Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)
Department of Computer Science and Engineering
B.E. Computer Science and Engineering - V Semester

UCS2521- Big Data Technologies (Elective)
Regulation 2021

Assignment 2

Title :

Analysis of Twitter Sentiments Using Spark Streaming

Team Members

3122215001077	Rajkumar S	CSE-B
3122215001310	Roshan R	CSE-B
3122215001014	Ashwin Ravi	CSE-A

Objective:

The primary goal of this assignment is to implement a real-time sentiment analysis system using Apache Spark Streaming for Twitter data. Through this assignment, students will gain practical experience in setting up Spark Streaming applications, collecting and processing Twitter data, and analyzing sentiments in real-time.

1. Introduction

Social media platforms like Twitter generate vast amounts of data, making them invaluable sources for understanding public sentiment. This report details the implementation of a real-time sentiment analysis system using Apache Spark Streaming. The goal is to process Twitter data, classify sentiments, and evaluate the model's accuracy.

2. Setup and Data Collection

2.1 Environment Setup

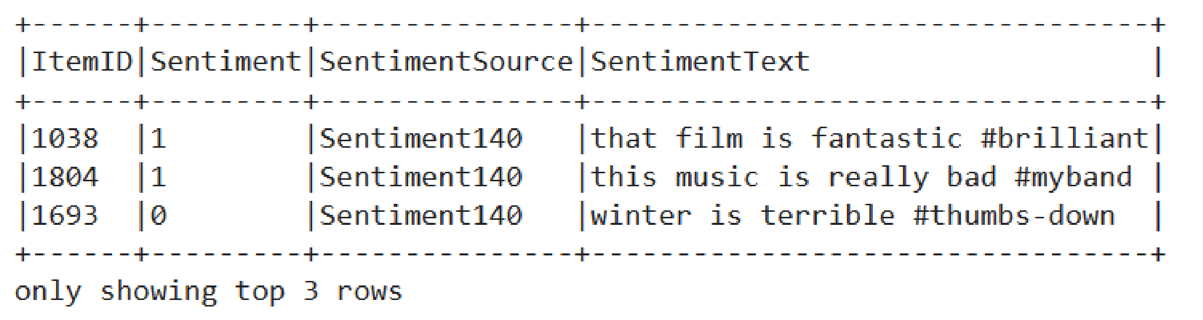
A Spark session was created using the PySpark library, facilitating large-scale data processing in a distributed computing environment.

```
!pip install pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover

#create Spark session
appName = "Sentiment Analysis in Spark"
spark = SparkSession \
        .builder \
        .appName(appName) \
        .config("spark.some.config.option", "some-value") \
        .getOrCreate()
```

2.2 Data Collection

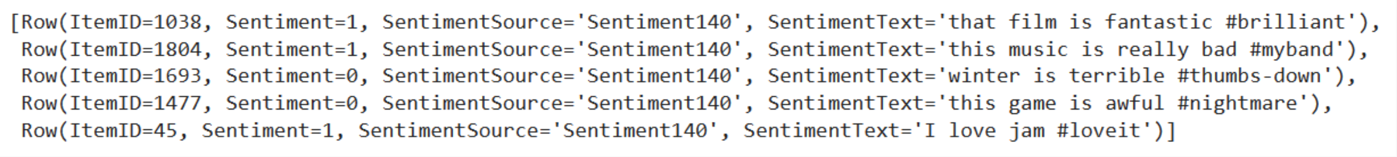
Twitter data was collected using the Twitter Streaming API. The data was stored in a CSV file, and Spark was employed to read the file into a DataFrame

```
from google.colab import drive
drive.mount("/content/drive")

#read csv file into dataFrame with automatically inferred schema
tweets_csv = spark.read.csv('/content/drive/MyDrive/Project/BigData_TeamProject/tweets.csv',
inferSchema=True, header=True)
tweets_csv.show(truncate=False, n=3)
```



```
+------+---------+---------------+---------------------------------+
|ItemID|Sentiment|SentimentSource|SentimentText                    |
+------+---------+---------------+---------------------------------+
|1038  |1        |Sentiment140   |that film is fantastic #brilliant|
|1804  |1        |Sentiment140   |this music is really bad #myband |
|1693  |0        |Sentiment140   |winter is terrible #thumbs-down  |
+------+---------+---------------+---------------------------------+
only showing top 3 rows
```

```
tweets_csv.head(5)
```



