Social Media Sentimental Analysis

Literature Review:

- Sentiment Analysis on Social Media by Jyoti Yadav
- Social Media Sentiment Analysis: A Comprehensive Analysis

Introductions

This project is focused on implementing a **Social Media Sentiment Analysis** system using a distributed system infrastructure that integrates **Hadoop** and **Spark** for efficient data processing and analytics. The goal is to build a scalable, reliable solution to analyze sentiments from social media platforms like **Twitter or Reddit.**

Tools and Technology

data collection

Name	Type	Link	Description
Twitter API	API	https://developer.x.com/en/docs/x-api	using twitter API to fetch tweet but it has rate limit and only 3200 free per month
twint	Scrapping	https://github.com/twintproject/twint	scrapping tools to scrape tweet without limits from twitter
Twitter Kaggle	Kaggle dataset	https://www.kaggle.com/datasets/kazanova/sentiment140	This dataset contain 1.6m tweet from the use of API of twitter
Reddit Kaggle	Kaggle dataset	https://www.kaggle.com/datasets/prakharrathi25/reddit-data-huge	Reddit dataset that has sub-reddit to choose from a very huge selection and big, good for NLP
tweepy	python library for twitter api		

Praw	python library for reddits api		

data processing and storing

Name	Type
Spark	processing
Hadoop	storing

Pipeline v1:

- [x] ~~Extract data~~
  - [x] ~~fetch data from reddits, twitter and kaggle~~
- [ ] Processed data
  - [x] ~~Convert text to lowercase~~
  - [x] ~~remove most common stop words such as a, about, above...~~
  - [x] ~~remove non character texts such as punctuations and emojis from text~~
  - [x] ~~filter and remove repeated words, URLs, and number from texts~~
  - [x] ~~tokenization was done to convert texts into tokens, which is to split sentences into smaller units or words. So meaning can assign to word more easily~~
  - [x] ~~Stemming was done to extract base form of the words by removing affixes from them (EX: words such as “likes” , “likely” and “liked” returned as “like” after stemming)~~
  - [ ] Term Frequency-Inverse Document Frequency Vectorizer (TF-IDF) was pre-owned to assess how relevant a term is in the corpus/text data, where TF-IDF vectorization is process for calculating the TF-IDF score for every word.
- [ ] Sentiment Analysis
  - [x] ~~Perform sentimental analysis using VADAER, TextBlob, BERT~~
  - [x] ~~add sentiment scores to each post (positive, negative, neutral)~~
- [ ] Feature Extraction
  - [ ] use TF-IDF or Word2Vec to convert text into numerical features
- [ ] Clustering
  - [ ] apply K-Means or DBSCAN to group base on sentiment or topic similarity
- [ ] Topic Modeling
  - [ ] apply LDA to find underlying topics within clusters
  - [ ] combine sentimental analysis and LDA to visualize how sentiment changes across topics
- [ ] Visualization

- [ ] use t-SNE or PCA for visualize clusters
- [ ] Visualize sentiment distribution within each topic using **pie charts** or **bar graphs**.

Methodology

Cluster setup Hadoop + Spark

command to start hadoop cluster:

- On hadoop-master-node:

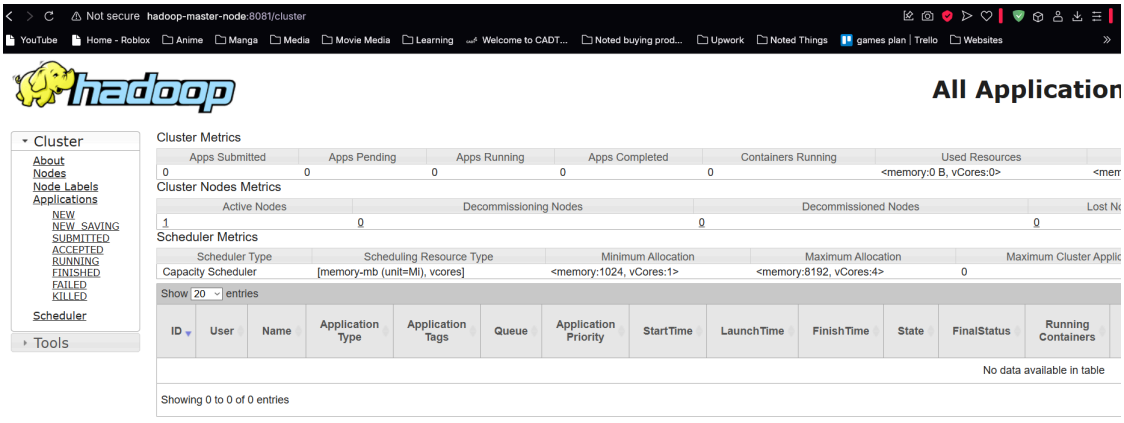
```
start-dfs.sh
start-yarn.sh
start-all.sh // for stop service
```

