INTRODUCTION

The aim of this project is to analyze structured or semi structured data using hadoop framework. Nowadays social media sites have enormous amount of data. Hence to deal with such enormous amount of data is a big challenge many companies faces.

Big data analytics is the process of examining large and varied data sets -- i.e., big data -- to uncover hidden patterns, unknown correlations, market trends, customer preferences and other useful information that can help organizations make more-informed business decisions.

YARN is a cluster management technology and one of the key features in second-generation Hadoop. Map Reduce, a software framework that allows developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of processors or stand-alone computers.

Flume is a reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flow. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses simple extensible data model that allows for online analytic application.

Hive is an open-source data warehouse system for querying and analyzing large datasets stored in Hadoop files. Pig is an open-source technology that offers a high-level mechanism for the parallel programming of Map Reduce jobs to be executed on Hadoop clusters.

SENTIMENTAL ANALYSIS

Sentiment analysis (sometimes known as **opinion mining** or **emotion AI**) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor).

A basic task in sentiment analysis is classifying the *polarity* of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy".

Even though in most statistical classification methods, the neutral class is ignored under the assumption that neutral texts lie near the boundary of the binary classifier, several researchers suggest that, as in every polarity problem, three categories must be identified. Moreover, it can be proven that specific classifiers such as the Max Entropy and the SVMs can benefit from the introduction of a neutral class and improve the overall accuracy of the classification. There are in principle two ways for operating with a neutral class. Either, the algorithm proceeds by first identifying the neutral language, filtering it out and then assessing the rest in terms of positive and negative sentiments, or it builds a three-way classification in one step. This second approach often involves estimating a probability distribution over all categories (e.g. naive Bayes classifiers as implemented by the NLTK). Whether and how to use a neutral class depends on the nature of the data: if the data is clearly clustered into neutral, negative and positive language, it makes sense to filter the neutral language out and focus on the polarity between positive and negative sentiments. If, in contrast, the data are mostly neutral with small deviations towards positive and negative effect, this strategy would make it harder to clearly distinguish between the two poles.

A different method for determining sentiment is the use of a scaling system whereby words commonly associated with having a negative, neutral, or positive sentiment with them are given an associated number on a −10 to +10 scale (most negative up to most positive) or simply from 0 to a positive upper limit such as +4. This makes it possible to adjust the sentiment of a given term relative to its environment (usually on the level of the sentence). When a piece of unstructured text is analyzed using natural language processing, each concept in the specified environment is given a score based on the way sentiment words relate to the concept and its associated score. This allows movement to a more sophisticated understanding of sentiment, because it is now possible to adjust the sentiment value of a concept relative to modifications that may surround it. Words, for example, that intensify, relax or negate the sentiment expressed by the concept can affect its score. Alternatively, texts can be given a positive and negative sentiment strength score if the goal is to determine the sentiment in a text rather than the overall polarity and strength of the text.

Twitter Data

As twitter post are very important source of opinion on different issues and topics. It can give a keen insight about a topic (GST in this case) and can be a good source of analysis. Analysis can help in decision making in various areas.

This project will give us hands on experience of handling and parallel processing of huge amount of data. Data collection process will introduce us to Java twitter streaming API. We will get exposure to work with prominent parallel data processing tool: Hadoop. Apache Hadoop framework is gaining significant momentum from both industry and academia as the volume of data to analyze growth rapidly. This project will help us not only to gain knowledge about installation and configuration of hadoop distributed file system but also map reduce programming model. Amongst the many fields of analysis, there is one field where humans have dominated the machines more than any – the ability to analyse sentiment, or sentiment analysis. The future of this data analysis field is vast. This project not only analyses the sentiments of the user but also computes other results like the user with maximum friends/followers, top tweets etc. hence hadoop can also be effectively used to compute such results in order to determine the current trends with respect to particular topics. This can be very useful in the marketing sector.

Software and Hardware requirements

Hardware Requirements

- Core i3 3rd Gen
- 8 GB DDR3 RAM
- Minimum of 10 GB hard disk

Software Requirements

- VMware Workstation Pro
- Hadoop-2.8.0.tar
- Apache-flume-1.7.0-bin.tar
- Apache-hive-1.2.2-bin.tar
- Microsoft Excel 2010
- OS- Ubuntu, Windows 10

Software Requirement Analysis

Problem

Classification of the polarity of a given text in the document, sentence or phrase.

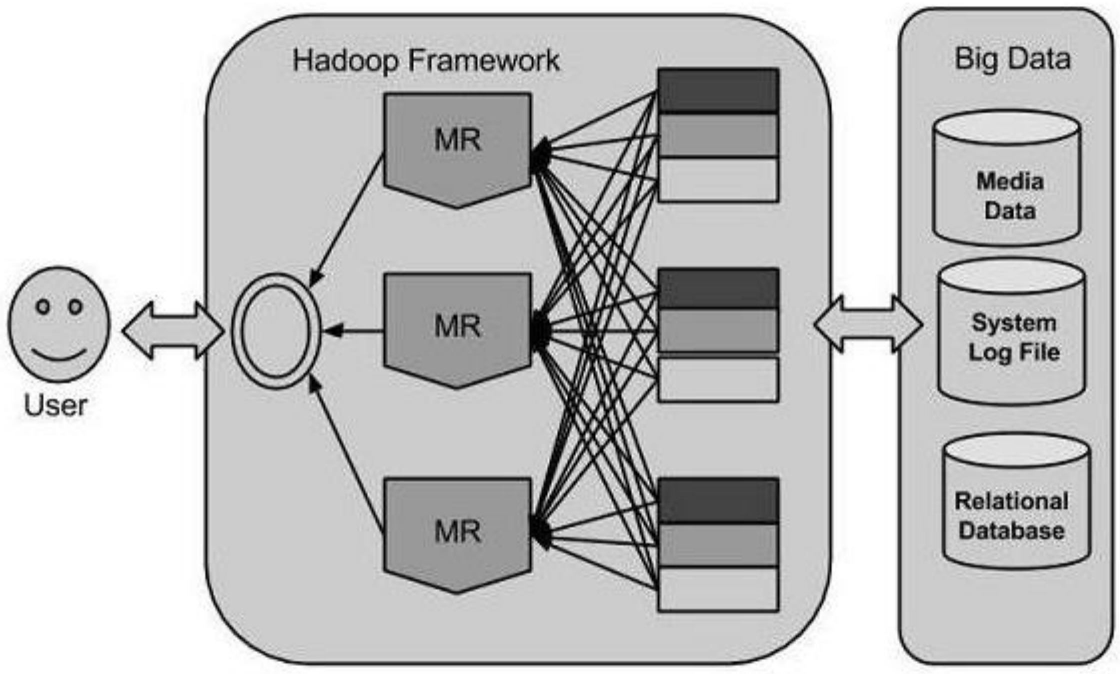
The goal is to determine whether the expressed opinion in the text is positive, negative or neutral.

In this project, first installation of hadoop along with hive and flume has been done. After this, data extracted using them. Data is visualized in graphical representation with the help of excel. They have been explained as follow

Hadoop

Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing).

It is being used by Facebook, Yahoo, Google, Twitter, LinkedIn and many more. Hadoop uses HDFS, a distributed file system based on GFS, as its shared file system. HDFS architecture divides files into large chunks (~64MB) distributed across data servers



How Does Hadoop Work?

Stage 1

A user/application can submit a job to the Hadoop (a hadoop job client) for required process by specifying the following items: The location of the input and output files in the distributed file system.

The java classes in the form of jar file containing the implementation of map and reduce functions. The job configuration is done by setting different parameters specific to the job.

Stage 2

The hadoop job client then submits the job (jar/executable etc) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

Stage 3

The TaskTrackers on different nodes execute the task as per MapReduce implementation and output of the reduce function is stored into the output files on the file system.

Following qualities make Hadoop stand out of the crowd:

- Single namespace by HDFS makes content visible across all the nodes
- Easily administered using High Performance Computing (HPC)

- Querying and managing distributed data are done using Hive
- Pig facilitates analyzing the large and complex datasets on Hadoop
- HDFS is designed specially to give high throughput instead of low latency.

History of Hadoop

Hadoop was created by Doug Cutting and hence was the creator of Apache Lucene. It is the widely used text to search library. Hadoop has its origins in Apache Nutch which is an open source web search engine itself a part of the Lucene project.

HDFS(Module of Hadoop)

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application. It states that the files will be broken into blocks and stored in nodes over the distributed architecture.