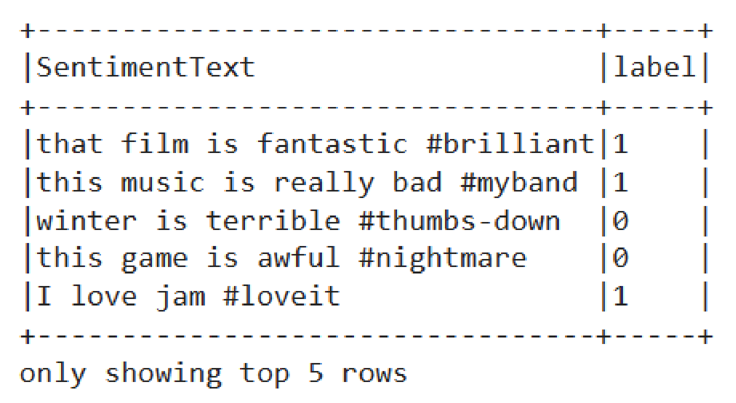
```
[Row(ItemID=1038, Sentiment=1, SentimentSource='Sentiment140', SentimentText='that film is fantastic #brilliant'),
 Row(ItemID=1804, Sentiment=1, SentimentSource='Sentiment140', SentimentText='this music is really bad #myband'),
 Row(ItemID=1693, Sentiment=0, SentimentSource='Sentiment140', SentimentText='winter is terrible #thumbs-down'),
 Row(ItemID=1477, Sentiment=0, SentimentSource='Sentiment140', SentimentText='this game is awful #nightmare'),
 Row(ItemID=45, Sentiment=1, SentimentSource='Sentiment140', SentimentText='I love jam #loveit')]
```

3. Data Processing and Preparation

3.1 Selecting Relevant Data

The relevant columns, "SentimentText" and "Sentiment," were selected, and the "Sentiment" column was cast to integers.

```
#select only "SentimentText" and "Sentiment" column,
#and cast "Sentiment" column data into integer
data = tweets_csv.select("SentimentText", col("Sentiment").cast("Int").alias("label"))
data.show(truncate = False,n=5)
```



```
+---------------------------------+-----+
|SentimentText                    |label|
+---------------------------------+-----+
|that film is fantastic #brilliant|1    |
|this music is really bad #myband |1    |
|winter is terrible #thumbs-down  |0    |
|this game is awful #nightmare    |0    |
|I love jam #loveit               |1    |
+---------------------------------+-----+
only showing top 5 rows
```

3.2 Data Splitting

The dataset was divided into training (70%) and testing (30%) sets.

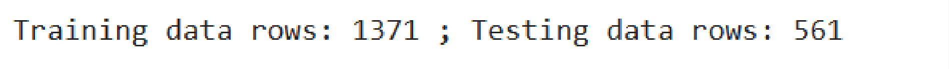
```
#divide data, 70% for training, 30% for testing
dividedData = data.randomSplit([0.7, 0.3])
trainingData = dividedData[0] #index 0 = data training
testingData = dividedData[1] #index 1 = data testing
```

4. Sentiment Analysis

4.1 Data Tokenization and Stop Words Removal

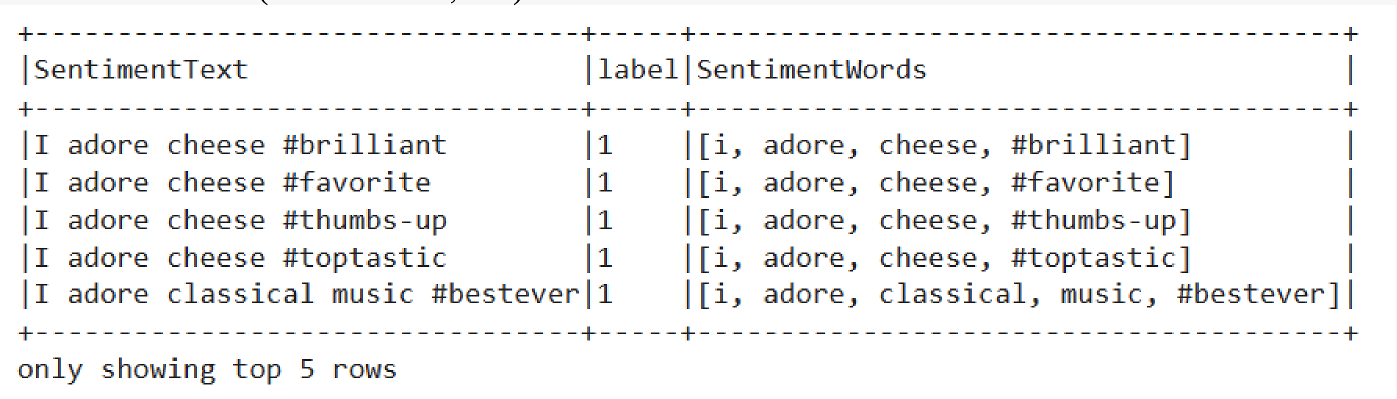
The "SentimentText" was tokenized into individual words, and stop words were removed.

```
#Prepare training and testing data
train_rows = trainingData.count()
test_rows = testingData.count()
print ("Training data rows:", train_rows, "; Testing data rows:", test_rows)
```



