

CHAPTER 6

IMPLEMENTATION

6.1 Problem Formulation:

With all the real time data collected over period of time, the system will analyze and draw meaningful inferences from the collection of tweets. Proposed system will analyze tweets data from many perspectives to make meaningful inferences. Trend analysis, sentiment analysis volume analysis are major parts of proposed system. In trend analysis, system will try to find trending discussions, parties, personalities throughout the period of time. From literature, K-means is more suitable algorithm for clustering of tweet data and to find trends. Volume analysis of tweets will give idea of popularity of particular topic or person over a period of time. Volume analysis with respect to geo-location and date will help to make certain conclusions. Sentiment analysis of tweets will help draw conclusion for political orientation of overall users respective to political parties, topics.

Input:

- A tweeter tweets related to election
- A hash-tag List

Output

- Trend Analysis
- Sentiment Analysis

6.2 Objective:

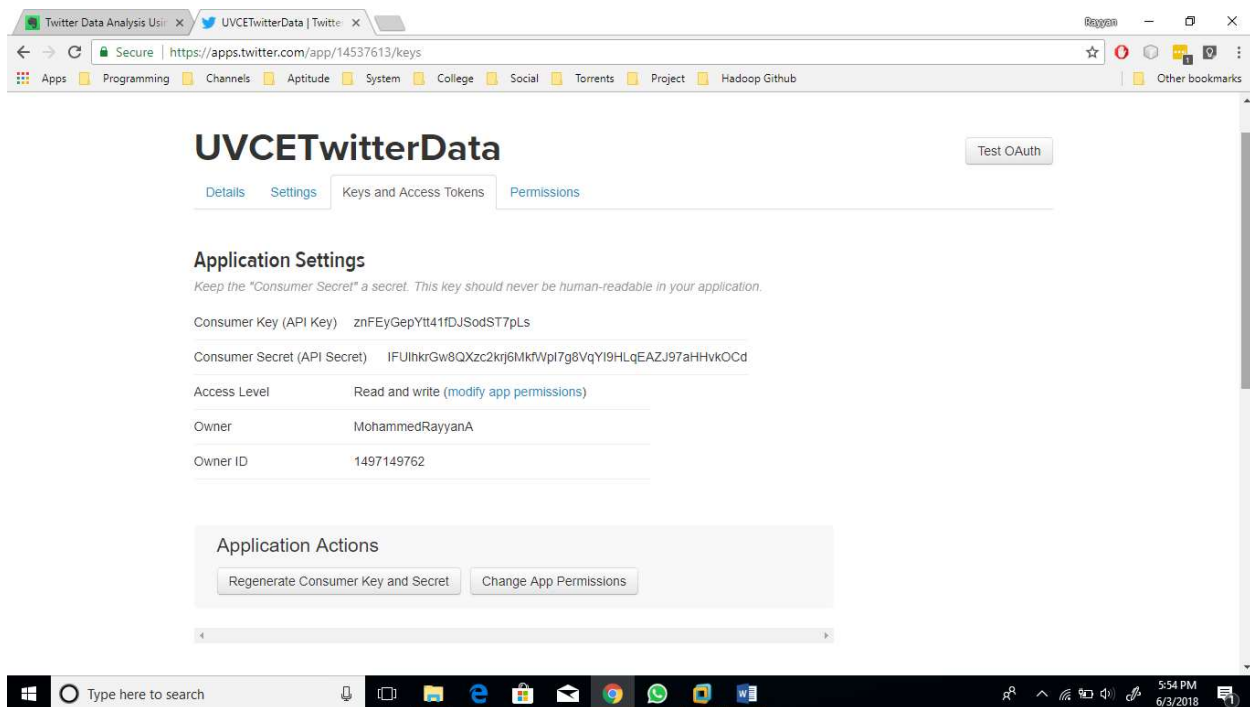
In this Hadoop project you are going to perform following activities:

- 1 Connect to a live social media (twitter) data stream, extract and store this data on Hadoop
- 2) Process the data in Hadoop; restructure, filter and provide useful insights from it
- 3) Create tables in Hadoop and provide an interface to end users for simple querying

- 4) Do sentiment analysis by comparing the sentiments of people on a particular item or subject
- 6) You need to work on Hadoop tools like HDFS, MapReduce, Hive and Flume

6.3 Creation of Twitter Account:

Twitter Developer Account can be created at Twitter Developers apps Page. In this page we need to provide valid twitter account page in the website field from which we need to get streaming data. If we provide valid details on this page we will get our app created as shown in below screen shots. For security reasons, I have blurred consumer access key, secret values in the below screens.