- on hadoop-slave-node:

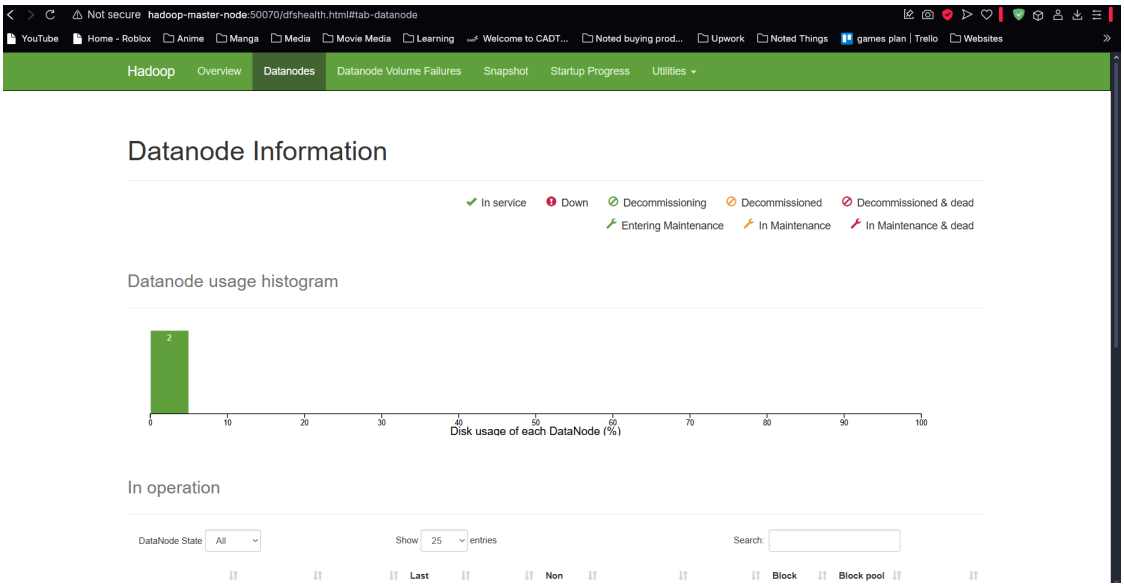
```
start-dfs.sh
start-yarn.sh
start-all.sh // for stop service
```

- URL:

http://hadoop-master-node:8081



- http://hadoop-master-node:50070



command to start and stop spark:

- on master node (we use the same node as hadoop):

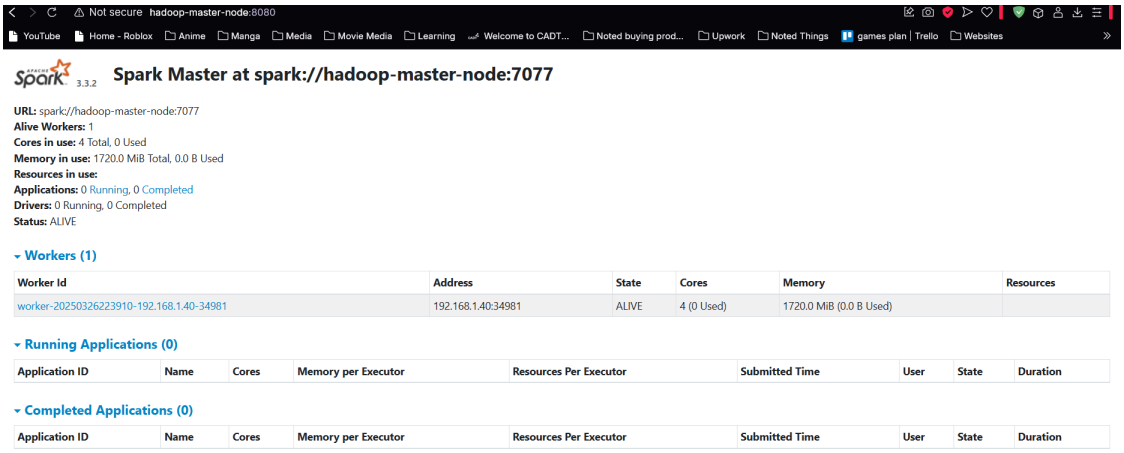
```
$SPARK_HOME/sbin/start-master.sh
$SPARK_HOME/sbin/stop-master.sh

$SPARK_HOME/sbin/start-all.sh // start all both master and worker
$SPARK_HOME/sbin/stop-all.sh
```

