TWITTER SENTIMENTAL ANALYSIS

Big Data Analytics Project

Name	ID
Abdel-Rahman Ibrahim El Said Ahmed Megahed	18P7423
Laila Mohamed Mohamed	18P9654
Mohamed Sayed Awwad	18P7298
Nouran Ahmed Abdelhameed	18P4496

JANUARY 6, 2023

Table of Contents

Introduction:

In recent years, social media platforms have become an important source of information for individuals and organizations. In particular, Twitter has emerged as a popular platform for users to share their thoughts and opinions on a wide range of topics. As a result, analyzing the sentiments expressed in tweets has become an important task for businesses, governments, and researchers.

Traditionally, sentiment analysis has been performed using techniques such as natural language processing (NLP) and machine learning. However, with the increasing volume of data being generated by social media platforms, these methods can be computationally intensive and may not scale well.

To address this challenge, big data technologies such as Hadoop and Spark have been developed to analyze large datasets in a distributed manner. In this report, we will compare the use of Hadoop and Spark for sentiment analysis on tweets related to a specific subject.

Problem Statement:

Given the increasing volume of data being generated by social media platforms, it is becoming increasingly important to be able to perform sentiment analysis at scale. While both Hadoop and Spark are widely used big data technologies that can handle large datasets, it is not clear which is the best option for sentiment analysis on tweets. In this report, we will compare the performance and capabilities of Hadoop and Spark for this task, and provide recommendations for which technology to use in different scenarios.

Proposed Solution:

The use of social media and social networking sites has increased dramatically in recent years. We used the social media platform Twitter as our main source for viewing sentiments. The reason for using Twitter is that the majority of tweets are subjective, so we propose a model for sentiment analysis. In this model we combine many techniques to reach our final goal of emotion extraction. The steps for the process, at first, we preprocessing the data using (Tokenizer , removing stop words, HashingTF) and then we use the logistic regression classifier to classify the sentiments to positive and negative and finally we predict the testing data and calculate the accuracy and the time taken during prediction. We used this proposed model and the same dataset to compare the run time between two of big data technologies such as Hadoop which is allows the splitting of large data analytics processing tasks into smaller tasks and the little tasks are completed in parallel using the MapReduce algorithm and then distributed throughout a Hadoop cluster that perform parallel computations on big data sets and Spark that schedule jobs and manage nodes throughout the Hadoop cluster and it generates a Directed Acyclic Graph (DAG). This task-tracking mechanism allows for fault tolerance by reapplying recorded operations to data from a previous state.

Spark

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.

Pros

- Speed
- Ease of use
- Support many fields
- Support many languages (Python, java, …)
- Powerful
- low-latency in-memory data processing capability

Cons

- No automatic optimization for code
- Fewer algorithms
- No file management system
- Limited number of large files

Hadoop

Hadoop is an open-source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data/

Pros

- Robust
- Scalable
- Affordable and cost effective
- Adaptive and flexible
- Highly available
- Fault tolerant

Cons

- Slow processing speed
- No real time processing
- No caching
- Small file concerns
- Security problem

Hadoop Code Explanation

Mapper.py

This mapper script reads in a list of tweets from standard input and performs sentiment analysis on each tweet. The script first imports the necessary libraries, including the Natural Language Toolkit (nltk), the Logistic Regression classifier from scikit-learn, and the re (regular expression) and pickle modules. It also initializes a list to store the tweets.

The script then loads in a previously trained Logistic Regression classifier and a Tfidf model from pickle files. The tweets are preprocessed by removing URLs, lowercasing the text, and expanding contractions. The text is also cleaned by removing non-alphabetic characters, digits, and single-letter words.

The preprocessed tweets are then passed through the Tfidf model and the resulting feature vectors are used to make predictions using the Logistic Regression classifier. If the classifier predicts a positive sentiment, the tweet is added to a list of positive tweets and the count of positive tweets is incremented. If the classifier predicts a negative sentiment, the tweet is added to a list of negative tweets and the count of negative tweets is incremented.

