**Extraction of different features on Twitter Streaming data using Spark RDD**

**Team Members**

**Gayathri**

**Ravi Teja**

**Dheeraj**

**Saketh**

**Goals and Objectives**

**MOTIVATION:**

One of the biggest problems with Twitter is that instead of sharing their thoughts and experiences, some people start to judge others negatively, especially women, for example, through body shaming. These circumstances made us realize that these types of tweets needed to be filtered or action taken. As a result, Twitter becomes a platform where people may share information rather than propagate hate or negativity. This form of extraction enables us to respond quickly to such problems.

**OBJECTIVES:**

The goal of this task is to use Twitter's Streaming API and the Spark RDD streaming module to analyze the Real-time/Ongoing tweets available on Twitter. The expected results are to extract features from tweets. The characteristics include a variety of opinions, conditional, vulgarity, and negativity. If a tweet contains negativity or vulgarity, it will be removed. Quick action can be taken to correct it, which is extremely beneficial to large enterprises in correcting their administrations and following up on it to work on their administrations. As a result of a lot of negativity spreading on their platform, their reputation is at stake.

**FEATURES:**

The system uses spark streaming to efficiently deal with Twitter streaming data and can extract linguistic features. It is a modern Python application designed to extract characteristics such as extremity vulgarity proportion, Negatives... Spark API enables adaptable, high-throughput, and open-minded stream handling of live data streams. The framework is intended to divide the input stream into different information streams and allows for the storage of handled data in HDFS, the neighborhood record framework.

**SIGNIFICANCE:**

The significance of this project is to extract the various features of a tweet, which helps organizations improve their platform's performance as there is a lot of negativities being spread on social media platforms, and it also helps in keeping track of their clients' activities on their platform and performing accordingly for the betterment of their market performance. We are using spark RDD as the main framework for this extraction, which uses real-time tweets to perform the operations, allowing industries to respond quickly and without delay.

**Related Work (Background):**

Several studies have been conducted in the field of real-time data analytics, which use various methodologies to contribute to a specific category in daily life. Social media data, on the other hand, is typically data based on opinions and feelings, but there is still a lot of it.

Analyzing posts on social media sites like Facebook and Twitter can help you draw conclusions and make predictions about events that happen in specific parts of the world at specific times.

**Below are a few related works that are based on the real-world scenarios**

* A study proposed a solution to the problem of early event identification by using real-time event detection for online behavioral analysis of Twitter users.
* A study on stream computing in healthcare applications presented a methodology for finding patterns in diseases from three Australian cities, as well as a text analysis based on categorizing the list of words. In addition, they used a scoring system to determine the relationship between tweets and diseases.
* Another study investigated detecting customer feelings toward a brand by mining their social media language, while another looked into sentiment analysis using Twitter data.
* Many data management studies employ methodologies that make use of Apache Storm to analyze data as it comes in. It processes an event at a time, as opposed to batch processing, and provides general primitives for performing real-time computation. Methodologies that use storm for data processing can typically perform general computational tasks, but if they want to use more complicated processes, they frequently do so after storing the results in a database or passing the results to another real-time data analytics tool. Additionally, Apache Storm does not support online machine learning.
* Apache Spark is a Big Data processing framework designed to provide speed in addition to sophisticated analytics, it performs streaming computation as a series of very small, deterministic batch jobs.

**Representation of Model**

**Workflow Diagram:** Here, we are discussing about the workflow of the project. Twitter is main source of our streaming, where we must connect to the twitter API using the Apache spark and the twitter data is collected and the tweets are preprocessed, and linguistic features are extracted from the tweets, and they are classified as different categories and finally the data is visualized using the graph



**Architecture Diagram:**

Text

Description automatically generated with medium confidence

Tweepy is a python module that connects to the twitter API and socket acts as a connector between twitter API and our local machine

**Detail Design of Dataset:**

This project's dataset is Twitter streaming data. Twitter is social network which allows 140-character status updates known as tweets in general society

Twitter is a great way to find out how people feel about current events because these tweets are sent out regularly. Using the Twitter streaming API, we were able to access Twitter's global stream of data. A persistent HTTP connection must be open for this real-time access to tweets.