HDFS nodes

NameNode

– Only one per Hadoop cluster

– Stores meta data

– It does not store actual data

– Single point failure

DataNode

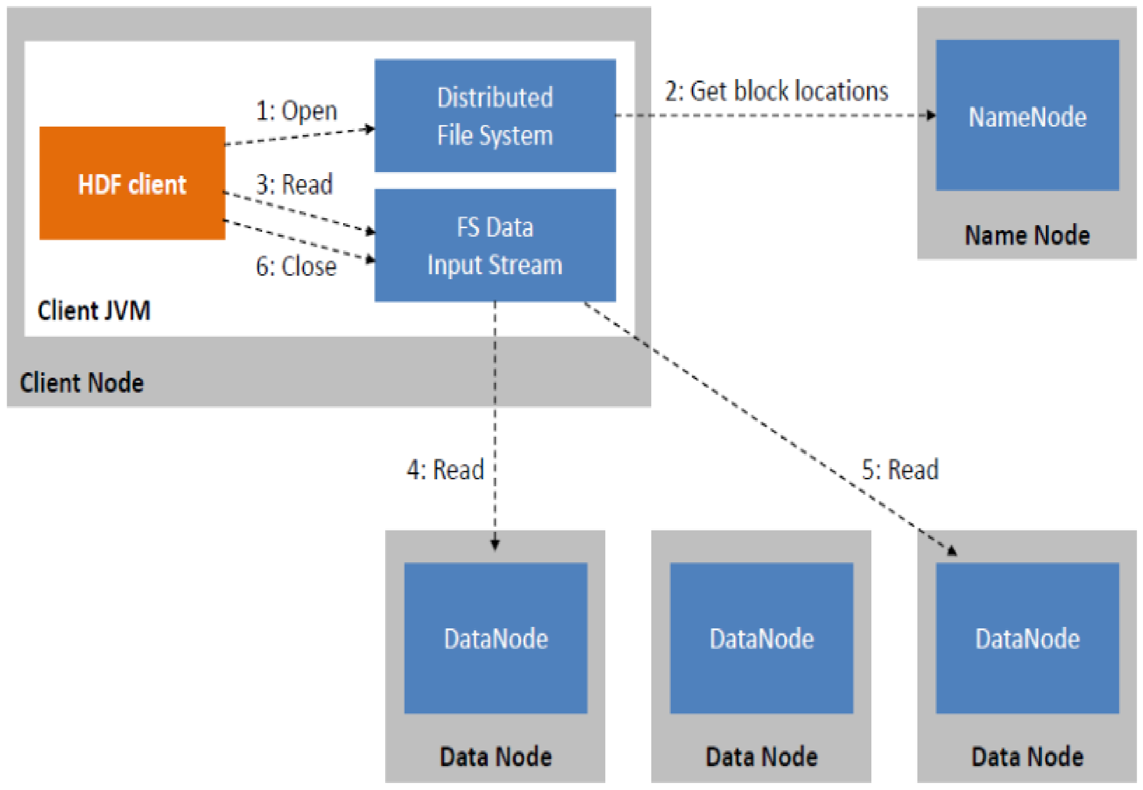
– Many per Hadoop cluster

– Manages blocks with data and serves them to clients

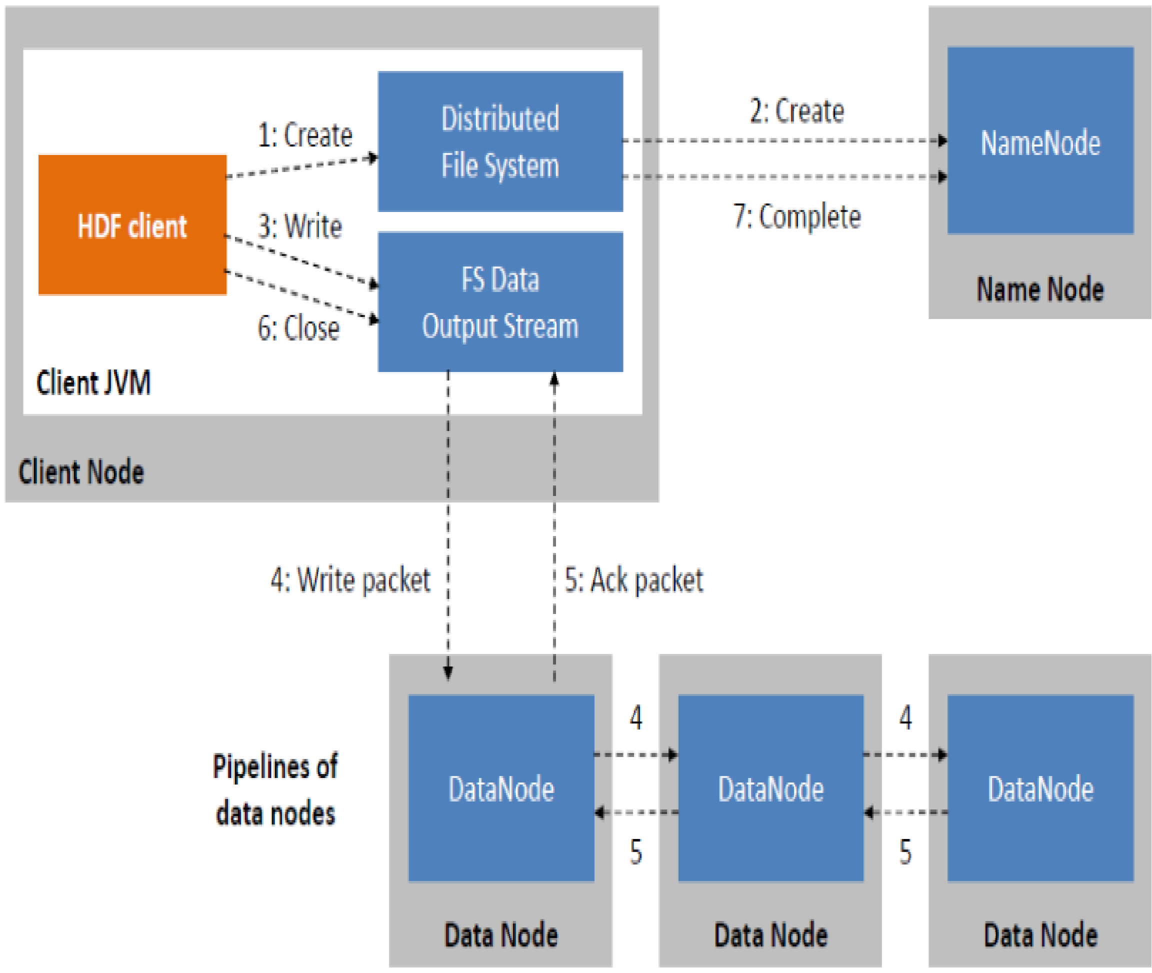
– Periodically reports to name node the list of blocks it stores

ARCHITECTURE SHOWING HDFS READ AND WRITE OPERATIONS:

HDFS Read Image:



HDFS Write Image:



HDFS Commands

The important operations of Hadoop Distributed File System using the shell commands used for file management in the cluster.

1. Directory creation in HDFS for a specific given path are given as.

```
hadoop fs-mkdir<paths>
```

Example:

```
hadoopfs-mkdir/user/saurzcode/dir1/user/saurzcode/dir2
```

2. Listing of the directory contents.

```
Hadoop fs-ls<args>
```

Example:

```
hadoopfs-ls/user/saurzcode
```

3. HDFS file Upload/download.

Upload:

```
hadoopfs -put:
```

Copy single source file, or multiple source files based on local file system to the Hadoop data file system

```
hadoopfs-put<localsrc> … <HDFS_dest_Path>
```

Example:

```
hadoopfs-put/home/saurzcode/Samplefile.txt/user/saurzcode/dir3/
```

Download:

```
hadoopfs -get:
```

Copies or downloads the files to the local file system

```
hadoopfs-get<hdfs_src><localdst>
```

Example:

```
hadoopfs-get/user/saurzcode/dir3/Samplefile.txt/home/
```

4. Viewing of file content

Same as Unix cat command:

hadoopfs-cat<path[filename]>

Example:

hadoopfs-cat/user/saurzcode/dir1/abc.txt

5. File copying from source to destination

hadoopfs-cp<source><dest>

Example:

hadoopfs-cp/user/saurzcode/dir1/abc.txt/user/saurzcode/dir2

6. File moving from source to destination.

But remember, you cannot move files across filesystem.

hadoopfs-mv<src><dest>

Example:

hadoopfs-mv/user/saurzcode/dir1/abc.txt/user/saurzcode/dir2

7. File or directory removal in HDFS.

hadoopfs-rm<arg>

Example:

hadoopfs-rm/user/saurzcode/dir1/abc.txt

Map Reduce:

Map reduce is mainly a data processing component of Hadoop. It is a programming model for processing large number of data sets.

It contains the task of data processing and distributes the particular tasks across the nodes. It consists of two phases –

- Map
- Reduce

Map converts a typical dataset into another set of data where individual elements are divided into key/value pairs

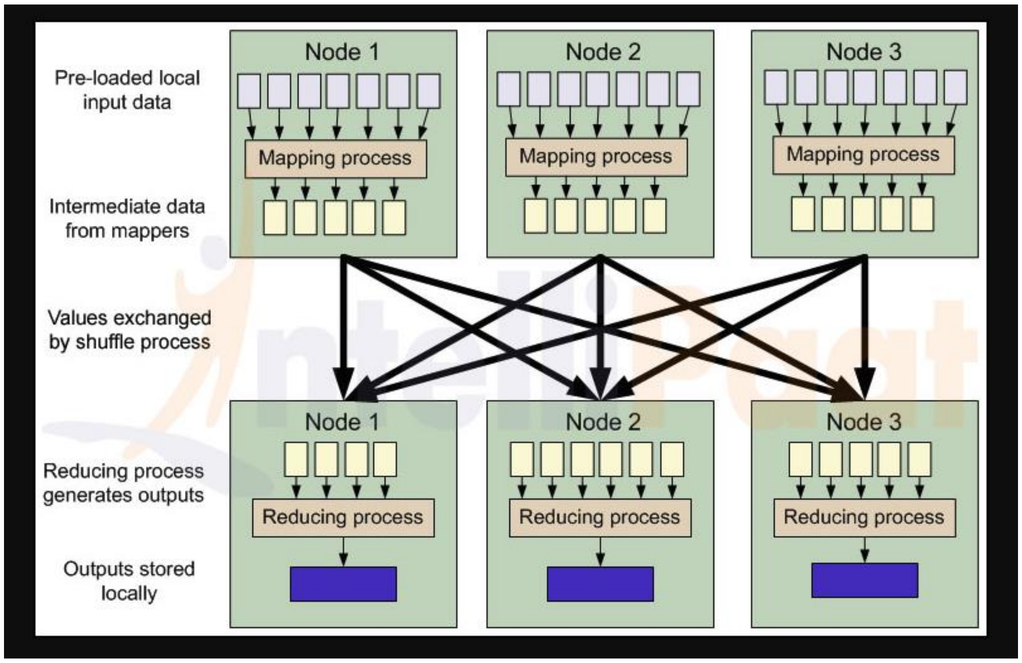
Reduce task takes the output files from a map considering as an input and then integrate the data tuples into a smaller set of tuples. Always it is been executed after the map job is done.