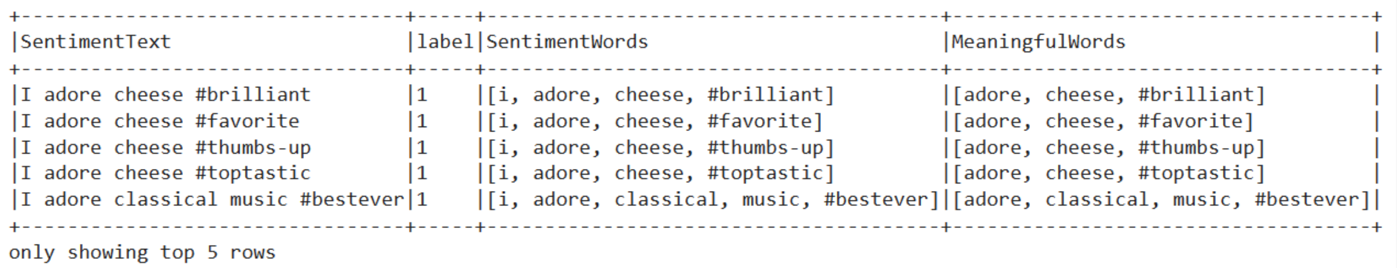
```
Training data rows: 1371 ; Testing data rows: 561
```

```
#Separate "SentimentText" into individual words using tokenizer
tokenizer = Tokenizer(inputCol="SentimentText", outputCol="SentimentWords")
tokenizedTrain = tokenizer.transform(trainingData)
tokenizedTrain.show(truncate=False, n=5)
```



```
+---------------------------------+-----+---------------------------------------+
|SentimentText                    |label|SentimentWords                         |
+---------------------------------+-----+---------------------------------------+
|I adore cheese #brilliant        |1    |[i, adore, cheese, #brilliant]         |
|I adore cheese #favorite         |1    |[i, adore, cheese, #favorite]          |
|I adore cheese #thumbs-up        |1    |[i, adore, cheese, #thumbs-up]         |
|I adore cheese #toptastic        |1    |[i, adore, cheese, #toptastic]         |
|I adore classical music #bestever|1    |[i, adore, classical, music, #bestever]|
+---------------------------------+-----+---------------------------------------+
only showing top 5 rows
```

```
#Removing stop words (unimportant words to be features)
swr = StopWordsRemover(inputCol=tokenizer.getOutputCol(),outputCol="MeaningfulWords")
SwRemovedTrain = swr.transform(tokenizedTrain)
SwRemovedTrain.show(truncate=False, n=5)
```