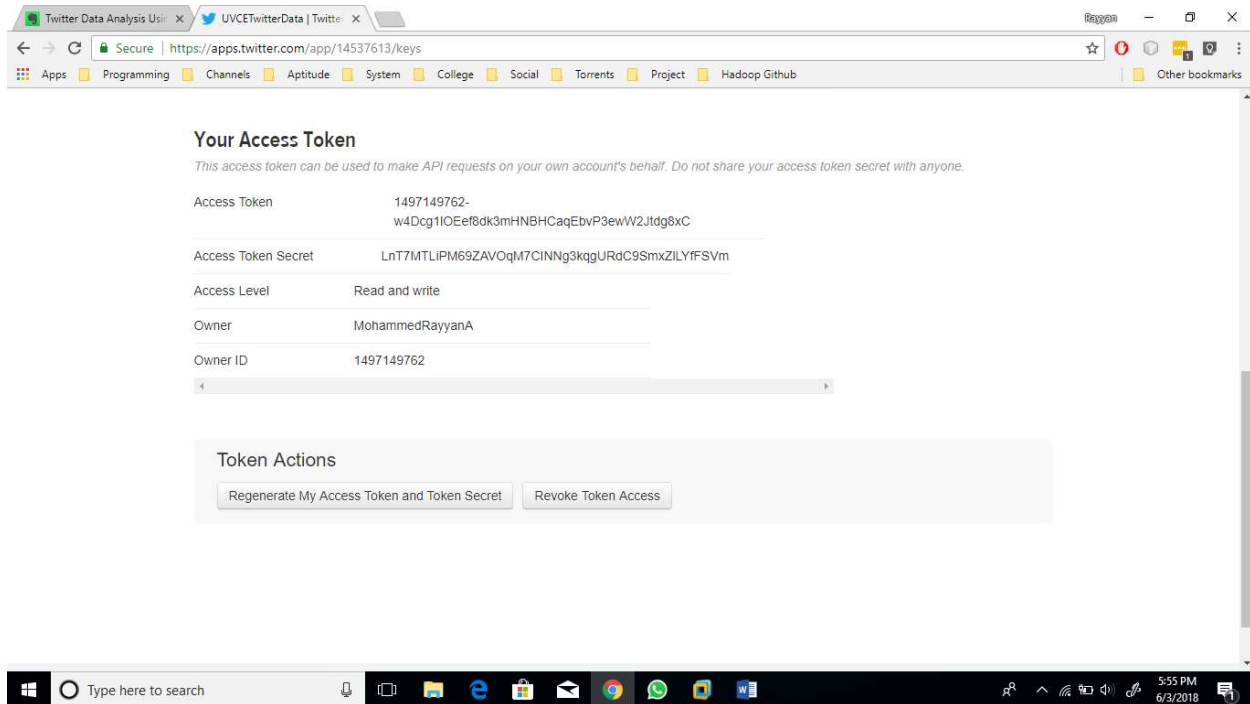


We need below four values for authenticated by Twitter.

- **Consumer Key (API Key):** znFEyGepYtt41fDJSodST7pLs
- **Consumer Secret (API Secret):**
IFUlhkrGw8QXzc2krj6MkfWpI7g8VqYI9HLqEAZJ97aHHvkOCd
- **Access Token :** 1497149762-w4Dcg1lOEef8dk3mHNBHCaqEbvP3ewW2Jtdg8xC

- **Access Token Secret :**

LnT7MTLiPM69ZAVOqM7CINNg3kqgURdC9SmxZILYfFSVm



6.4 Installing Hadoop

Disable IPv6

```
>>sudo apt-get install vim
```

```
>>sudo gedit /etc/sysctl.conf
```

```
# disable ipv6
```

```
net.ipv6.conf.all.disable_ipv6 = 1
```

```
net.ipv6.conf.default.disable_ipv6 = 1
```

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

```
>>cat /proc/sys/net/ipv6/conf/all/disable_ipv6 ... (should return zero)
```

Adding a dedicated Hadoop User

```
>>sudo addgroup hadoop
```

```
>>sudo adduser --ingroup hadoop hduser
```

Install SSH

```
>>sudo apt-get install ssh
```

Give hduser Sudo Permission

```
>>sudo adduser hduser sudo
```

Setup SSH Certificates

```
>>su hduser
```

```
>>ssh-keygen -t rsa -P ""
```

```
>>cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

```
>>ssh localhost
```

Download Hadoop

```
>>su hduser
```

```
>>sudo wget http://www-us.apache.org/dist/hadoop/common/hadoop-2.9.0/hadoop-2.9.0.tar.gz
```