This is a framework which helps Java programs to do the parallel computation on data using key value pair. The Map task takes input data and converts it into a data set which can be computed in Key value pair. The output of Map task is consumed by reduce task and then the out of reducer gives the desired result.

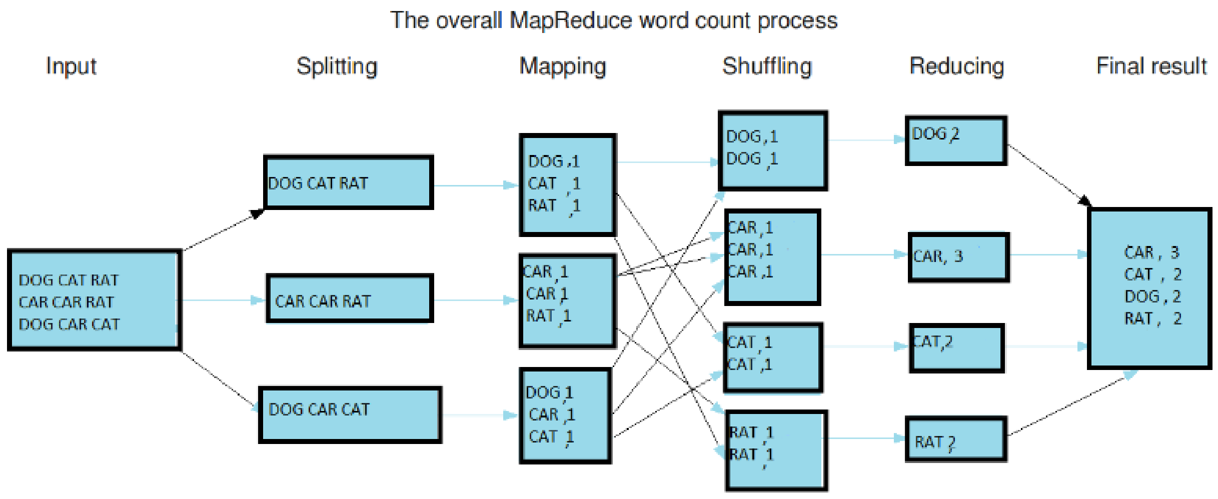
Map reduce simple steps follow:

1. Executes map function on each input is received
2. Map function emits key, value pair
3. Shuffle, Sort and Group the outputs
4. Executes the reduce function on the group
5. Emits the output results is given per group basis.

<u>MAPREDUCE PROCESS SHOWN BELOW:</u>


Pre-loaded local input data
Intermediate data from mappers
Values exchanged by shuffle process
Reducing process generates outputs
Outputs stored locally
Node 1
Node 2
Node 3
Mapping process
Reducing process
Intellipaat

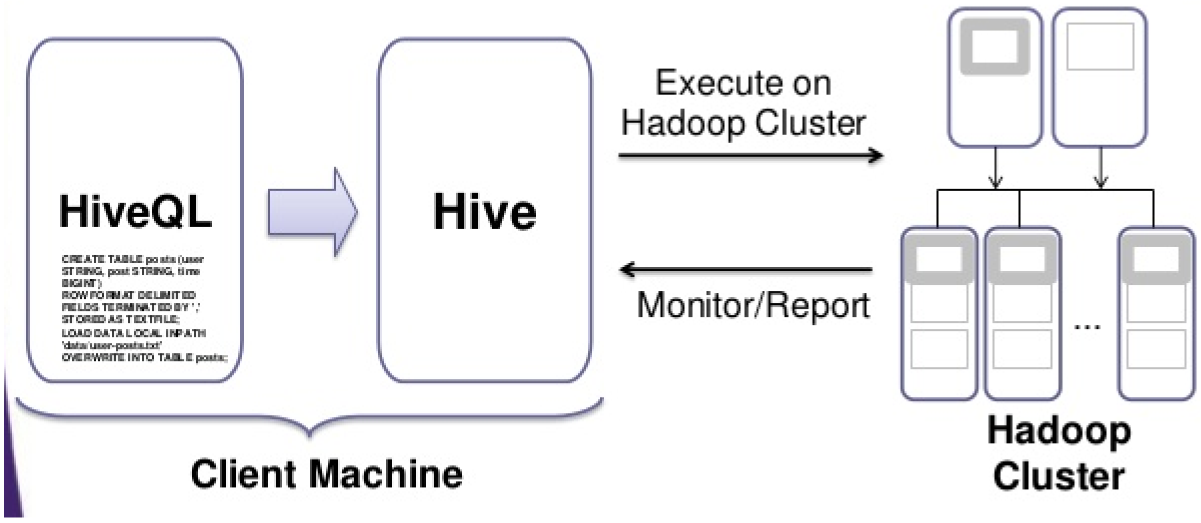
ANOTHER EXAMPLE SHOWING MAPREDUCE WORD COUNT PROCESS



HIVE

Hive is a data ware house system for Hadoop. It runs SQL like queries called HQL (Hive query language) which gets internally converted to map reduce jobs. Hive supports Data definition Language(DDL), Data Manipulation Language(DML) and user defined functions.

- Hive is not RDBMS.
- Hive Query Language is similar to SQL and gets reduced to map reduce jobs in backend.
- Hive's default database is derby.
- Hive was developed by Facebook.



```
CREATE TABLE posts (user
STRING, post STRING, time
BIGINT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
STORED AS TEXTFILE;
LOAD DATA LOCAL INPATH
'data/user-posts.txt'
OVERWRITE INTO TABLE posts;
```

FIGURE SHOWING HOW HIVE WORKS

CONSIDER A TEXT FILE WITH THE FOLLOWING CONTENTS

```
1|1|Dharmender|kumar

2|1|Amit sharma|golu:g:gb

3|2|Gourav|Neetu:lalu

4|3|Vejay|sonia:sagar:himsuhu:manu
```

Commands :

```
hive>create table com(id int , officeid int , name string , child Array<string>) row format delimited fields terminated by '|' collection items terminated by ':';
```

```
hive> load data local inpath 'souce file path’ overwrite into table com;
```

This command load the data from the file into the table created using the first command.

```
hive> select id , offid from com;
```

This is used to retrieve id and offid from the table com.

```
hive>select id ,name ,child[2] from com where id = 3;
```

Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date, city, and department. Using partition, it is easy to query a portion of the data.

Tables or partitions are sub-divided into buckets, to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

For example, a table named Tab1 contains employee data such as id, name, dept, and yoj (i.e.,year of joining). Suppose you need to retrieve the details of all employees who joined in 2012. A query searches the whole table for the required information. However, if you partition the employee data with the year and store it in a separate file, it reduces the query processing time. The following example shows how to partition a file and its data:

The following file contains employeedata table.

```
/tab1/employeedata/file1

id, name, dept, yoj

1, gopal, TP, 2012

2, kiran, HR, 2012

3, kaleel,SC, 2013

4, Prasanth, SC, 2013
```

The above data is partitioned into two files using year.

```
/tab1/employeedata/2012/file2

1, gopal, TP, 2012

2, kiran, HR, 2012
```

```
/tab1/employeedata/2013/file3

3, kaleel,SC, 2013

4, Prasanth, SC, 2013

hive> ALTER TABLE employee

> ADD PARTITION (year='2013')

> location '/2012/part2012';
```

Why hive?

It consists of a query language based on the standard SQL instead of giving a rapid development of map and reduces tasks. Hive takes HiveQL statements and then automatically transforms each and every query into one or more MapReduce jobs. Later it runs the overall MapReduce program and executes the output to the user whereas Hadoop streaming decreases the mandatory code, compile, and submit cycle. Hive removes it completely instead requires only the composition of HiveQL statements.

Flume

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

Flume's high-level architecture is focused on delivering a streamlined codebase that is easy-to-use and easy-to-extend. The project team has designed Flume with the following components:

Event – a singular unit of data that is transported by Flume (typically a single log entry

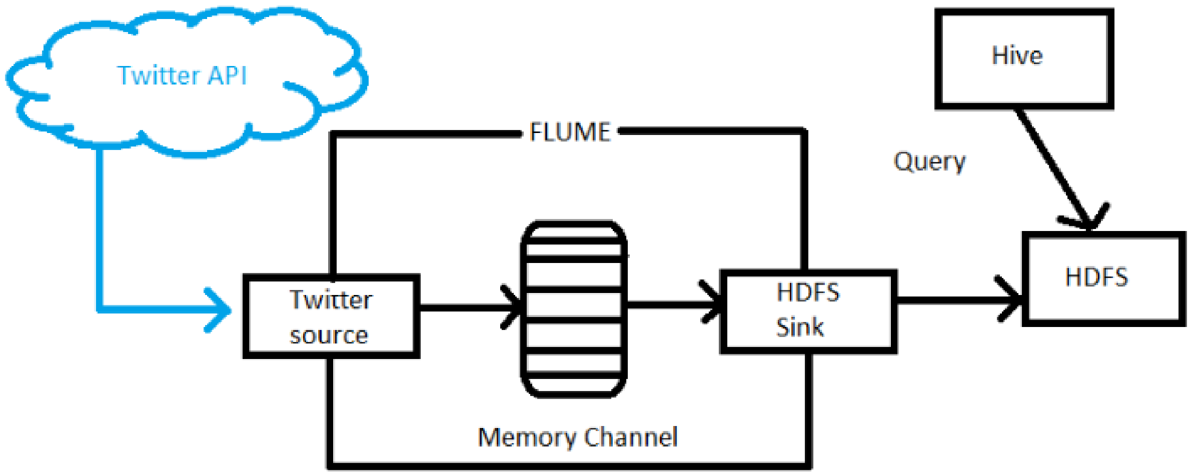
Source – the entity through which data enters into Flume. Sources either actively poll for data or passively wait for data to be delivered to them. A variety of sources allow data to be collected, such as log4j logs and syslogs.