A developer account needs to be created to get started because the Twitter Streaming API requires authentication. We then used Tweepy to connect to Twitter using the authentication credentials (Consumer Key, Consumer Secret, Token, and Token Secret).

* Tweepy requires a user-defined listener class to open a Twitter stream. The listener will automatically determine the type of data sent by Twitter and will invoke an appropriate method to deal with the specific data type.
* It will display the text of any tweet obtained from the Twitter API. We also override the listeners on error method so that we can properly handle errors from the Twitter API.
* We start the listener by creating an instance of our stream listener class, then an instance of the Tweepy stream class, and finally we start streaming tweets by calling the filter method.
* After the connection is established, Spark Streaming, which is built on top of Spark Core, handles the reception of real-time tweets, which are then processed by the Spark Core engine.

Below is the Dataset of the twitter streaming, it contains the following fields

Graphical user interface, text, application

Description automatically generated

There are so many different fields in the twitter dataset, where they are all unique and it represents different type of values.

**Example:**

Text

Description automatically generated

**Detail Design of Features:**

Our project's process is centered on Spark. Following the establishment of a connection with the Twitter streaming API, the application filters tweets about a specific product or topic from hundreds of tweets posted every second.

Spark Streaming takes care of the streaming data and compresses it and Organize tweets into batches and send them to the spark core engine for processing. Feelings of Each tweet is examined in real time, and if it is determined that it belongs to a feature category, more actions can be taken to focus adverts, provide customer service, and collect meaningful information about negative feedback remarks, so that corrective action can be performed right away to avoid losing potential customers.

**Spark-streaming**

Spark Streaming is an extension that enables scalable, high-throughput, fault-tolerant stream processing of live data streams. Data can be ingested from many sources like Twitter etc. and can be processed using algorithms expressed with high-level functions like map, filter etc., Finally, processed data can be pushed out to HDFS, databases.

**Text, whiteboard

Description automatically generated**

Table

Description automatically generated with medium confidenceSpark Streaming receives live input data streams and divides them into batches, which are then processed by the engine to produce the result stream in batches.

Spark Streaming provides a high-level abstraction known as discretized stream, or DStream, that denotes a constant stream of information.

Python is the programming language we use for this. We used the Python package Tweepy to connect to the streaming API and because it could handle errors properly.

**Tweepy**:

It's an open-source Python library that makes it relatively easy to use Python to access the Twitter API. Tweepy is a collection of classes and methods that represent Twitter's models and API endpoints, and it handles implementation details like data encoding and decoding, HTTP requests, result pagination, OAuth Authentication, Rate Caps, and Streams transparently. Tweepy works with almost all the Twitter API's features.

You'd have to deal with low-level details like HTTP requests, data serialization, authentication, and rate constraints if you didn't use Tweepy. This may take a long time and be prone to mistakes. Tweepy, on the other hand, enables you to concentrate on the features you want to create.

**Analysis of Data**

* The aim of our project is to create a framework for analyzing Twitter data in real-time. The main challenge here is that there is a need to develop a method for analyzing thousands of tweets per second in a short period of time.
* Hence, we need a framework that is not dependent on the volume of imported data, which is critical because the volume of tweets is rapidly increasing. We encountered challenges in the data processing and data storage.
* We use streaming data processing, a scalable and distributed platform capable of combining large amounts of historical and streaming data.
* Real-time data analysis necessitates data ingestion and processing of the data stream prior to data storage.
* We considered the following parameters to handle streaming data reliably and consistently:

**Data Storage:**

Due to storage constraints, a system in a streaming environment cannot store all the data. However, managing data processing and incoming data can result in some incoming data being lost.

To ensure system reliability, incoming data should be stored for a few time ticks. Once processed, the data can be trunked to make room for incoming data.

**Data pre-processing:**

The challenge with processing data in real time is managing your data processing module while not losing incoming live data streams.