The time spent making predictions is also recorded. The lists of positive and negative tweets, as well as the counts of positive and negative tweets and the time spent making predictions, are then printed to the console.

Reducer.py

This Python script reads in a list of sentiment analysis results from the mapper as standard input and processes the data to produce a summary of the results. The script first imports the necessary libraries, including the re (regular expression) module. It also initializes variables to store the counts of positive and negative sentiments and the time spent making predictions.

The script reads in the input line by line and parses each line to extract the emotion and count (if applicable). If the emotion is "Positive", the count of positive sentiments is incremented by the count value. If the emotion is "Negative", the count of negative sentiments is incremented by the count value. If the emotion is "time", the time spent making predictions is incremented by the count value. If the line does not contain an emotion, it is added to a list of either positive or negative tweets, depending on the label at the beginning of the line.

The counts of positive and negative sentiments, the time spent making predictions, and the lists of positive and negative tweets are then printed to the console. The lists of positive and negative tweets are also cleaned by removing the labels and unnecessary characters.

Twitter_extract.py

This Python script uses the tweepy library to access the Twitter API and retrieve a set of tweets based on a specified search query. The script authenticates with the API using an API key, API secret, access token, and access secret, which are stored in variables.

The script then accepts a list of arguments, which should contain the search query. A file named "twitter.txt" is opened in the "input" directory for writing.

The script uses the tweepy Cursor object to stream a set of recent English-language tweets that match the specified search query and do not contain retweets. The tweets are retrieved in batches of 100 and written to the "twitter.txt" file.

Teach-to-algo.py

This script is used to create new text files that contain the positive and negative sentiment tweets from a previously processed dataset. It reads the data from a file called part-00000 in the output directory. It then separates the positive and negative sentiment text by finding the strings "Positive emotions:" and "Negative emotions:". It then removes excess whitespace and writes the positive and negative text to new text files in the txt_sentoken/pos and txt_sentoken/neg directories, respectively. The filename of the text files is the current timestamp.