Sink – the entity that delivers the data to the destination. A variety of sinks allow data to be streamed to a range of destinations. One example is the HDFS sink that writes events to HDFS.

Channel – the conduit between the Source and the Sink. Sources ingest events into the channel and the sinks drain the channel.

Agent – any physical Java virtual machine running Flume. It is a collection of sources, sinks and channels.

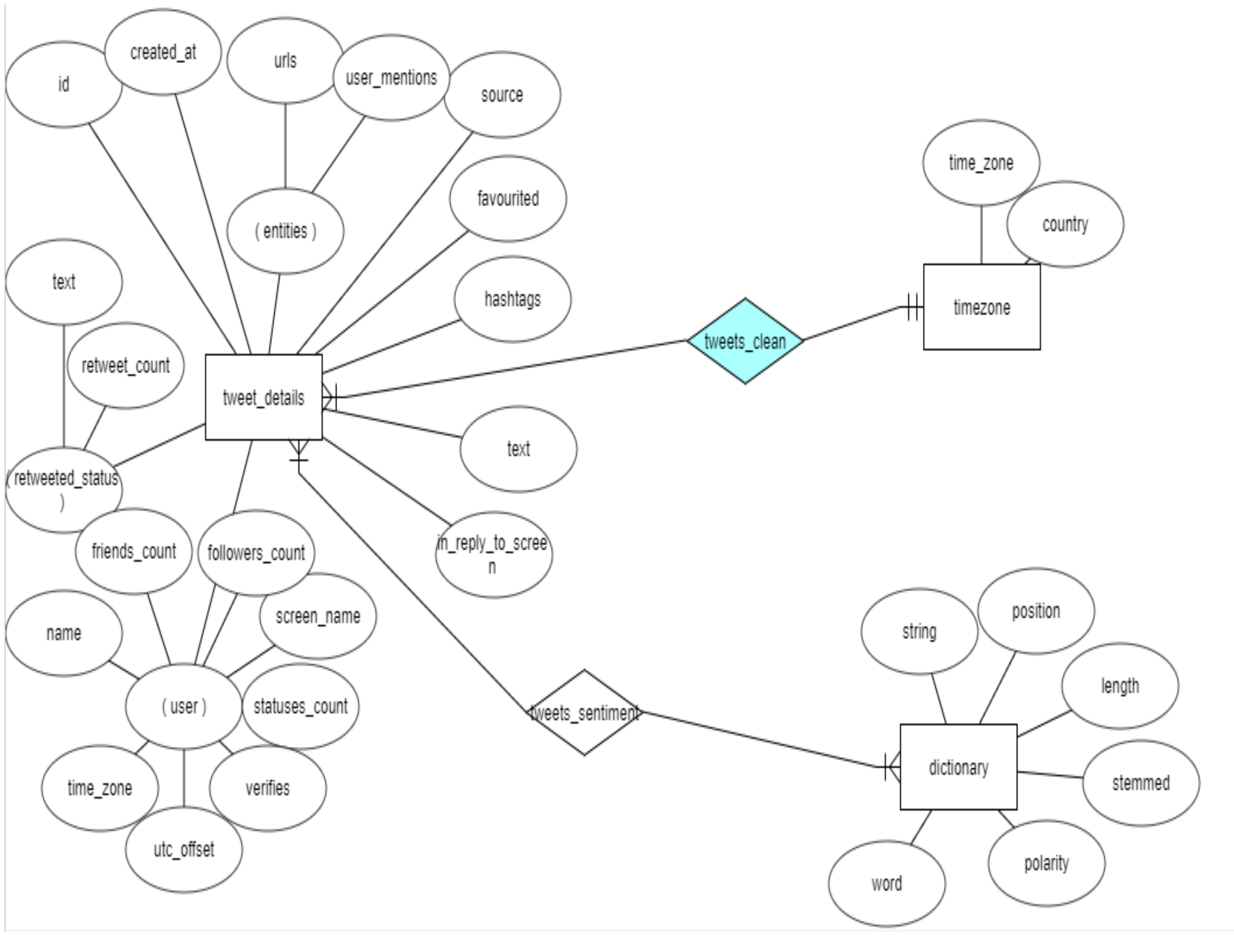
Client – produces and transmits the Event to the Source operating within the Agent



Microsoft Excel

Microsoft Excel has the basic features of all spreadsheets, using a grid of *cells* arranged in numbered *rows* and letter-named *columns* to organize data manipulations like arithmetic operations. It has a battery of supplied functions to answer statistical, engineering and financial needs. In addition, it can display data as line graphs, histograms and charts, and with a very limited three-dimensional graphical display. It allows sectioning of data to view its dependencies on various factors for different perspectives (using *pivot tables* and the *scenario manager*). It has a programming aspect, *Visual Basic for Applications*, allowing the user to employ a wide variety of numerical methods, for example, for solving differential equations of mathematical physics, and then reporting the results back to the spreadsheet. It also has a variety of interactive features allowing user interfaces that can completely hide the spreadsheet from the user, so the spreadsheet presents itself as a so-called *application*, or *decision support system* (DSS), via a custom-designed user interface, for example, a stock analyzer, or in general, as a design tool that asks the user questions and provides answers and reports. In a more elaborate realization, an Excel application can automatically poll external databases and measuring instruments using an update schedule, analyze the results, make a Word report or PowerPoint slide show, and e-mail these presentations on a regular basis to a list of participants. Excel was not designed to be used as a database.

E-R Diagram


id
created_at
urls
user_mentions
source
(entities)
favourited
hashtags
text
retweet_count
tweet_details
(retweeted_status)
text
in_reply_to_screen
friends_count
followers_count
screen_name
name
(user)
statuses_count
time_zone
verifies
utc_offset
tweets_clean
time_zone
country
timezone
tweets_sentiment
string
position
length
dictionary
stemmed
polarity
word

CODE

INSTALLATION

Hadoop

- Install java

```
sudo apt-get update
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get install oracle-java8-installer
```

- Generate SSH keys and give permission

```
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
ssh-copy-id -i $HOME/ .ssh/id_rsa.pub ubuntu@localhost
chmod 700 ~/.ssh/authorized_keys
```

- Extract hadoop

```
 tar xvzf hadoop-2.6.0.tar.gz
```

Edit configration file hadoop-2.6.0/etc/hadoop/hadoop-env.sh and set JAVA_Home

export JAVA_HOME=path to be the root of your java installation(eg: /usr/lib/jvm/java-7-oracle)

path is "/hadoop-2.6.0/etc/hadoop/hadoop-env.sh"

java is install in "cd hadoop-2.6.0" $ls /usr/lib/jvm

edit .bashrc file

```
export HADOOP_HOME=/home/ubuntu/hadoop_home
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export JAVA_HOME=/usr/lib/jvm/java-7-oracle
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
```

Edit configuration file conf/core-site.xml and add following entries:

```
<configuration>
 <property>
  <name>hadoop.tmp.dir</name>
  <value>/home/ubuntu/hadoop_home/tmp</value>
  <description>A base for other temporary directories.</description>
 </property>

 <property>
  <name>fs.default.name</name>
  <value>localhost:9000</value>
   </property>
</configuration>
```

 Edit configuration file conf/hdfs-site.xml and add following entries:

```
<configuration>
 <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.name.name.dir</name>
   <value>/home/ubuntu/hadoop_home/hdfs/name</value>
 </property>
 <property>
   <name>dfs.data.data.dir</name>
   <value>/home/ubuntu/hadoop_home/hdfs/name</value>
 </property>
</configuration>
```

Edit configuration file hadoop/etc/hadoop/mapred-site.xml add following entires

```
<configuration>
 <property>
  <name>mapred.job.tracker</name>
  <value>localhost:9001</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>
</configuration>
```

Format namenode

```
hadoop namenode -format
start-all.sh
```

To check if hadoop working properly

```
jps
```

To make folders on hdfs

```
hadoop fs -mkdir /user


hadoop fs -mkdir /user/hive


hadoop fs -mkdir /user/flume
```

To open hdfs on browser

http://127.0.0.1:50070/explorer.html#/

HIVE

```
tar xvzf apache-hive-1.2.2-bin.tar
```

edit .bashrc file

```
export HIVE_HOME=/usr/local/hive
export PATH=$PATH:$HIVE_HOME/bin
export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
```

Edit the **hive-env.sh** file by appending the following line:

```
export HADOOP_HOME=/usr/local/hadoop
```

Create **Hive** directories within **HDFS**. The directory **'warehouse'** is the location to store the table or data related to hive.

- hdfs dfs -chmod g+w /user/hive/warehouse

- hdfs dfs -chmod g+w /tmp

Edit hive-site.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?><!--
Licensed to the Apache Software Foundation (ASF) under one or more
```

```
contributor license agreements. See the NOTICE file distributed with

this work for additional information regarding copyright ownership.