The time latency between incoming data at time t and processing incoming data at time t-1 should be as short as possible to avoid losing incoming data.

When it tends to come to Spark, we must set a time (time set) so that it will read over the streams like a sliding read header and insert the data into a spark RDD for every set time (seconds or milliseconds). We can work on each RDD that is being processed concurrently and aggregate the results.

To overcome the above problems, we are using spark streaming and slider window protocol to retrieve the tweets and extract the features from it.

**Implementation:**

The main component of the feature extraction from twitter is to extract the real-world tweets from the twitter. In Spark Streaming, the data can be collected or retrieved from many sources and can be processed using complex algorithms like map, reduce, window, and join. In this project, we use sliding window technique because, we will be receiving thousands of tweets in seconds, and it is hard to manage all the tweets at a time. Hence, we are using the sliding window protocol, where the twitter stream data is stored as the streams as per the time limit. So that we can retrieve the tweets and start analyzing the data.

There are certain steps that are needed to follow for execution of the code and to retrieve the results

1. **Install Scala:** Download Scala from the link: [http://downloads.lightbend.com/scala/2.11.8/scala-](http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.msi) [2.11.8.msi](http://downloads.lightbend.com/scala/2.11.8/scala-2.11.8.msi)
   1. Set environmental variables:
      1. User variable:
         1. Variable: SCALA\_HOME.
         2. Value: C:\Program Files (x86) \scala
      2. System variable:
         1. Variable: PATH
         2. Value: C:\Program Files (x86)\scala\bin

Text

Description automatically generated

1. **Install Java 8:** Download Java 8 from the link: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
   1. Set environmental variables:
      1. User variable:
         1. Variable: JAVA\_HOME
         2. Value: C:\Program Files\Java\jdk1.8.0\_202
      2. System variable:
         1. Variable: PATH
         2. Value: C:\Program Files\Java\jdk1.8.0\_202\bin
   2. Check on cmd, see below:

Text

Description automatically generated

1. Graphical user interface, text, application, email

   Description automatically generated**Install Spark 1.3.1.** Download it from the following link: <http://spark.apache.org/downloads.html>and extract it into C drive, such as C:\Spark.
   1. Set environmental variables:
      1. User variable:
         1. Variable: SPARK\_HOME
         2. Value: C:\spark-3.1.3-bin-hadoop3.2
      2. System variable:
         1. Variable: PATH
         2. Value: C:\spark-3.1.3-bin-hadoop3.2\bin
2. **Install Eclipse** Mars. Download it from the link: <https://eclipse.org/downloads/>and extract it into C drive.
   1. Set environmental variables:
      1. User variable:
         1. Variable: ECLIPSE\_HOME
         2. Value: C:\Eclipse\eclipse
      2. System variable:
         1. Variable: PATH
         2. Value: C:\Eclipse\eclipse\bin
3. **Download Windows Utilities**: Download it from the link: <https://github.com/steveloughran/winutils/tree/master/hadoop-2.6.0/bin> And paste it in C:\ spark-3.1.3-bin-hadoop3.2\bin
4. **Install Maven 3.3**. Download Apache-Maven-3.3.9 from the link: <http://apache.mivzakim.net/maven/maven-3/3.3.9/binaries/apache-maven-3.3.9-bin.zip> And extract it into C drive, such as C:\apache-maven-3.3.9
   1. Set Environmental variables:
      1. User variable
         1. Variable: MAVEN\_HOME
         2. Value: C:\apache-maven-3.3.9
      2. System variable
         1. Variable: Path
         2. Value: C:\apache-maven-3.3.9\bin
   2. Check on cmd, see below

Text

Description automatically generated

After the Installing of the above software’s, we must install the specific python libraries, and must import the python codes and run the shell scripts.

**Algorithms/Pseudocode**

Please refer to the below, pseudocode where we have used the twitter consumer and authorization key, which we created using the developer twitter account.