```
>>tar xvzf hadoop-2.9.0.tar.gz
```

```
>>cd hadoop-2.9.0
```

```
>>sudo mkdir /usr/local/hadoop
```

```
>>sudo mv * /usr/local/hadoop
```

Set up the Configuration files

```
>> sudo gedit ~/.bashrc
```

```
#HADOOP VARIABLES START

export JAVA_HOME=/usr/lib/java/jdk-9.0.4

export HADOOP_INSTALL=/usr/local/hadoop

export PATH=$PATH:$HADOOP_INSTALL/bin

export PATH=$PATH:$HADOOP_INSTALL/sbin

export HADOOP_MAPRED_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_HOME=$HADOOP_INSTALL

export HADOOP_HDFS_HOME=$HADOOP_INSTALL

export YARN_HOME=$HADOOP_INSTALL

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"

#HADOOP VARIABLES END
```

```
>> Sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/java/jdk-9.0.4
```

```
>>sudo mkdir -p /app/hadoop/tmp
```

```
>>sudo chown hduser:hadoop /app/hadoop/tmp
```

```
>>sudo gedit /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
<property>
```

```
    <name>hadoop.tmp.dir</name>
```

```
    <value>/app/hadoop/tmp</value>
```

```
    <description>A base for other temporary directories.</description>
```

```
</property>
```

```
<property>
```

```
    <name>fs.default.name</name>
```

```
    <value>hdfs://localhost:54310</value>
```

```
    <description>The name of the default file system. A URI whose  
  
scheme and authority determine the FileSystem implementation. The  
  
uri's scheme determines the config property (fs.SCHEME.impl) naming  
  
the FileSystem implementation class. The uri's authority is used to  
  
determine the host, port, etc. for a filesystem.</description>
```

```
</property>
```

```
>>cp/usr/local/hadoop/etc/hadoop/mapred-site.xml.template  
/usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
>>sudo gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

```
<property>
```

```
<name>mapred.job.tracker</name>
```

```
<value>localhost:54311</value>
```

```
<description>The host and port that the MapReduce job tracker runs
```

at. If "local", then jobs are run in-process as a single map

and reduce task.

```
</description>
```

```
</property>
```

```
>>sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
```

```
>>sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
```

```
>>sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

```
>>sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

```
<property>
```

```
<name>dfs.replication</name>
```

```
<value>1</value>
```

```
<description>Default block replication.
```

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

```
</description>
```

```
</property>
```

```
<property>
```

```
    <name>dfs.namenode.name.dir</name>
```

```
    <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
```

```
</property>
```

```
<property>
```

```
    <name>dfs.datanode.data.dir</name>
```

```
    <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
```

```
</property>
```

```
>>sudo chmod 777 -R /usr/local/hadoop/
```

Format Hadoop filesystem

```
>> hadoop namenode -format
```

Starting Hadoop

```
>>su hduser
```

```
>>hadoop fs -mkdir /user/hduser/flume
```

```
>>hadoop fs -mkdir /user/hduser/hive
```

```
>>hadoop fs -chown hduser:hadoop /user/hduser
```

```
>>sudo chown -R hduser:hadoop /usr/local/hadoop/
```

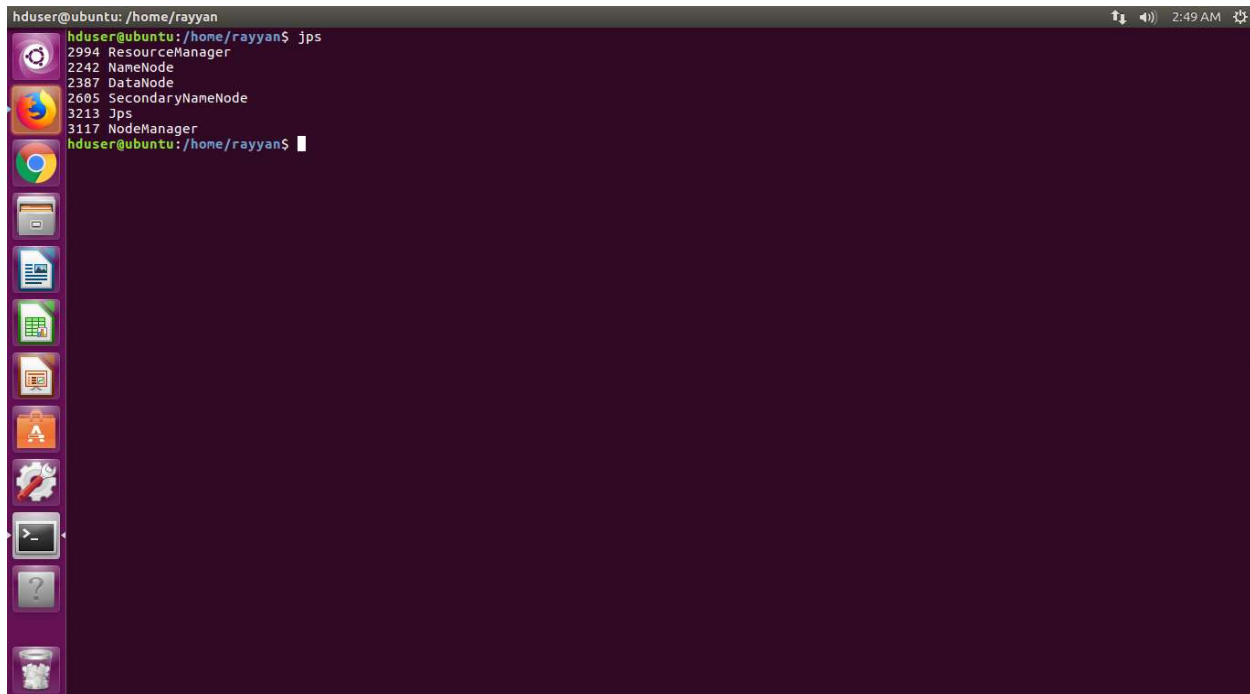
```
>>cd /usr/local/hadoop/sbin
```

```
>>start-dfs.sh
```



```
>>start-yarn.sh
```