The ASF licenses this file to You under the Apache License, Version 2.0

(the "License"); you may not use this file except in compliance with

the License. You may obtain a copy of the License at


http://www.apache.org/licenses/LICENSE-2.0


Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.

See the License for the specific language governing permissions and

limitations under the License.

-->

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/home/edureka/apache-hive-2.1.0-
bin/metastore_db;create=true</value>

<description>

JDBC connect string for a JDBC metastore.

To use SSL to encrypt/authenticate the connection, provide database-specific SSL
flag in the connection URL.

For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.

</description>

</property>
```

```
<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

<description>location of default database for the warehouse</description>

</property>

<property>

<name>hive.metastore.uris</name>

<value/>

<description>Thrift URI for the remote metastore. Used by metastore client to
connect to remote metastore.</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>org.apache.derby.jdbc.EmbeddedDriver</value>

<description>Driver class name for a JDBC metastore</description>

</property>

<property>

<name>javax.jdo.PersistenceManagerFactoryClass</name>

<value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>

<description>class implementing the jdo persistence</description>

</property>

</configuration>
```

Initialize Derby database

```
bin/schematool -initSchema -dbType derby
```

Launch Hive

```
hive
```

Flume

```
tar  xvzf  apache-flume-1.7.0-bin.tar
```

Edit .bashrc file

```
export FLUME_HOME=/home/user/apache-flume-1.6.0-bin/
export PATH=$PATH:$FLUME_HOME/bin/
```

In flume.conf file, edit keys generated as shown later, put path of hdfs, and wirte the domain name i.e. #gst in it

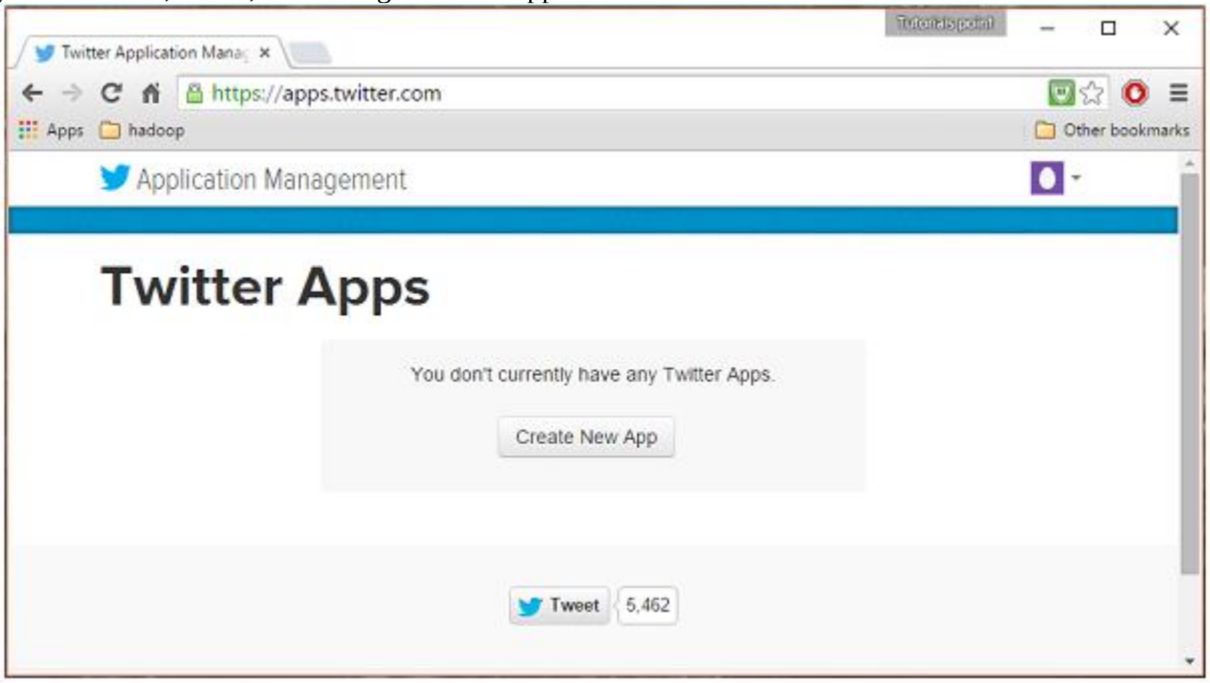
flume.conf file

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS
# Describing/Configuring the source
TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource
TwitterAgent.sources.Twitter.consumerKey=RvS9ZVJcUleyaj2jhtCwgh9HV
TwitterAgent.sources.Twitter.consumerSecret=m8u5N0PS9prtEn0cxGUPuzbc0Xn
niu0EQbDqAwtMYYTRfKN6QI
TwitterAgent.sources.Twitter.accessToken=886451470312194048-
uRGYX98UfiKvVih4vDJxDMAqJww069m
TwitterAgent.sources.Twitter.accessTokenSecret=lT5ePiwQT5TMA1rIsavET4Z2k
Tjd1v5bFpNQXv1brYT8C
TwitterAgent.sources.Twitter.keywords=#GST
# Describing/Configuring the sink
#TwitterAgent.sources.Twitter.keywords= hadoop,election,sports, cricket,Big data
TwitterAgent.sinks.HDFS.channel=MemChannel
TwitterAgent.sinks.HDFS.type=hdfs
TwitterAgent.sinks.HDFS.hdfs.path=hdfs://localhost:9000/user/flume/gst
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
TwitterAgent.sinks.HDFS.hdfs.rollInterval=600
TwitterAgent.channels.MemChannel.type=memory
TwitterAgent.channels.MemChannel.capacity=10000
TwitterAgent.channels.MemChannel.transactionCapacity=1000
TwitterAgent.sources.Twitter.channels = MemChannel
TwitterAgent.sinks.HDFS.channel = MemChannel
```

Steps to generate various twitter api keys

Step-1

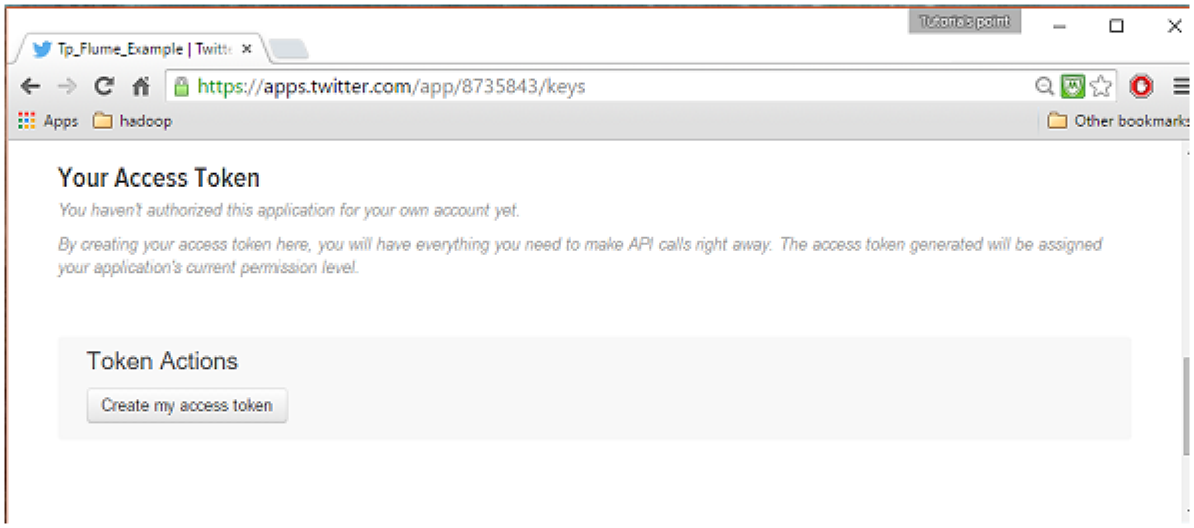
To create a Twitter application, click on the following link https://apps.twitter.com/. Sign in to your Twitter account. You will have a Twitter Application Management window where you can create, delete, and manage Twitter Apps.



You don't currently have any Twitter Apps.

Step 2

Under keys and Access Tokens tab at the bottom of the page, you can observe a button named Create my access token. Click on it to generate the access token