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

After creating the code, please import the tweepy library and Jason library. A listener is created, which will handle the tweeter streams. This is the basic and import algorithm to import the twitter data through the twitter API.

Text, application

Description automatically generated

We can see in the above code that it will collect JSON from Twitter and preprocess it based on the given conditions. We'll need to import some libraries from Twitter for this.

It will eliminate all stop words and tokenize the tweets based on the streaming.

Graphical user interface, text, application, email

Description automatically generated

In the above output, we could see that we are giving the certain key words, which are used to find the tweets using that. Once that keyword is there in the tweet it will increase the count of that field (i.e., “opinion count”)

Graphical user interface, text, application, email

Description automatically generated

In the above code, we have predefined all the keywords related to vulgarity, so that if any of these keywords appear in a tweet, it is considered a negative tweet and causes the graph to rise. Similar to the vulgarity, we have given the key words related to the Positivity, Neutral and negative

Graphical user interface, text, application, email

Description automatically generated

**Script.sh**

We are using the shell script to, visualize the result by executing the below shell script. By executing the script, we are storing the data in the home directory. Each twitter data is stored as a separate file in the directory, and count of the that particular field.

Text

Description automatically generated

Graphical user interface, text

Description automatically generated**Visualization of the result**

In the below code, we are importing the “Matplot” libraries to visualize the result and to represent the output in the form of graph. We are defining the axis and coordinates to get the results.

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

In the above graph, we have labeled each axis and given it a title to make it easier to represent the lines in the graph and to make it more understandable.

This is the implementation of the project and output is shown below.

**Results:**

We have created new twitter developer account because Twitter Streaming API requires authentication. We then passed the authentication credentials like Consumer Key, Consumer Secret key, Token, and Token Secret into Tweepy to connect with twitter.

**Creation of the twitter account and generating authentication keys**

We have created a new twitter account and generated the API keys and authentication keys which are unique and different from others, these keys are used to connect to Twitter API to connect and retrieve the real time tweets.

Graphical user interface, text, application

Description automatically generated

**Access key creation:**

We have created a new access key for our account to retrieve the information from twitter API for real world twitter streaming. Graphical user interface, text, application

Description automatically generated

Python modules required to run listener were installed. We have used Pip to install Python modules. Tweepy was installed for streaming. Pyspark was installed to handle real-time data processing. Algorithms for working with data in object stores have been installed. Textblob was installed for NLP text processing. We have also installed Spacy for NLP. We generated a consumer key, a consumer secret, an access token, and an access secret to retrieve tweets from Twitter, and we wrote a python program to extract and filter bad tweets.

After creating the Twitter account, we need to execute the three source codes parallelly, which we have represented in the implementation part.

Chart, line chart

Description automatically generated

*Fig 1*

There are various categories of tweets in the above output, which are represented by the lines in the graph. The various color lines depict various sentiment representations. When the curve rises, it indicates that there are a greater number of tweets in that category. For example, when the negative graph rises, it indicates that there is more negativity in the tweets at that

time

Chart, line chart

Description automatically generated

*Fig 2*

In the above graph, we have streamed the data another time, there are various categories of tweets in the above output, which are represented by the lines in the graph. The various color lines depict various sentiment representations. When the curve goes below, it indicates that there are a greater number of tweets in that category. For example, when the positive graph rises, it indicates that there is more positivity in the tweets at that

time

A picture containing chart

Description automatically generated

***Fig 3***

Text

Description automatically generated

***Fig 4***

**Implementation Status Report**

**Work completed:**

* Description:
* In this Second increment, after creating the developer twitter account and generated the access and authentication keys.
* Have used Tweepy to connect to the twitter API using the access keys and retrieve the twitter data and to perform the feature extraction.
* Installed the required libraries/modules in the python
* Installed "Pyspark” for real time processing of the data by connecting with the twitter streaming.
* We have used the Streamslidingwindow protocol to stream the data and provide the required conditions to filter the tweets.
* Using the sliding window protocol, we can retrieve twitter streaming details from the Twitter API and extract features based on the keywords we specified in the sliding window protocol program. Aside from the sliding window protocol, we project the results as a graph and visualize the data.