Testing if it is working

A terminal window titled 'hduser@ubuntu: /home/rayyan' showing the output of the 'jps' command. The output lists several Hadoop processes: '2994 ResourceManager', '2242 NameNode', '2387 DataNode', '2605 SecondaryNameNode', '3213 Jps', and '3117 NodeManager'. The prompt 'hduser@ubuntu: /home/rayyan\$' is visible at the bottom of the terminal output.

```
hduser@ubuntu: /home/rayyan$ jps
2994 ResourceManager
2242 NameNode
2387 DataNode
2605 SecondaryNameNode
3213 Jps
3117 NodeManager
hduser@ubuntu: /home/rayyan$
```

Stopping Hadoop

```
>>stop-dfs.sh
```

```
>>stop-yarn.sh
```

6.5 Flume Installation

Download flume:

```
>>sudo wget http://www-eu.apache.org/dist/flume/1.8.0/apache-flume-1.8.0-bin.tar.gz
```

Unzip flume tar file

```
>>tar -xvzf apache-flume-1.8.0-bin.tar.gz
```

```
>>cd /usr/local
```

```
>>sudo mkdir flume
```

```
>>sudo mv * /usr/local/flume
```

```
>>sudo gedit ~/.bashrc
```

```
#flume variables
```

```
export FLUME_HOME=/usr/local/flume
```

```
PATH=$PATH:$FLUME_HOME/bin
```

```
export CLASSPATH=$CLASSPATH:$FLUME_HOME/lib/*
```

Download jar file

```
flume-source-1.0-SNAPSHOT.jar
```

```
>>sudo mv flume-sources-1.0-SNAPSHOT.jar /usr/local/flume/lib
```

```
In flume/conf
```

```
>>sudo cp flume-env.sh.template flume-env.sh
```

Modify flume-env.sh

```
>>sudo vim flume-env.sh
```

```
export JAVA_HOME=/usr/lib/java/jdk1.8.0_161
```

```
FLUME_CLASSPATH="/usr/local/flume/lib/flume-sources-1.0-SNAPSHOT.jar"
```

Create flume-twitter.conf file in conf folder and paste given lines

```
>>sudo gedit flume-twitter.conf
```

Configuration Code

```
TwitterAgent.sources = Twitter
```

```
TwitterAgent.channels = MemChannel

TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource

TwitterAgent.sources.Twitter.channels = MemChannel

TwitterAgent.sources.Twitter.consumerKey = znFEyGepYtt4lfDJSodST7pLs

TwitterAgent.sources.Twitter.consumerSecret =
IFUlhkrGw8QXzc2krj6MkfWpI7g8VqYI9HLqEAZJ97aHHvkOCd

TwitterAgent.sources.Twitter.accessToken = 1497149762-
w4Dcg1lOEef8dk3mHNBHCaqEbvP3ewW2Jtdg8xC

TwitterAgent.sources.Twitter.accessTokenSecret =
LnT7MTLiPM69ZAVOqM7CINN3kqgURdC9SmxZILYfFSVm

TwitterAgent.sources.Twitter.keywords =narendramodi,aap,bjp,Arvind
kejriwal,shivsena,congress


TwitterAgent.sinks.HDFS.type = hdfs

TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/user/Hadoop/twitter_data/

TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream

TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text

TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000

TwitterAgent.sinks.HDFS.hdfs.rollSize = 0

TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000

# Describing/Configuring the channel TwitterAgent.channels.MemChannel.type = memory
```

```
TwitterAgent.channels.MemChannel.capacity = 10000
```

```
TwitterAgent.channels.MemChannel.transactionCapacity = 100
```