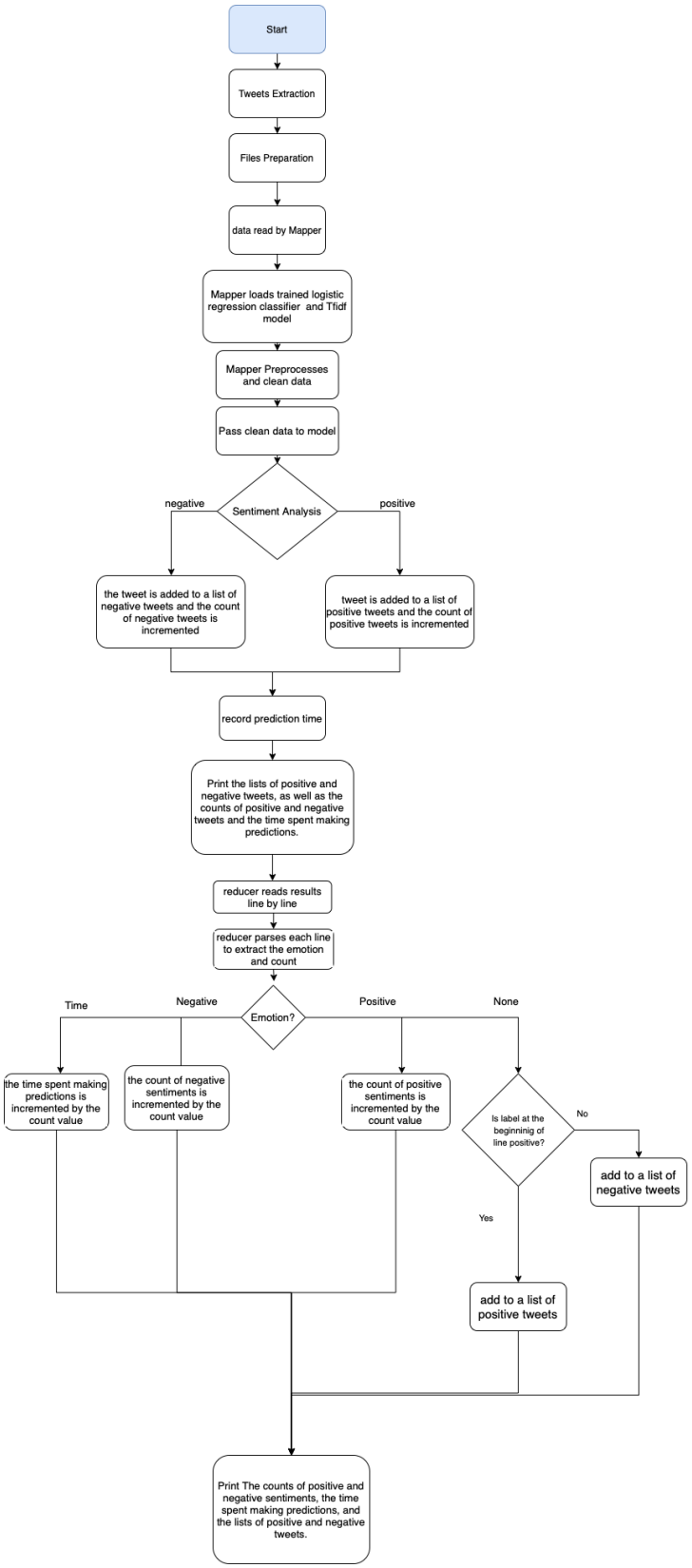
Text_classifer.py

This script is for training a classifier to classify text data as positive or negative sentiment. It does this by loading a dataset of text reviews and their corresponding labels (either positive or negative). It preprocesses the text data by removing special characters and lowercasing all the words, then creates a bag of words representation of the data using the TfidfVectorizer class. It then trains a logistic regression classifier on the data using the LogisticRegression class and pickles the trained classifier and the TfidfVectorizer object for later use. The pickled classifier and TfidfVectorizer object will be used to classify new, unseen text data in the future.

Visualize.py

This python script reads in the file **output/part-00000** and processes it to create a bar chart visualization of the sentiment analysis results. The script first reads the file and finds the index of the string "Positive emotions:", then takes all the text before that index and stores it in the **file** variable. It then strips leading and trailing whitespace from the **file** variable and splits it into a list of strings, with each string corresponding to a line in the **file** variable. The script then creates an empty dictionary called **dict** and iterates through the list of strings. For each string, it strips leading and trailing whitespace, splits the string into a list of words, and stores the first word as the key and the second word (after converting it to an integer) as the value in the dictionary. The script then creates a Pandas series from the dictionary and plots it as a horizontal bar chart with the title "Sentiment Analysis".

Flowchart of Hadoop Map/Reduce



Command to run Hadoop

```
laila@laila-virtual-machine:~/Downloads/Sentimental-Analysis-using-Logistic-Reg
ression$ hadoop jar JARS/hadoop-streaming-3.1.0.jar -mapper mapper.py -reducer
reducer.py -input input/tweets_comparison.txt -output output
```