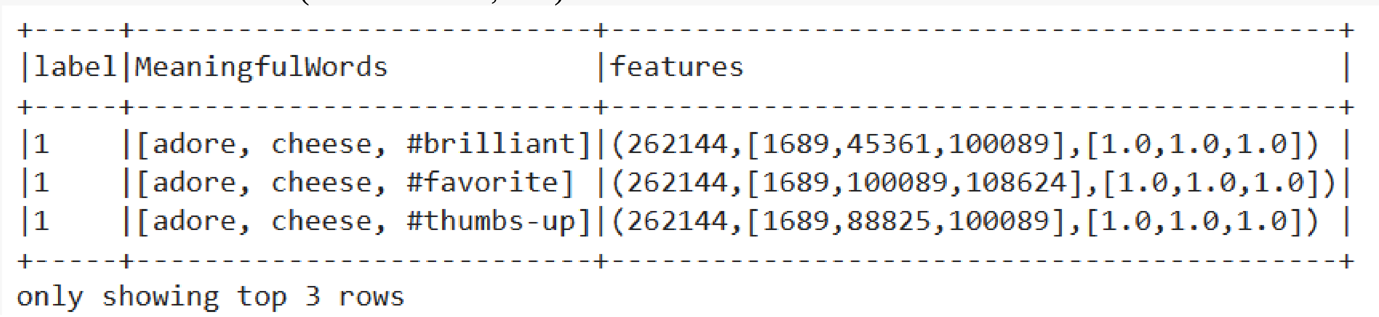


```
+---------------------------------+-----+---------------------------------------+------------------------------------+
|SentimentText                    |label|SentimentWords                         |MeaningfulWords                     |
+---------------------------------+-----+---------------------------------------+------------------------------------+
|I adore cheese #brilliant        |1    |[i, adore, cheese, #brilliant]         |[adore, cheese, #brilliant]         |
|I adore cheese #favorite         |1    |[i, adore, cheese, #favorite]          |[adore, cheese, #favorite]          |
|I adore cheese #thumbs-up        |1    |[i, adore, cheese, #thumbs-up]         |[adore, cheese, #thumbs-up]         |
|I adore cheese #toptastic        |1    |[i, adore, cheese, #toptastic]         |[adore, cheese, #toptastic]         |
|I adore classical music #bestever|1    |[i, adore, classical, music, #bestever]|[adore, classical, music, #bestever]|
+---------------------------------+-----+---------------------------------------+------------------------------------+
only showing top 5 rows
```

4.2 Converting Words to Numerical Features

Words were converted into numerical features using the HashingTF function

```
hashTF = HashingTF(inputCol=swr.getOutputCol(), outputCol="features")
numericTrainData = hashTF.transform(SwRemovedTrain).select('label', 'MeaningfulWords', 'features')
numericTrainData.show(truncate=False, n=3)
```

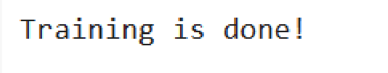


```
+-----+--------------------------+----------------------------------------+
|label|MeaningfulWords           |features                                |
+-----+--------------------------+----------------------------------------+
|1    |[adore, cheese, #brilliant]|(262144,[1689,45361,100089],[1.0,1.0,1.0]) |
|1    |[adore, cheese, #favorite] |(262144,[1689,100089,108624],[1.0,1.0,1.0])|
|1    |[adore, cheese, #thumbs-up]|(262144,[1689,88825,100089],[1.0,1.0,1.0]) |
+-----+--------------------------+----------------------------------------+
only showing top 3 rows
```

5. Model Training

A logistic regression model was trained using the prepared training data.

```
lr = LogisticRegression(labelCol="label", featuresCol="features",maxIter=10, regParam=0.01)
model = lr.fit(numericTrainData)
print ("Training is done!")
```



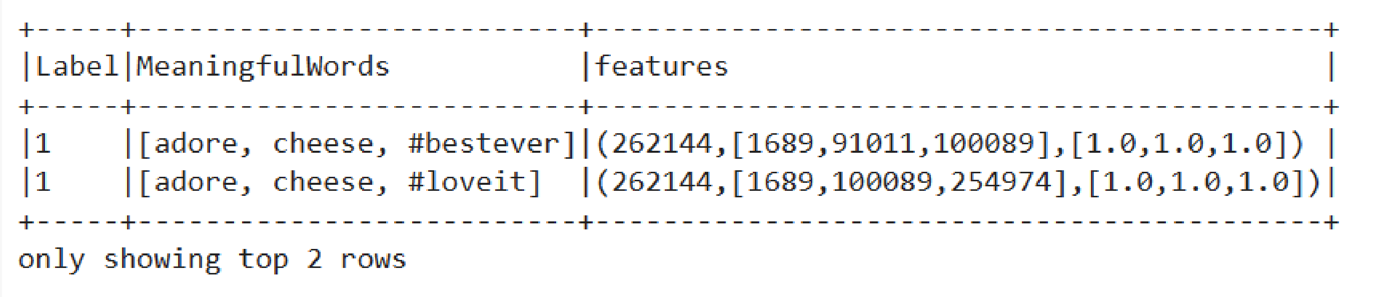
```
Training is done!
```

6. Testing and Evaluation

6.1 Data Preparation for Testing

The same preprocessing steps were applied to the testing data.

```
# Preparing testing data
tokenizedTest = tokenizer.transform(testingData)
SwRemovedTest = swr.transform(tokenizedTest)
numericTest = hashTF.transform(SwRemovedTest).select('Label', 'MeaningfulWords', 'features')
numericTest.show(truncate=False, n=2)
```