Binding the source and sink to the channel

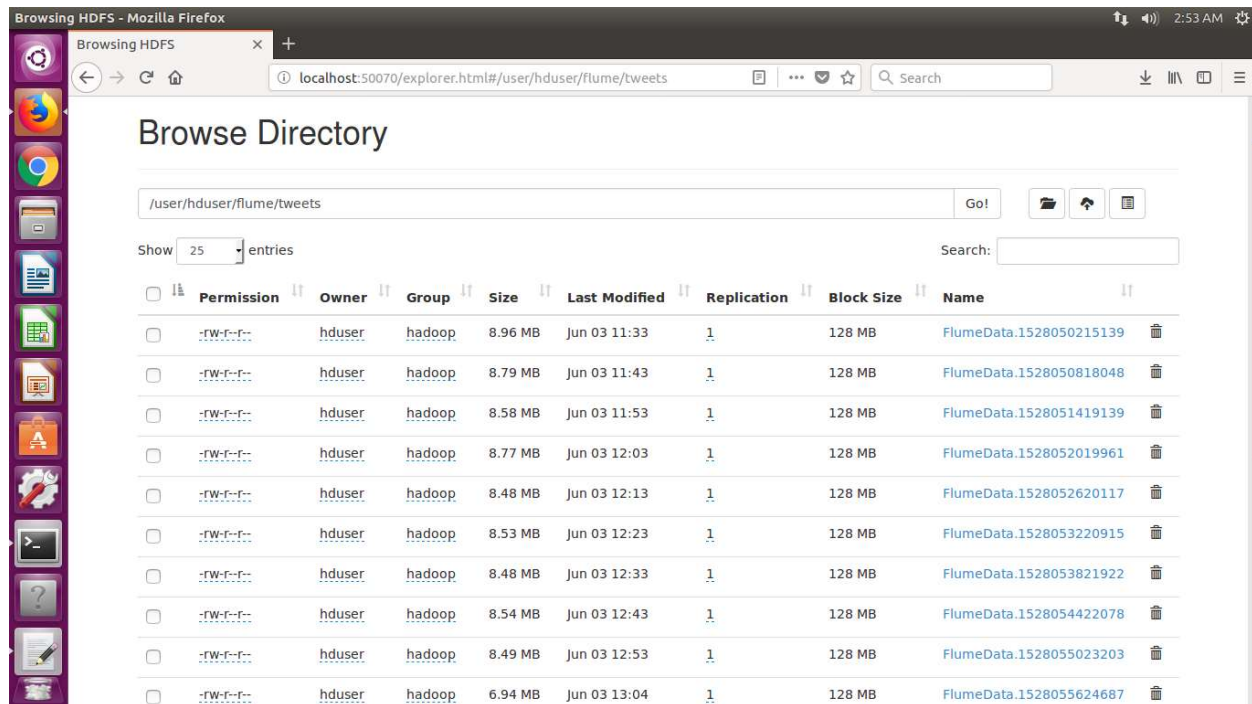
```
TwitterAgent.sources.Twitter.channels = MemChannel
```

```
TwitterAgent.sinks.HDFS.channel = MemChannel
```

To Load Twitter Data into HDFS

```
>>bin/flume-ng agent -n TwitterAgent --conf ./conf/ -f conf/flume-twitter.conf -
Dflume.root.logger=DEBUG,console
```