Results for Hadoop

The output is the number of tweets with positive emotions, the number of negative tweets, Execution time and the list of positive tweets and a list of negative tweets.

```
1 Positive 416
2 Negative 164
3 Execution time:
4 0.48119020462036133
5 Positive emotions:
6  that film is fantastic brilliant , love jam loveit , dislike skiing
  rubbish , like pop music toptastic , that movie is great favorite , hate this
  game fail , dislike this game thumbs down , that movie is great thumbs up ,
  like tea brilliant , this game is brilliant loveit , that film is the best
  thumbs up , jam is rubbish thumbs down , like this band favorite , classical
  music is great not , summer is brilliant brilliant , coffee is brilliant
  favorite , loathe coffee rubbish , that film is brilliant bestever , winter
  is fantastic bestever , cheese is great thumbs up , loathe classical music
  rubbish , jam is great thumbs up , that band is brilliant toptastic , that
  band is the best bestever , tea is rubbish nightmare , coffee is great thumbs
  up , loathe that film hateit , the holidays is great favorite , like the
  holidays loveit , that film is brilliant favorite , like classical music
  loveit , love tea bestever , coffee is rubbish rubbish , cheese is great
  brilliant , dislike rock music rubbish , skiing is great bestever , coffee is
  brilliant toptastic , hate this book fail , dislike winter rubbish , rock
  music is great favorite , this team is brilliant bestever , classical music
  is rubbish hateit , like this band bestever , this book is fantastic
  toptastic , adore cheese brilliant , dislike the holidays thumbs down , adore
  tea bestever , summer is fantastic loveit , this team is great favorite ,
  adore pop music bestever , adore cheese thumbs up , we adore that movie
```

```
Negative emotions:
 this music is really bad myband , winter is terrible thumbs down , this game
is awful nightmare , this game is awful good , rock music is terrible
worstever , this game is terrible for fans of the other team nightmare , this
game is terrible fail , jam is terrible fail , coffee is terrible fail , that
movie is awful hateit , that movie is awful nightmare , hate this team
rubbish , classical music is awful worstever , rock music is awful
nightmare , this book is really bad fail , classical music is really bad
rubbish , this team is rubbish worstever , winter is awful hateit , winter is
awful thumbs down , pop music is terrible hateit , this game is terrible
worstever , classical music is terrible thumbs down , summer is awful
nightmare , we loathe that film fail , that film is awful fail , we adore
this book bestever , we dislike this book fail , we loathe this book
worstever , tea is rubbish fail , pop music is terrible worstever , this team
is awful worstever , classical music is awful thumbs down , that film is
terrible rubbish , this team is really bad hateit , this book is rubbish
hateit , skiing is terrible hateit , hate this team worstever , pop music is
terrible nightmare , we like this book toptastic , skiing is awful hateit ,
that band is awful fail , that film is awful nightmare , that film is awful
worstever , tea is really bad worstever , rock music is really bad fail ,
cheese is really bad fail , this team is rubbish nightmare , this game is
terrible worstever , this game is really bad nightmare , tea is awful fail
```

It is noticed from the tweets above and their classification that the model is working pretty well and classifying positive and negative sentiments properly and that tweets are clean and preprocessed.

Spark Code Explanation

First we import all the needed libraries from spark to use it to make spark session, handling features (HashingTF ,tokenizer and Stop-word Remover) ,use logistic regression for classification and the time library to calculate the runtime taken.

Secondly, we create the spark session, read the file data (tweets.csv) that include 4 columns (ItemID, Sentiment, Sentiment Score and Sentiment Text) and then select only the related columns like Sentiment Text (tweet) and the Sentiment (1 for positive sentiment and 0 for negative sentiment) and then renamed that column to label.

Then we divide the data into 70% train and 30% test, then we preprocess the data by separating Sentiment Text column into individual words using tokenizer (BAG OF WORDS) to extract features from text document then we remove the Stop-words which are words that do not contain enough significance to be used without our algorithm as I, The, is, a.