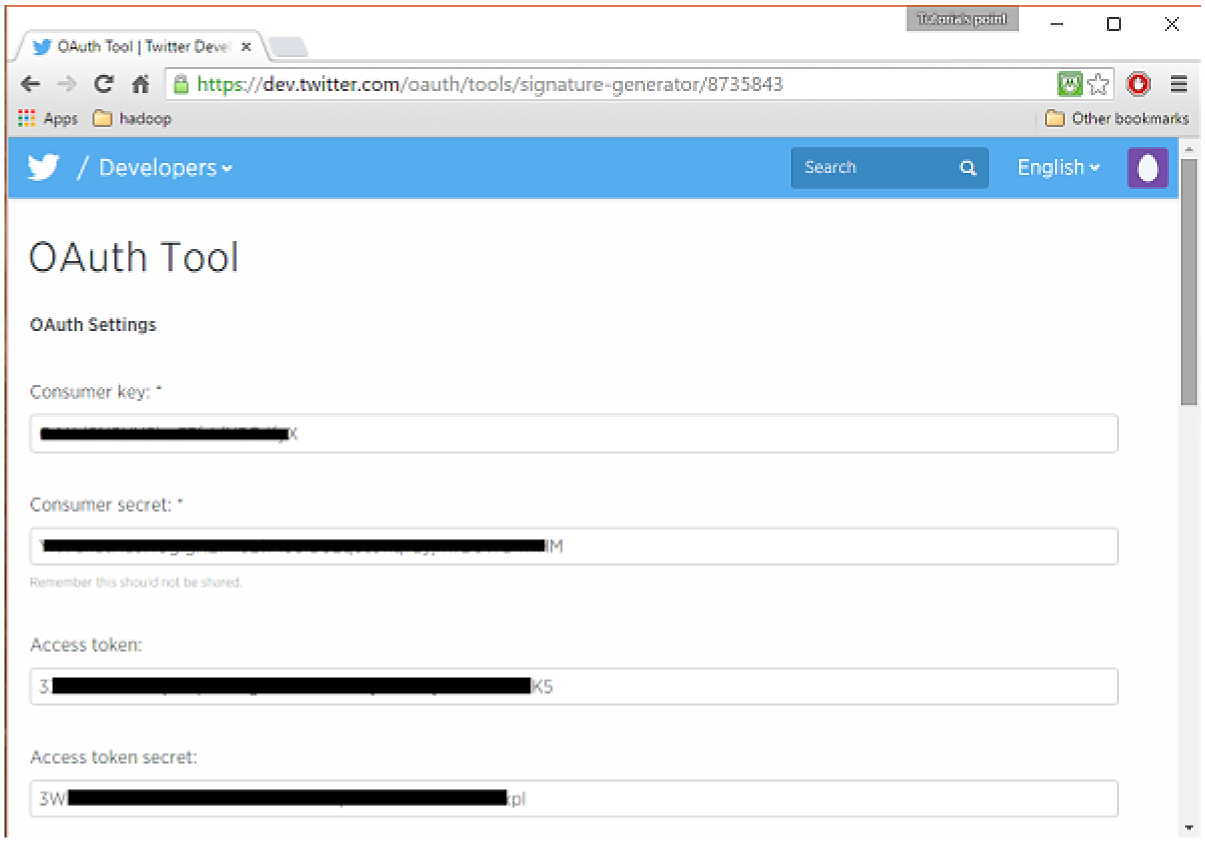
Your Access Token

You haven't authorized this application for your own account yet.

By creating your access token here, you will have everything you need to make API calls right away. The access token generated will be assigned your application's current permission level.

Step 3

Finally, click on the Test OAuth button which is on the right side top of the page. This will lead to a page which displays your Consumer key, Consumer secret, Access token, and Access token secret. Copy these details. These are useful to configure the agent in Flume.


OAuth Tool | Twitter Devel
https://dev.twitter.com/oauth/tools/signature-generator/8735843
Apps
hadoop
Other bookmarks
/ Developers
Search
English

OAuth Tool

OAuth Settings

Consumer key: *

[redacted]X

Consumer secret: *

[redacted]IM

Remember this should not be shared.

Access token:

3[redacted]K5

Access token secret:

3W[redacted]pl

In hive create the table given below

hive>ADD JAR file:///home/hadoop/work/hive-0.10.0/lib/hive-serdes-1.0-SNAPSHOT.jar

hive> CREATE TABLE tweet_details(

id BIGINT,

created_at STRING,

source STRING,

```
favorited BOOLEAN,
retweeted_status STRUCT<
text:STRING,
user:STRUCT<screen_name:STRING,name:STRING>,
retweet_count:INT>,
entities STRUCT<
urls:ARRAY<STRUCT<expanded_url:STRING>>,
user_mentions:ARRAY<STRUCT<screen_name:STRING,name:STRING>>,
hashtags:ARRAY<STRUCT<text:STRING>>>,
text STRING,
user STRUCT<
screen_name:STRING,
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 'com.cloudera.hive.serde.JSONSerDe';
```

Extract twitter data

```
flume-ng agent -n TwitterAgent -f /usr/lib/flume/conf/flume.conf -
Dtwitter4j.streamBaseURL=https://stream.twitter.com/1.1/
```

Download the file and paste it on desktop

In hive run the following command

```
hive> load data inpath '/home/hadoop/work/flumedata' into table tweet_details;
```

By this, data is now extracted.

Create sentiment dictionary

```
CREATE TABLE dictionary (
    type string,
    length int,
    word string,
    pos string,
    stemmed string,
    polarity string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;

CREATE TABLE timezone (
    time_zone string,
    country string,
    notes string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE;
```

Clean up tweets

```
CREATE VIEW tweets_simple AS
```

```
SELECT
  id,
  cast ( from_unixtime( unix_timestamp(concat( '2013 ', substring(created_at,5,15)),
'yyyy MMM dd hh:mm:ss')) as timestamp) ts,
  text,
  user.time_zone
FROM tweet_details
;


CREATE VIEW tweets_clean AS
SELECT
  id,
  ts,
  text,
  m.country
 FROM tweets_simple t LEFT OUTER JOIN timezone m ON t.time_zone =
m.time_zone;
```

Compute sentiment

- Explode each tweet text in array of words.
  - Example: 330166074362433536 ["waiting","for","iron","man","3","to","start"]

```
create view l1 as select id, words from tweets_raw lateral view
explode(sentences(lower(text))) dummy as words;
```

- Explode each tweet -> word on multiple rows
  - 330166074362433536 waiting
  - 330166074362433536 for
  - 330166074362433536 iron

- 330166074362433536 man
- 330166074362433536 3
- 330166074362433536 to
- 330166074362433536 start

```
create view l2 as select id, word from l1 lateral view explode( words ) dummy as word ;
```

Use the dictionary file to score the sentiment of each Tweet by the number of positive words compared to the number of negative words, and then assigned a positive, negative, or neutral sentiment value to each Tweet.

```
create view l3 as select
    id,
    l2.word,
    case d.polarity
      when  'negative' then -1
      when 'positive' then 1
      else 0 end as polarity
 from l2 left outer join dictionary d on l2.word = d.word;


 create table tweets_sentiment stored as orc as select
  id,
  case
    when sum( polarity ) > 0 then 'positive'
    when sum( polarity ) < 0 then 'negative'
    else 'neutral' end as sentiment
 from l3 group by id;
```

CREATE TABLE tweetsbi AS SELECT t.*, s.sentiment FROM tweets_clean t LEFT OUTER JOIN tweets_sentiment s on t.id = s.id;

CREATE TABLE tweetsbiaggr AS SELECT country,sentiment, count(sentiment) as tweet_count FROM tweetsbi group by country,sentiment;

CREATE VIEW **A** as select country,tweet_count as positive_response from tweetsbiaggr where sentiment='positive';

CREATE VIEW **B** as select country,tweet_count as negative_response from tweetsbiaggr where sentiment='negative';

CREATE VIEW **C** as select country,tweet_count as neutral_response from tweetsbiaggr where sentiment='neutral';

CREATE TABLE **tweetcompare** as select A.*,B.negative_response as negative_response,C.neutral_response as neutral_response from A join B on A.country= B.country join C on B.country=C.country;

QUERIES

1.Query for A:

Select * from A;

// to find no. of Positive tweets country wise

2.Query for B:

Select * from B;

//// to find no. of Negaive tweets country wise

3.Query for C:

Select * from C;

//// to find no. of Neutral tweets country wise

4. Query for tweetcompare

Select * from tweetcompare;

// to compare sentiments of India and USA

****THE ABOVE 4 QUERIES ARE IMPLEMENTED IN THE BASH SCRIPT: Sentiment_based.sh**

Query for most retweets:

1. SELECT t.retweeted_screen_name, sum(retweets) AS total_retweets, count(*) AS tweet_count FROM (SELECT retweeted_status.user.screen_name as retweeted_screen_name, retweeted_status.text, max(retweeted_status.retweet_count) as retweets FROM tweet_details GROUP BY retweeted_status.user.screen_name, retweeted_status.text) t GROUP BY t.retweeted_screen_name ORDER BY total_retweets DESC LIMIT 10;

*// **to determine who is the most influential person in a particular field is to to figure out whose tweets are re-tweeted the most.***

Query for most followers:

2.** select user.screen_name, user.followers_count c from tweet_details order by c desc; // **to know which user has the most number of followers

*****THE ABOVE 2 SCRIPTS ARE IMPLEMENTED IN THE BASH SCRIPT: tweet_based.sh***

Query for most popular hashtag:

select entities.hashtags.text[0], count(*) as totalcount from tweet_details group by entities.hashtags.text[0] order by totalcount desc limit 10;

COMMAND TO CONVERT QUERY RESULT TO CSV FILE IN LOCAL SERVER:

hive -e 'write query,eg: select books from table' > /home/lvermeer/temp.tsv(path)

Bashscripts

intro.sh

```
clear
echo
"*****************************************************************
*******************************************************"
figlet BIG DATA PROJECT
echo
"*****************************************************************
*******************************************************"
echo " "
figlet MADE BY – SIMRANJEET SINGH
echo " "
var=
echo "Press X to CONTINUE"
```

```
read var
case $var in
X)./menu.sh;;
*) echo "Enter a valid one"
esac
```

menu.sh

```
#!/bin/bash
clear
select=
echo "Welcome to --- TWITTER SENTIMENT ANALYSIS"
echo""
echo "Select the category for you analysis"
echo""
echo "1 Sentiment based"
echo""
echo "2 Tweet based"
echo""
echo "E Exit Analysis"
read select
case $select in
1) ./sentiment_based.sh;;
2) ./tweet_based.sh;;
E) ./exits.sh;;
*) echo "Enter a valid one"
esac
```

sentiment_based.sh

```
echo "Sentient Based"
echo ""
echo "Press 1 to find the number of POSITIVE tweets from every country"
echo "Press 2 to find the number of NEGATIVE tweets from every country"
echo "Press 3 to find the number of NEUTRAL tweets from every country"
echo "Press 4 to compare sentiments of India and U.S.A"

echo "Press X to GO TO MAIN MENU"
echo "Press E to Exit"

read var
case $var in
   1) hive  <<EOF
   add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
         use minuse minorProject;
```

```
select * from A;

EOF
        ;;
   2) hive  <<EOF
   add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
        use minorProject;
select * from B;
EOF
        ;;
    3)  hive <<EOF
add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
        use minorProject;
       select * from C;
EOF
        ;;
    4) hive <<EOF
add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
        use minorProject;
select * from tweetcompare;
EOF
        ;;



     X) ./menu.sh ;;
     E) ./exits.sh ;;
*) echo "Enter a valid one"
esac
```