The data in web HDFS will be stored as shown below



Browsing HDFS - Mozilla Firefox

localhost:50070/explorer.html#/user/hduser/flume/tweets

Browse Directory

/user/hduser/flume/tweets

Show 25 entries

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hduser	hadoop	8.96 MB	Jun 03 11:33	1	128 MB	FlumeData.1528050215139
-rw-r--r--	hduser	hadoop	8.79 MB	Jun 03 11:43	1	128 MB	FlumeData.1528050818048
-rw-r--r--	hduser	hadoop	8.58 MB	Jun 03 11:53	1	128 MB	FlumeData.1528051419139
-rw-r--r--	hduser	hadoop	8.77 MB	Jun 03 12:03	1	128 MB	FlumeData.1528052019961
-rw-r--r--	hduser	hadoop	8.48 MB	Jun 03 12:13	1	128 MB	FlumeData.1528052620117
-rw-r--r--	hduser	hadoop	8.53 MB	Jun 03 12:23	1	128 MB	FlumeData.1528053220915
-rw-r--r--	hduser	hadoop	8.48 MB	Jun 03 12:33	1	128 MB	FlumeData.1528053821922
-rw-r--r--	hduser	hadoop	8.54 MB	Jun 03 12:43	1	128 MB	FlumeData.1528054422078
-rw-r--r--	hduser	hadoop	8.49 MB	Jun 03 12:53	1	128 MB	FlumeData.1528055023203
-rw-r--r--	hduser	hadoop	6.94 MB	Jun 03 13:04	1	128 MB	FlumeData.1528055624687

The data downloaded in HDFS is in JSON Format as show below



6.6 Hive Installation

Download and Extract

The downloaded Hive tar file needs to be extracted using the tar command with `-xvf` option as shown below -

```
>>tar -xvf apache-hive-2.1.0-bin.tar.gz
```

Setting Home Dir

```
>>sudo gedit ~/.bashrc
```

```
# Set HIVE_HOME
```

```
export HIVE_HOME="$HOME/hive/ apache-hive-2.1.0-bin"
```

```
PATH=$PATH:$HIVE_HOME/bin
```

```
Export $PATH
```

```
>>source .bashrc
```

Set HADOOP_HOME in hive-config.sh

```
>>sudo cd /usr/local/hive/conf
```

```
>>sudo gedit hive-config.sh
```

```
export HADOOP_HOME=/opt/hadoop
```

Create a directory for the hive warehouse into hdfs.

```
>>hadoop dfs -mkdir -p /user/hive/warehouse
```

Modify the permissions

```
>>hadoop dfs -chmod 765 /user/hive/warehouse
```

```
>>cd $HIVE_HOME/bin/
```

```
>>schematool -initschema -dbtype derby
```

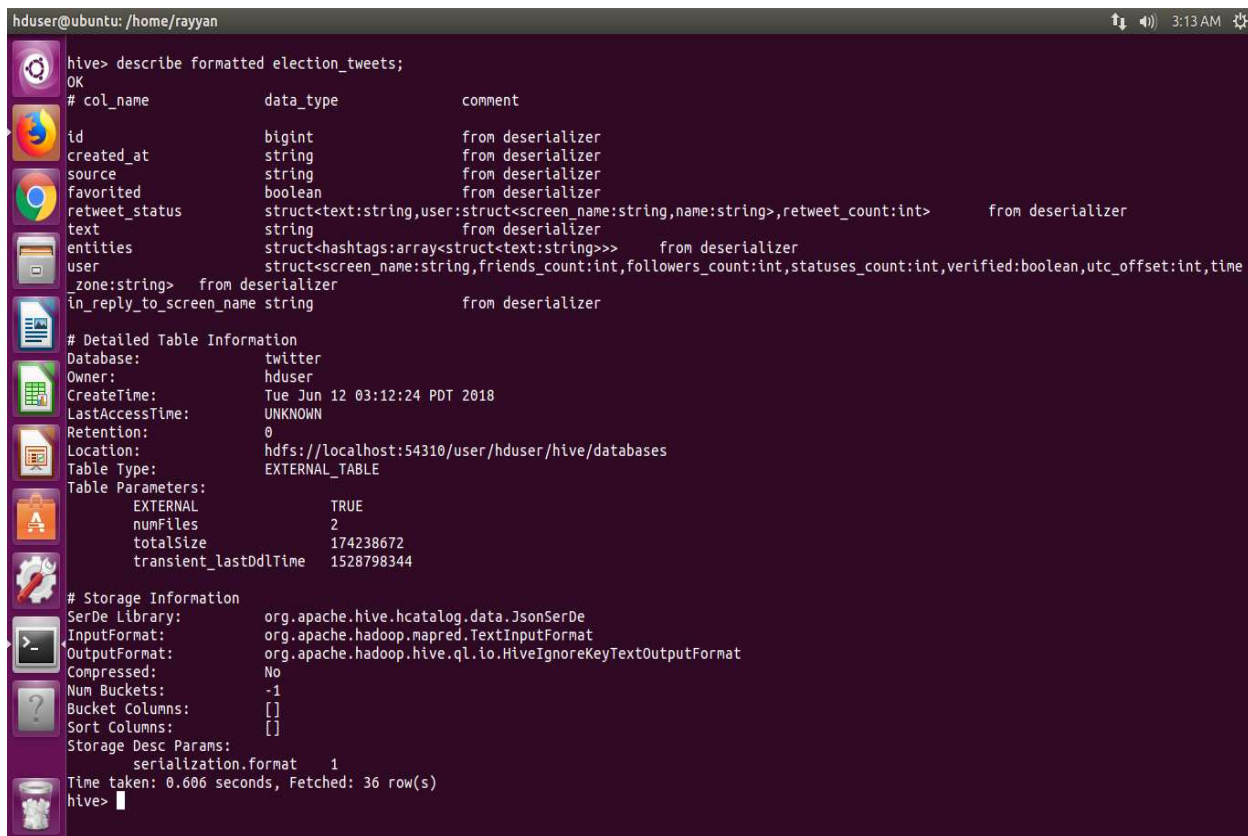
The first thing we need to do, is build the JSON SerDe from source. This will create a JAR file that will contain all the libraries necessary to convert the JSON structured tweets into a Hive table.

