Phase-2 “*Sentiment Analysis on Twitter Data Using Spark and Ray*” Report

- **Vani Seth, Rayhan Mahady, Divya Reddy**

The objective of Phase 2 of the project is to build a scalable and modular sentiment analysis pipeline that processes raw Twitter data, generates sentiment labels using a Large Language Model (LLM), and trains a classification model using distributed machine learning. The pipeline integrates Apache Spark for data processing, Google Gemini API for auto-labeling, and Ray with PyTorch for distributed model training. This project addresses the challenge of working with unlabeled real-world text data by using data engineering techniques with scalable machine learning.

Design and Implementation:

Stage 1: Data Cleaning (Apache Spark): Raw tweets (in JSON lines format) are ingested using Spark, and relevant fields (string id, full text) are extracted. A UDF cleans the text by removing URLs, mentions, hashtags, and punctuation, with a length threshold applied to filter out low-quality tweets.

Stage 2: Vocabulary Generation (Spark RDDs): A subset of the cleaned data is used to compute word frequencies via RDD transformations. Words are filtered based on frequency and compiled into a vocabulary with special tokens and are saved as a .pkl file for use during model training.

Stage 3: Labeling Tweets with Gemini API: Spark's “mapPartitions” enables parallel batched calls to the Gemini API, prompting the LLM to classify tweet sentiment (Positive, Negative, Neutral). The response is parsed and returned in a structured JSON format. Error handling ensures robustness against rate limits or invalid content.

Stage 4: Storing Labeled Data: Successfully labeled tweets are saved in the Parquet format, optimizing storage and downstream data access. This dataset includes the tweet ID, cleaned text, and Gemini-generated sentiment label.

Model Training with Ray:

We trained an LSTM model with Attention Mechanism. The model includes:

- Word Embedding Layer
- Bi-directional LSTM
- Additive Attention Layer
- Dropout
- Fully Connected Layer

Distributed Training with Ray: Ray initializes a distributed setup with multiple workers. The labeled Parquet data and vocabulary are loaded using ray.data.read_parquet() method. The data is split into train/val/test sets.

A TorchTrainer manages the training lifecycle:

- Manual batching handles variable-length input via a custom collate_batch() function.
- Checkpointing and metrics reporting are managed via Ray's built-in tools.
- DDP (Distributed Data Parallel) support is implicitly handled.

Model Evaluation: After training, the best checkpoint is restored, and the model is evaluated on the test set. Metrics like accuracy, F1-score, and a classification report are computed. Visualizations include confusion matrix and loss/accuracy curves over epochs.

Visualizations and Results:

The sentiment classification model was evaluated on the test set using standard performance metrics. The model achieved an accuracy of 66.67%, with a weighted F1 score of 0.662. This indicates that the model performs reasonably well across all sentiment classes, accounting for class imbalance. However, the macro-averaged F1 score was 0.596. This suggests that while the model performs adequately on average, it struggles with underrepresented classes, likely due to class imbalance or label noise.

The results are interpreted as confusion matrix and accuracy and loss curves.

Conclusion:

This project demonstrates a scalable sentiment analysis pipeline that combines Apache Spark for distributed data processing, the Gemini API for automated sentiment labeling, and Ray for efficient distributed training. By leveraging batching and fault-tolerant infrastructure, the system efficiently processes large datasets and trains models with minimal manual effort. The integration of these tools enables fast sentiment classification.