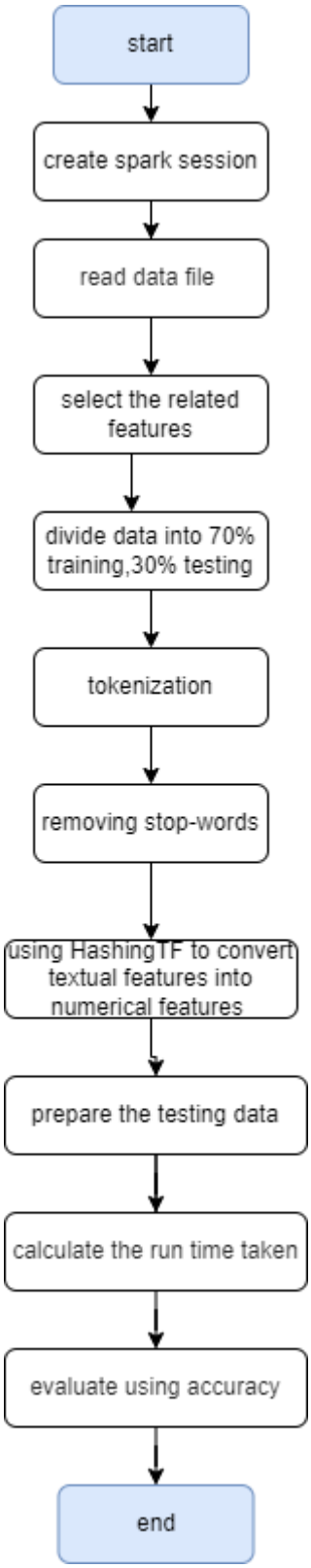
We use HashingTF converts documents into a numerical representation and the HashingTF converts documents to vectors of fixed size, The terms are mapped to indices using a Hash Function and the hash function used is Murmur Hash, so the term frequencies are computed with respect to the mapped indices.

Then we train our training data using the logistic regression with maxIter=10 and regParam=0.01 to classify the data to positive sentiments and negative sentiments.

We repeat the #Repeats the above steps for testing data step1: tokenize the data, step2: remove stop words and step 3: convert the text data into numeric data (extract features).

Finally we Predict testing data, calculate the run time taken to compare it with the time taken with Hadoop using the same dataset and calculate the accuracy model

Flow Chart of Spark code



Command to run Spark

```
nouran@nouran-virtual-machine:~/Downloads$ spark-submit sentiment_spark.py
```

Spark Code Results Explanation

We show the results of every step in our code.

First of all, we show the first 3 rows in our data file before choosing the related columns then we select the Sentiment Text(tweet) column and the sentiment column that we rename it to label and show the first 5 rows of the data.

```
+------+---------+---------------+---------------------------------+
|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

+---------------------------------+-----+
|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
```

Secondly, we divide the data into 70% train as shown (1358) and 30% test as shown (574) and we start to preprocess the data by using the tokenizer so here we tokenize Sentiment Text column into individual words as shown below in the figure.