Now we need to create tweet table. Run hive, and execute the following commands:

```
hduser@ubuntu: /home/rayyan
> LOCATION '/user/hduser/hive/databases/';
FAILED: ParseException line 5:44 missing : at 'STRUCT' near '<EOF>'
hive> !clear;

hive> ADD JAR /usr/local/hive/lib/hive-hcatalog-core-2.3.3.jar;
Added [/usr/local/hive/lib/hive-hcatalog-core-2.3.3.jar] to class path
Added resources: [/usr/local/hive/lib/hive-hcatalog-core-2.3.3.jar]
hive> CREATE EXTERNAL TABLE election_tweets(
>
>     id BIGINT,created_at STRING,source STRING,favorited BOOLEAN,
>     retweet_status STRUCT<text:STRING, 'user':STRUCT<screen_name:STRING,name:STRING,retweet_count:INT>,
>     text STRING,
>     entities STRUCT<
>         hashtags:ARRAY<STRUCT<text:STRING>>>,
>         'user' STRUCT<
>             screen_name:STRING,
>             friends_count:INT,
>             followers_count:INT,
>             statuses_count:INT,
>             verified:BOOLEAN,
>             utc_offset:INT,
>             time_zone:STRING>,
>             in_reply_to_screen_name STRING
>         )
> )
> ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
> LOCATION '/user//hduser/hive/databases/';
OK
Time taken: 1.219 seconds
hive>
```


Now the Format of table created is shown below:



```

hduser@ubuntu: /home/rayyan
hive> describe formatted election_tweets;
OK
# col_name          data_type          comment
id                  bigint             from deserializer
created_at          string             from deserializer
source              string             from deserializer
favorited           boolean            from deserializer
retweet_status      struct<text:string,user:struct<screen_name:string,name:string>,retweet_count:int>  from deserializer
text                string             from deserializer
entities            struct<hashtags:array<struct<text:string>>>  from deserializer
user                struct<screen_name:string,friends_count:int,followers_count:int,statuses_count:int,verified:boolean,utc_offset:int,time_zone:string>  from deserializer
in_reply_to_screen_name string             from deserializer

# Detailed Table Information
Database:           twitter
Owner:              hduser
CreateTime:         Tue Jun 12 03:12:24 PDT 2018
LastAccessTime:     UNKNOWN
Retention:          0
Location:           hdfs://localhost:54310/user/hduser/hive/databases
Table Type:         EXTERNAL_TABLE
Table Parameters:
    EXTERNAL              TRUE
    numFiles               2
    totalSize              174238672
    transient_lastDdlTime  1528798344

# Storage Information
SerDe Library:      org.apache.hive.hcatalog.data.JsonSerDe
InputFormat:        org.apache.hadoop.mapred.TextInputFormat
OutputFormat:       org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
Compressed:         No
Num Buckets:        -1
Bucket Columns:     []
Sort Columns:       []
Storage Desc Params:
    serialization.format  1
Time taken: 0.606 seconds, Fetched: 36 row(s)
hive>
  