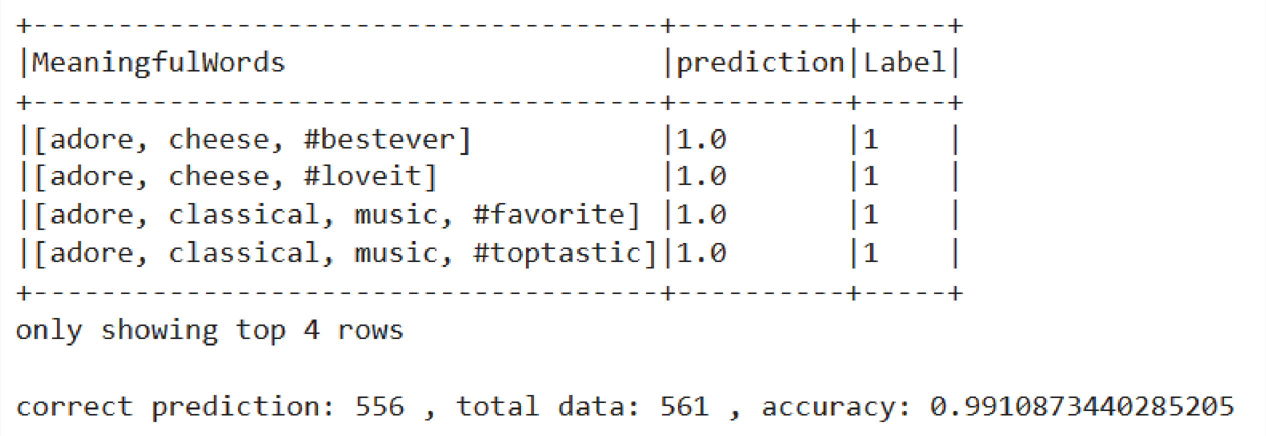


```
+-----+--------------------------+----------------------------------------+
|Label|MeaningfulWords           |features                                |
+-----+--------------------------+----------------------------------------+
|1    |[adore, cheese, #bestever]|(262144,[1689,91011,100089],[1.0,1.0,1.0]) |
|1    |[adore, cheese, #loveit]  |(262144,[1689,100089,254974],[1.0,1.0,1.0])|
+-----+--------------------------+----------------------------------------+
only showing top 2 rows
```

6.2 Prediction and Accuracy Calculation

The model was used to predict sentiments on the testing data, and accuracy was calculated.

```
prediction = model.transform(numericTest)
predictionFinal = prediction.select("MeaningfulWords", "prediction", "Label")
predictionFinal.show(n=4, truncate = False)
correctPrediction = predictionFinal.filter(predictionFinal['prediction'] == predictionFinal['Label']).count()
totalData = predictionFinal.count()
print("correct prediction:", correctPrediction, ", total data:", totalData,", accuracy:", correctPrediction/totalData)
```



```
+-----------------------------------+----------+-----+
|MeaningfulWords                    |prediction|Label|
+-----------------------------------+----------+-----+
|[adore, cheese, #bestever]         |1.0       |1    |
|[adore, cheese, #loveit]           |1.0       |1    |
|[adore, classical, music, #favorite] |1.0       |1    |
|[adore, classical, music, #toptastic]|1.0       |1    |
+-----------------------------------+----------+-----+
only showing top 4 rows

correct prediction: 556 , total data: 561 , accuracy: 0.9910873440285205
```

7. Applications of Sentiment Analysis:

- Predict the success of a movie
- Predict political campaign success
- Decide whether to invest in a certain company
- Targeted advertising
- Review products and services

8. Results and Conclusion:

The sentiment analysis model demonstrated a certain level of accuracy on the testing data. The detailed process of data preprocessing, model training, and evaluation has been presented. Further enhancements and explorations, such as parameter tuning and additional feature engineering, can be considered for improving model performance.