- on worker slave node (use same as hadoop):

```
$SPARK_HOME/sbin/start-worker.sh spark://hadoop-master-node:7077
$SPARK_HOME/sbin/stop-worker.sh
```

- http://hadoop-master-node:8080



- Now download spark and hadoop for the main window PC (client)
- this is the path

```
$env:SPARK_HOME="C:\spark-3.5.5-bin-hadoop3"
$env:HADOOP_HOME="C:\hadoop-3.4.1"
$env:JAVA_HOME="C:\jdk-11.0.26.4-hotspot"
$env:PATH="$env:SPARK_HOME\bin;$env:HADOOP_HOME\bin;$env:PATH"
$env:PYSPARK_PYTHON="python"
```

▼ Data collection and cleaning

▼ Pre-processing

▼ Sentiment analysis

This step is use to give values score or sentiment score to the words in the sentence or in this case the word that is already tokenized.

We have 2 option to pick from which are the VADER and textBlob.

Both are valid and we try both but what we go with is VADER which is more for social media like tweet, emojies and slang. For textBlob is for general english grammar and sentences.

Sentiment Analysis using VADER

first we need to install the python library for it

```
pip install vaderSentiment
```

Sentiment Analysis using textBLOB

first we need to install the python library for it

```
pip install textblob
python -m textblob.download_corpora
```

▼ Data Pipeline

For your sentiment analysis pipeline, the process generally depends on your data source, the tools you are using, and your desired output. Here's a typical structure for a pipeline with Hadoop, Spark, and sentiment analysis:

Pipeline Steps:

1. **Data Collection**:

   we start by gathering the text data from your source (e.g., social media, news, customer reviews, etc.). If your data is coming from a large source, like Twitter or Reddit, we can use APIs to collect data. This step could also involve loading data from existing datasets in formats like CSV, JSON, or others.

2. **Data Preprocessing**:

   - **Cleaning**: Remove unwanted characters, URLs, special symbols, etc.
   - **Tokenization**: Split the text into individual words or tokens.
   - **Stop Word Removal**: Remove common words that don't contribute much to sentiment analysis (e.g., "the", "and", etc.).
   - **Normalization**: Lowercasing, stemming, or lemmatization to reduce words to their base form.

   This step can be done using Spark for large-scale data processing, leveraging distributed computing power to handle the data more efficiently. We can process the data either on the Hadoop filesystem (HDFS) or on a local system.

3. **Store Preprocessed Data (Optional)**:

   After preprocessing, We can store the cleaned and processed data back into HDFS or a distributed file system for easy access during the next step.

4. **Sentiment Analysis with Spark**:

   - **Model Selection**: Choose a sentiment analysis model (like Naive Bayes, SVM, or use pre-trained models such as VADER or BERT).
   - **Text Vectorization**: Convert the text data into numerical features using methods like TF-IDF, Word2Vec, or embeddings.
   - **Model Prediction**: Use the selected model to predict sentiment (positive, negative, neutral) for each piece of text.

   This part of the pipeline can be done using Spark's machine learning libraries like **MLlib** or **Spark NLP**.

5. **Store Sentiment Results**:

   After performing sentiment analysis, store the results (e.g., sentiment scores or labels) in HDFS or a database for further use or visualization.

6. **Data Visualization / Reporting**:

   We can visualize the sentiment analysis results using a tool like Tableau, Power BI, or even Spark's built-in visualization tools. This helps in understanding trends, patterns, and sentiments over time.

Example of Your Flow:

1. **Collect data** → 2. **Preprocess data** (Spark or Hadoop) → 3. **Store in HDFS** (optional) → 4. **Sentiment Analysis** (Spark, MLlib or NLP) → 5. **Store results in HDFS** → 6. **Visualization** (for reporting or business intelligence).

Key Points:

- If the data is massive and we need scalability and storage, we leverage **Hadoop (HDFS)** for storing raw or processed data and let **Spark** handle the heavy computations.
- **Spark** is great for distributed processing and performing the sentiment analysis task.

In summary, you can either:

- **Process the data in Spark**, and **input it directly into HDFS**, then run sentiment analysis in Spark.
- **Store raw data** in **HDFS**, then read and process it with **Spark**, running sentiment analysis afterward.