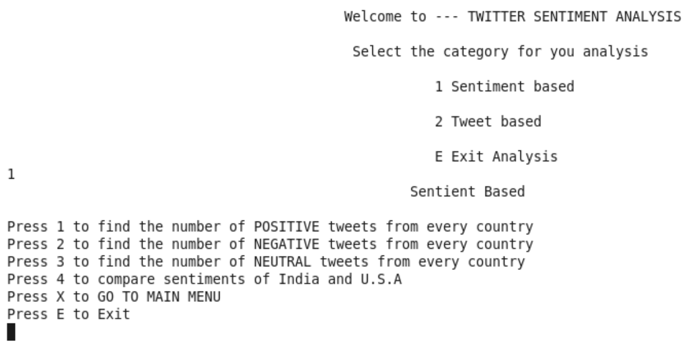
tweet_based.sh

```
echo "TWEET BASED"
echo ""
echo "Press 1 to categorise all hashtags"
echo "Press 2 to find number of tweet according to time zone"
echo "Press 3 to find most popular Hashtags"
echo "Press 4 to find the number of retweets of each user from distinct time zones"

echo "Press X to GO TO MAIN MENU"
echo "Press E to Exit"

read var
case $var in
       1) hive  <<EOF
   add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
```

```
        use minorProject;
select distinct(entities.hashtags.text[0]) from tweet_details limit 10;
EOF
        ;;

2) hive  <<EOF
    add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
            use minuse minorProject;
 select user.time_zone,count(*) as totalcount from tweet_details group by
user.time_zone limit 5;
EOF
        ;;

    3)  hive <<EOF
add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
        use minorProject;
       select entities.hashtags.text[0], count(*) as totalcount from tweet_details group
by entities.hashtags.text[0] order by totalcount desc limit 10;
EOF
        ;;
    4) hive <<EOF
add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
        use minorProject;
select user.time_zone, entities.user_mentions[0].screen_name,
retweeted_status.retweet_count from tweet_details where user.time_zone is not
NULL;
EOF
        ;;

     X) ./menu.sh ;;
     E) ./exits.sh ;;
*) echo "Enter a valid one"
esac
exit.sh
clear
echo
"*****************************************************************
***************************"
figlet THANK YOU
echo
"*****************************************************************
***************************"
```

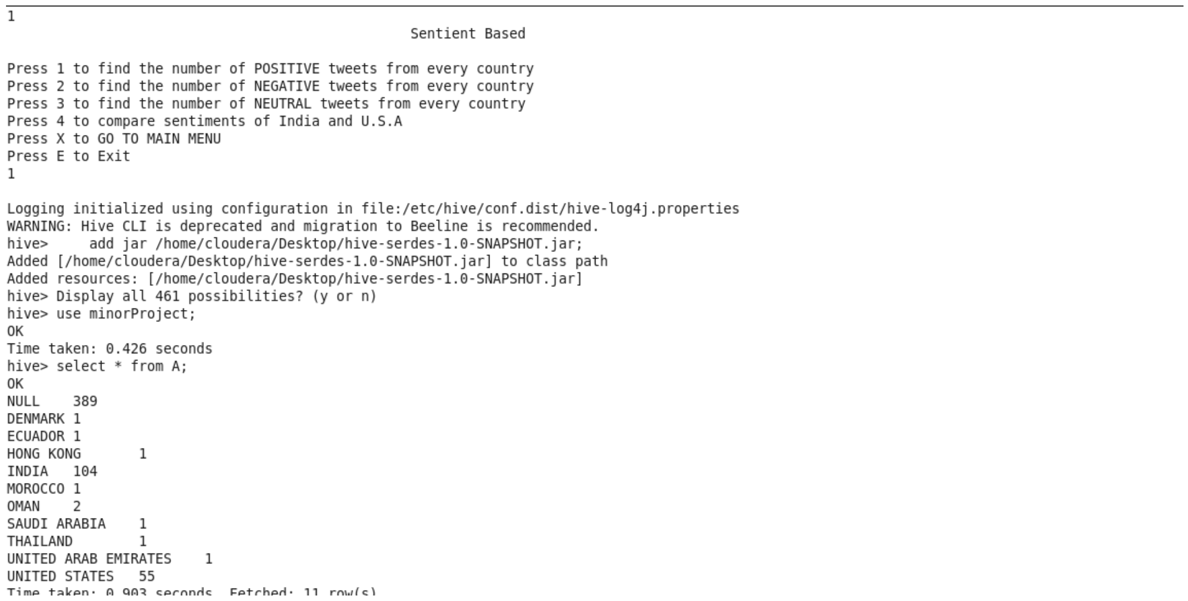
OUTPUT SCREENS

menu.sh



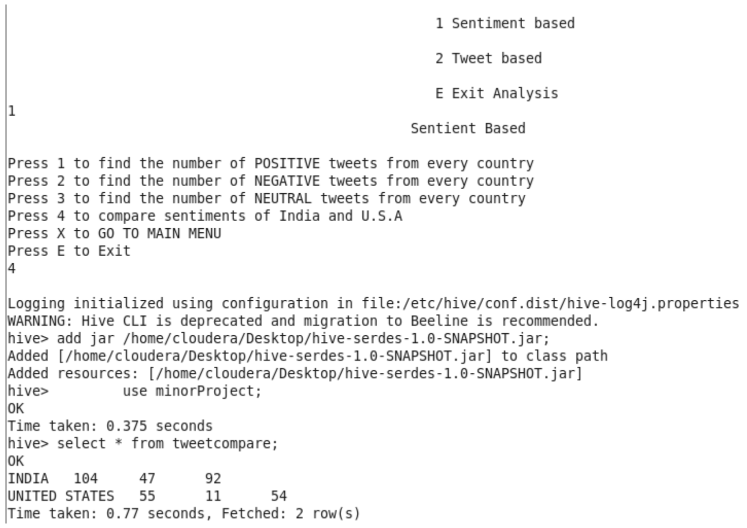
```
                                          Welcome to --- TWITTER SENTIMENT ANALYSIS

                                           Select the category for you analysis

                                                     1 Sentiment based

                                                     2 Tweet based

                                                     E Exit Analysis
1
                                                 Sentient Based

Press 1 to find the number of POSITIVE tweets from every country
Press 2 to find the number of NEGATIVE tweets from every country
Press 3 to find the number of NEUTRAL tweets from every country
Press 4 to compare sentiments of India and U.S.A
Press X to GO TO MAIN MENU
Press E to Exit
```

Sentiment_based.sh



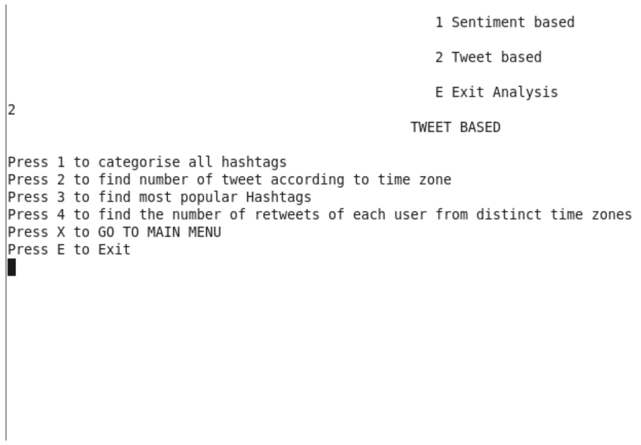
```
1
                                                 Sentient Based

Press 1 to find the number of POSITIVE tweets from every country
Press 2 to find the number of NEGATIVE tweets from every country
Press 3 to find the number of NEUTRAL tweets from every country
Press 4 to compare sentiments of India and U.S.A
Press X to GO TO MAIN MENU
Press E to Exit
1

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>     add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar]
hive> Display all 461 possibilities? (y or n)
hive> use minorProject;
OK
Time taken: 0.426 seconds
hive> select * from A;
OK
NULL    389
DENMARK 1
ECUADOR 1
HONG KONG       1
INDIA   104
MOROCCO 1
OMAN    2
SAUDI ARABIA    1
THAILAND        1
UNITED ARAB EMIRATES    1
UNITED STATES   55
Time taken: 0.903 seconds, Fetched: 11 row(s)
```

```
                                                    1 Sentiment based

                                                    2 Tweet based

                                                    E Exit Analysis
1
                                                 Sentient Based

Press 1 to find the number of POSITIVE tweets from every country
Press 2 to find the number of NEGATIVE tweets from every country
Press 3 to find the number of NEUTRAL tweets from every country
Press 4 to compare sentiments of India and U.S.A
Press X to GO TO MAIN MENU
Press E to Exit
2

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive>    add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar]
hive>         use minorProject;
OK
Time taken: 0.422 seconds
hive> select * from B;
OK
NULL    83
CANADA  1
ECUADOR 1
INDIA   47
KUWAIT  1
UNITED KINGDOM  1
UNITED STATES   11
Time taken: 0.933 seconds, Fetched: 7 row(s)

1
                                                 Sentient Based

Press 1 to find the number of POSITIVE tweets from every country
Press 2 to find the number of NEGATIVE tweets from every country
Press 3 to find the number of NEUTRAL tweets from every country
Press 4 to compare sentiments of India and U.S.A
Press X to GO TO MAIN MENU
Press E to Exit
3

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar]
hive>         use minorProject;
OK
Time taken: 0.62 seconds
hive>        select * from C;
OK
NULL    360
        1
GREECE  2
INDIA   92
IRAQ    1
SPAIN   1
SWEDEN  1
UKRAINE 2
UNITED ARAB EMIRATES    2
UNITED STATES   54
Time taken: 0.922 seconds, Fetched: 10 row(s)
```