• Responsibility (Task, Person):

Each person in the team has taken equal responsibility in developing the project for phase 2 implementation by developing all the modules and setting up the libraries and installing the different type of software’s

Please find the below contributions and responsibilities from each person

• Contributions (members/percentage):

|  |  |  |
| --- | --- | --- |
| **Person** | **Contribution** | **Percentage** |
| Gayathri | Twitter account creation and generating access/authentication keys, Sliding window protocol,  Documentation | 25 |
| Saketh | Coding for Tweepy to connect to twitter API, Visualizing the Graph Documentation | 25 |
| Ravi | Twitter dataset extraction and connecting twitter API Documentation | 25 |
| Dheeraj | Software/libraries installation in python and coding the tweepy protocol in connecting with twitter API Documentation | 25 |

• Issues/Concerns:

* We faced few challenges while installing the required software’s.
* Faced few challenges while extracting the dataset from twitter API
* At the start of twitter streaming, there are some connectivity issues with the twitter API
* We encountered a few difficulties in retrieving the exact graph for visualizing the output.

**References:**

1. N. Mohamed, J. Al-jaroodi, Real-Time Big Data Analytics: Applications and Challenges. International Conference on High Performance Computing & Simulation (HPCS), 2014.

2. Yadranjiaghdam, B., Yasrobi, S., & Tabrizi, N. (2017, June). Developing a Real-time Data Analytics Framework For Twitter Streaming Data. In Big Data (BigData Congress), 2017 IEEE International Congress on (pp. 329-336). IEEE.

3. B. Yadranjiaghdam, N. Pool, N. Tabrizi, “A Survey on Real-time Big Data Analytics: Applications and Tools,” in progress of International Conference on Computational Science and Computational Intelligence, 2016.

4. Bifet, A. (2013). Mining big data in real time. Informatica, 37(1), 15.

5. Nguyen, D. T., & Jung, J. E. (2017). Real-time event detection for online behavioral analysis of big social data. Future Generation Computer Systems, 66, 137-145.

6. J. Zaldumbide, R. O. Sinnott, “Identification and Validation of RealTime Health Events through Social Media,” 2015 IEEE International Conference on Data Science and Data Intensive Systems, Pages 9 – 16, doi 10.1109/DSDIS.2015.27.

7. M.M. Mostafa, "More than words: Social networks’ text mining for consumer brand sentiments", Expert Systems with Applications, vol. 40, no. 10, pp. 4241-4251, 2013.

8. A. Pak and P. Patrick, "Twitter as a Corpus for Sentiment Analysis and Opinion Mining", LREC, vol. 10, pp. 1320- 1326, 2010.

9. A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, JM. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, “Storm@ twitter”, InProceedings of the 2014 ACM SIGMOD international conference on Management of data 2014 Jun 18 (pp. 147-156). ACM.

10. V. Ta, C. Liu, G.W. Nkabinde, “Big Data Stream Computing in Healthcare Real-Time Analytics”, 2016, IEEE International Conference on Cloud Computing and Big Data Analysis, Pages: 37 - 42, doi: 10.1109/ICCCBDA.2016.7529531.

11. [10] M. Wachowicz, M.D. Artega, S. Cha, and Y. Bourgeois, “Developing a streaming data processing workflow for querying space–time activities from geotagged tweets” Computers, Environment and Urban Systems Journal. 2015.

12. [14] A. Sotsenko, M. Jansen, M. Milrad, J. Rana, “Using a Rich Context Model for RealTime Big Data Analytics in Twitter”, 4th International Conference on Future Internet of Things and Cloud Workshops, 2014, Pages 228 -233, DOI 10.1109/W-FiCloud.2016.55.

13. M. Z. Mosharaf Chowdhury and T. Das, “Resilient distributedmdatasets: a fault-tolerant abstraction for in-memory cluster computing,” in NSDI’12 Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. San Jose, CA:USENIX Association Berkeley, Apr. 2012