3. **Dataset Splitting**:
   * The dataset was split into training and testing sets using an 80/20 ratio. The training set consisted of 80% of the total rows, while the remaining 20% formed the test set.
4. **Integration with OpenAI API**:
   * Initially, the project was set up to utilize OpenAI’s GPT-3.5 model for sentiment classification. A function was created to send prompts to the GPT-3.5 model, which would classify reviews into positive or negative sentiments.
   * The model was queried in batches to avoid processing limitations, with each batch containing 50 reviews.
   * An API key was directly provided for accessing the OpenAI API.
5. **Rate Limiting Issue**:
   * While running the code in batches, a RateLimitError occurred due to exceeding the allocated API quota. This error stopped the process, highlighting the need for an alternative solution.
6. **Switch to Hugging Face**:

* Instead of OpenAI’s API, I used Hugging Face’s transformers library with the pre-trained model cardiffnlp/twitter-roberta-base-sentiment.
* I used pipeline from Hugging Face to create a model that does sentiment analysis (positive/negative/neutral).
* I made a custom function get\_sentiment() that processed the model’s output into readable labels: “positive”, “neutral”, or “negative”.

1. **Batch Processing**:

* I applied the sentiment analysis to the entire reviews DataFrame using apply(), which looped through all the reviews and classified them using get\_sentiment().
* The output was stored in a new column, pred\_sentiment, in the DataFrame.

8. **Testing the Model**:

* I tested the sentiment model with a single input: "I love this product! It's amazing and works perfectly."
* It gave an expected result (positive sentiment), showing that the model works.

**Results**:

* The predicted sentiment for each review was added to the DataFrame under pred\_sentiment, which I printed out for inspection.

**How to Run the Code**

1. **Install Dependencies**:
   * Install the required libraries:

pip install transformers datasets pandas

1. **Prepare Dataset**:
   * Load your dataset (reviews and sentiments) using Pandas:

reviews\_df = pd.read\_csv("Dataset-SA.csv")

1. **Clean Reviews**:
   * Clean the reviews using regular expressions to remove unwanted characters.
2. **Run Sentiment Analysis**:
   * Use the Hugging Face pipeline to classify the sentiment of each review:

classifier = pipeline("sentiment-analysis", model="cardiffnlp/twitter-roberta-base-sentiment")

1. **Apply the Sentiment Function**:
   * Apply the get\_sentiment() function to the "Review" column of your dataset:

reviews\_df['pred\_sentiment'] = reviews\_df['Review'].apply(get\_sentiment)

1. **Inspect the Results**:
   * Print the DataFrame to see the predicted sentiments.
2. **Test with Custom Input**:
   * Test the model with a single string to check if it works:

input = "I love this product! It's amazing and works perfectly."

result = classifier(input)

print(result)