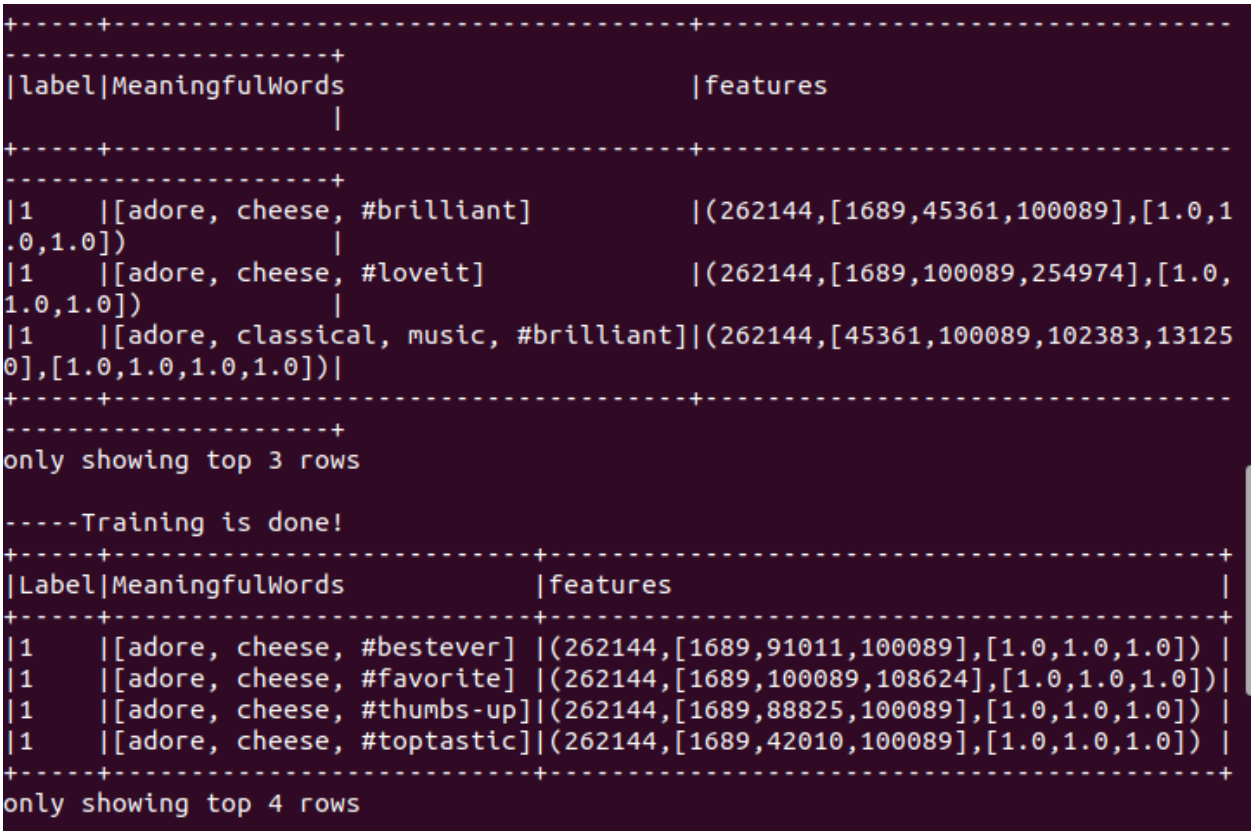
```
Training data rows: 1358 ; Testing data rows: 574
+---------------------------------+-----+-----------------------------------
---+
|SentimentText                    |label|SentimentWords
   |
+---------------------------------+-----+-----------------------------------
---+
|I adore cheese #brilliant        |1    |[i, adore, cheese, #brilliant]
   |
|I adore cheese #loveit           |1    |[i, adore, cheese, #loveit]
   |
|I adore classical music #brilliant|1    |[i, adore, classical, music, #brillia
nt]|
|I adore classical music #favorite |1    |[i, adore, classical, music, #favorit
e] |
|I adore classical music #loveit   |1    |[i, adore, classical, music, #loveit]
   |
+---------------------------------+-----+-----------------------------------
---+
only showing top 5 rows
```

Then we remove the stop word like (I) that exist in every tweet and do not contain enough significance.

```
+---------------------------------+-----+-----------------------------------
---+-------------------------------------+
|SentimentText                    |label|SentimentWords
   |MeaningfulWords                      |
+---------------------------------+-----+-----------------------------------
---+-------------------------------------+
|I adore cheese #brilliant        |1    |[i, adore, cheese, #brilliant]
   |[adore, cheese, #brilliant]          |
|I adore cheese #loveit           |1    |[i, adore, cheese, #loveit]
   |[adore, cheese, #loveit]             |
|I adore classical music #brilliant|1    |[i, adore, classical, music, #brillia
nt]|[adore, classical, music, #brilliant]|
|I adore classical music #favorite |1    |[i, adore, classical, music, #favorit
e] |[adore, classical, music, #favorite] |
|I adore classical music #loveit   |1    |[i, adore, classical, music, #loveit]
   |[adore, classical, music, #loveit]   |
+---------------------------------+-----+-----------------------------------
---+-------------------------------------+
only showing top 5 rows
```