```

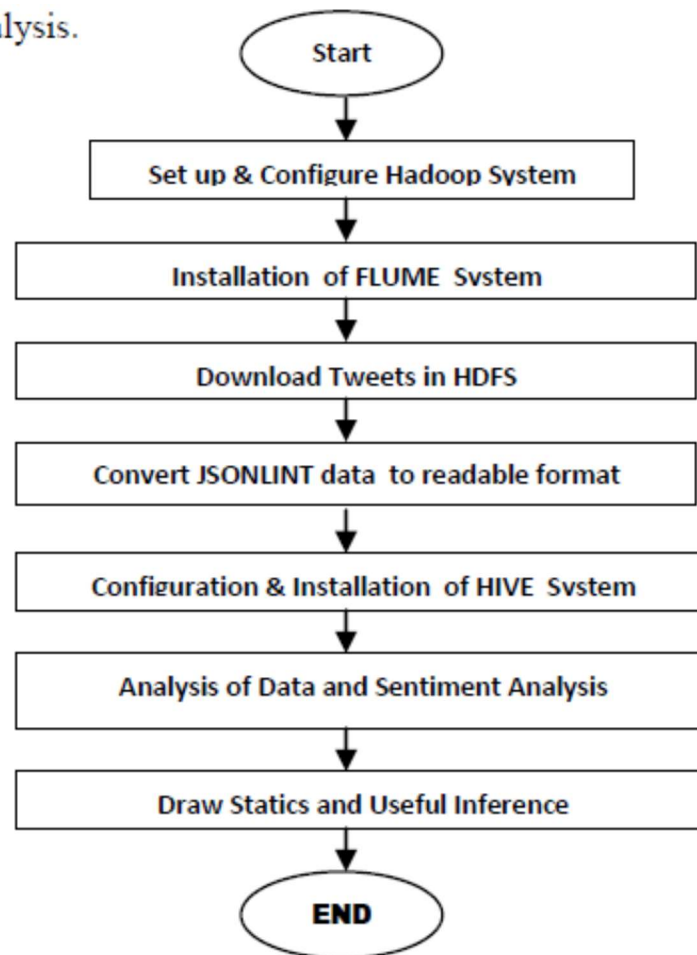
The table structure is now complete. The next step is to import the data into the tweets table from HDFS.

Load Flume Data into Hive Table

load data inpath '/user/flume/tweets/FlumeData.*' into TABLE election_tweets;

6.7 Installation & Workflow

analysis.



6.9 Twitter Analysis Steps

Analysis of data consist following steps:

1. **Tokenization:**

All the words in a tweet are broken down into tokens. This is the tokenization process. For example, '@Jack That is an awesome car!' is broken down into individual tokens such as

'@Jack', 'That', '_is', 'an', 'awesome', 'car'. Emoticons, abbreviations, hashtags and URLs are recognized as individual tokens. Each word in a tweet is separated by a space. Therefore, on encountering a space, a token is identified.

2. Normalization:

The normalization process verifies each token and performs some computing based on what kind of token it is.

- If the token is an emoticon, its corresponding polarity is taken into account by searching the emoticon dictionary.
- If the token is an acronym, it is checked in the acronym dictionary and the full form is stored as individual tokens.
- Intensifiers such as 'AWESOME' are converted into lowercase and the token is stored as 'awesome'.
- Spelling of character repetitions such as 'veryyyy' are first corrected into 'very' and then stored as 'very'.
- The normalization process also discards all those tokens which, in no way, contribute to the sentiment of a tweet such tokens are called stop word. It also discards URL's.

For analyzing the tweets, we have to take polarity into consideration using various types of dictionaries.

1) Lexical Dictionary:

It mainly consists of most of the English words which will help us to analyze the tweets by matching the word in the tweet with the words in the lexical dictionary. It also consists of idioms, phrases, headwords and multiword.

2) Acronym Dictionary:

It is used to expand all the abbreviations and acronyms which will further generate words which can be analyzed using lexical dictionary.

3) Emoticon Dictionary:

A tweet containing emoticons can be analyzed by using this dictionary. Emoticons are basically the textual portrayal of the tweeter's mode which conveys some meaning.

4) Stop Words Dictionary:

These are the words in the tweet which do not have any polarity and they need not be analyzed. So they are eliminated and tagged as stop words. We maintain a dictionary with the list of all stop words for example able, are, both, etc.

Sentiment Classifier:

The tweets are broken down into tokens where each token is assigned polarity which is a floating point number ranging from 1 to -1.

A. Positive Tweets:

Positive tweets are the tweets which show a good or positive response towards something. For example tweets such as —It was an inspiring movie!!!! or —Best movie everl.

B. Negative Tweets:

Negative tweets can be classified as the tweets which show a negative response or oppose towards something. For example tweets such as —Waste of timel or —Worst movie everl.

C. Neutral Tweets:

Neutral tweets can be classified as the tweets which neither show a support or appreciate anything nor oppose or depreciate it. It also includes tweets which are facts or theories. For example tweets such as —Earth is roundl.

3. Part-of-speech Tagging:

The valid tokens are then passed to the part-of-speech tagger which attaches a tag to each token, specifying whether it's a noun, verb, adverb, adjective etc. Part-of-speech tagging helps

determine the sentiment of the overall tweet because words have different meanings when represented as different parts of speech.

4. Classification:

At the end system will classify the twitter data into Positive, Negative, Neutral with the help of data dictionary.