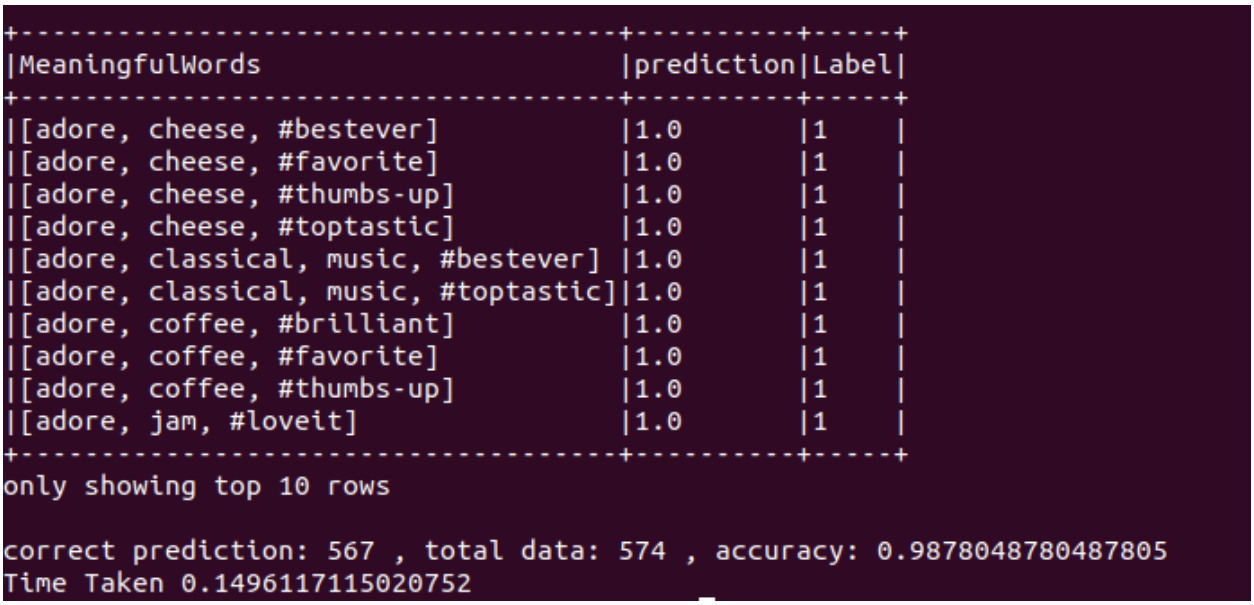
Then we use HashingTF to converts documents into a numerical representation and in the HashingTF The term frequencies are computed with respect to the mapped indices. It can be seen in the below figure that the dimension of the vector is set to default 262,144. Also, every term in the tweet is mapped to index in the second column by the hashing function and all the terms have frequency equal to 1.

Then we train the data using the logistic regression classifier and after that we repeat the preprocessing steps into the testing data and shows the first 4 rows in the testing data.



After that we predict the testing data and calculate the accuracy model by dividing all the correct prediction by the total amount of the data and it seems good because it is 98%.

We also calculate the run time taken during the prediction to compare it with the Hadoop code using the same dataset



```
+-----------------------------------+----------+-----+
|MeaningfulWords                    |prediction|Label|
+-----------------------------------+----------+-----+
|[adore, cheese, #bestever]         |1.0       |1    |
|[adore, cheese, #favorite]         |1.0       |1    |
|[adore, cheese, #thumbs-up]        |1.0       |1    |
|[adore, cheese, #toptastic]        |1.0       |1    |
|[adore, classical, music, #bestever]|1.0       |1    |
|[adore, classical, music, #toptastic]|1.0       |1    |
|[adore, coffee, #brilliant]        |1.0       |1    |
|[adore, coffee, #favorite]         |1.0       |1    |
|[adore, coffee, #thumbs-up]        |1.0       |1    |
|[adore, jam, #loveit]              |1.0       |1    |
+-----------------------------------+----------+-----+
only showing top 10 rows

correct prediction: 567 , total data: 574 , accuracy: 0.9878048780487805
Time Taken 0.1496117115020752
```