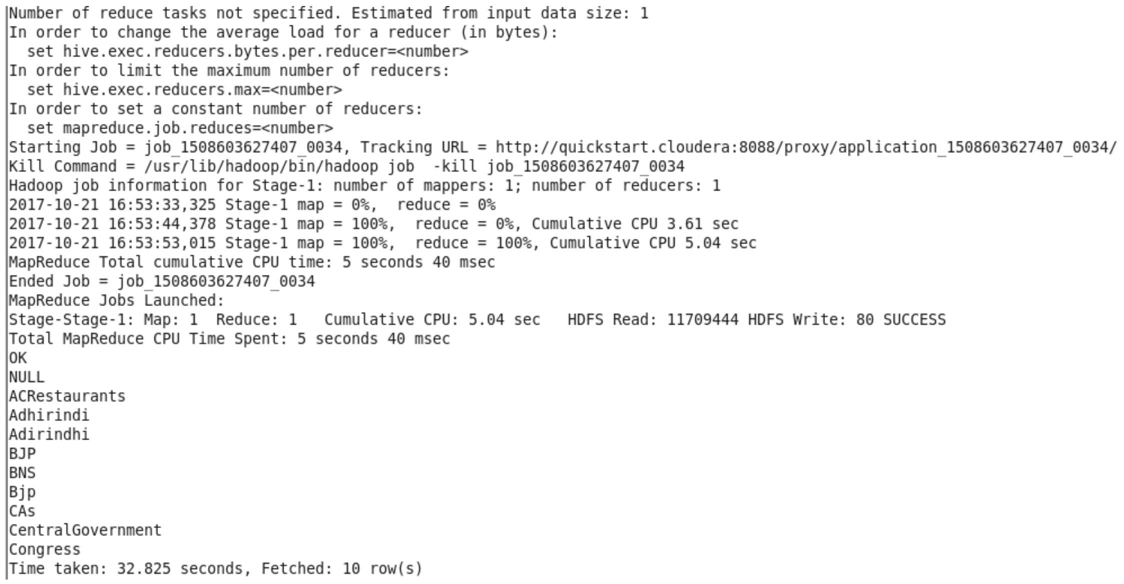


```
                                                    1 Sentiment based

                                                    2 Tweet based

                                                    E Exit Analysis
1
                                                 Sentient Based

Press 1 to find the number of POSITIVE tweets from every country
Press 2 to find the number of NEGATIVE tweets from every country
Press 3 to find the number of NEUTRAL tweets from every country
Press 4 to compare sentiments of India and U.S.A
Press X to GO TO MAIN MENU
Press E to Exit
4

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties
WARNING: Hive CLI is deprecated and migration to Beeline is recommended.
hive> add jar /home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar;
Added [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar] to class path
Added resources: [/home/cloudera/Desktop/hive-serdes-1.0-SNAPSHOT.jar]
hive>         use minorProject;
OK
Time taken: 0.375 seconds
hive> select * from tweetcompare;
OK
INDIA   104     47      92
UNITED STATES   55      11      54
Time taken: 0.77 seconds, Fetched: 2 row(s)
```

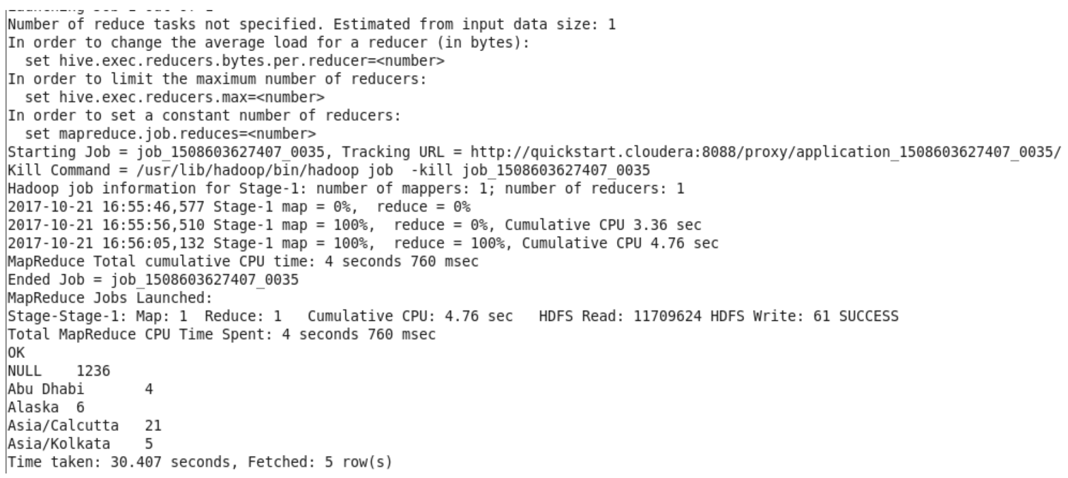
tweet_based.sh



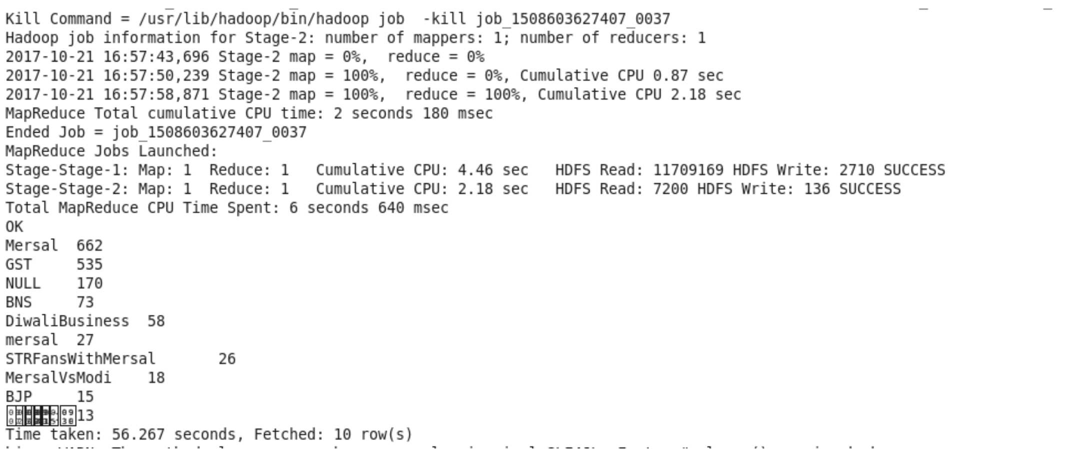
```
                                                    1 Sentiment based

                                                    2 Tweet based

                                                    E Exit Analysis
2
                                                 TWEET BASED

Press 1 to categorise all hashtags
Press 2 to find number of tweet according to time zone
Press 3 to find most popular Hashtags
Press 4 to find the number of retweets of each user from distinct time zones
Press X to GO TO MAIN MENU
Press E to Exit
```



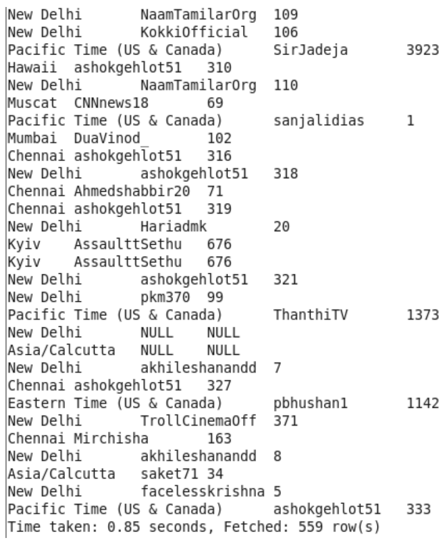
```
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1508603627407_0034, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1508603627407_0034/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1508603627407_0034
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-21 16:53:33,325 Stage-1 map = 0%,  reduce = 0%
2017-10-21 16:53:44,378 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.61 sec
2017-10-21 16:53:53,015 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.04 sec
MapReduce Total cumulative CPU time: 5 seconds 40 msec
Ended Job = job_1508603627407_0034
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.04 sec   HDFS Read: 11709444 HDFS Write: 80 SUCCESS
Total MapReduce CPU Time Spent: 5 seconds 40 msec
OK
NULL
ACRestaurants
Adhirindi
Adirindhi
BJP
BNS
Bjp
CAs
CentralGovernment
Congress
Time taken: 32.825 seconds, Fetched: 10 row(s)
```



```
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1508603627407_0035, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1508603627407_0035/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1508603627407_0035
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2017-10-21 16:55:46,577 Stage-1 map = 0%,  reduce = 0%
2017-10-21 16:55:56,510 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.36 sec
2017-10-21 16:56:05,132 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.76 sec
MapReduce Total cumulative CPU time: 4 seconds 760 msec
Ended Job = job_1508603627407_0035
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.76 sec   HDFS Read: 11709624 HDFS Write: 61 SUCCESS
Total MapReduce CPU Time Spent: 4 seconds 760 msec
OK
NULL    1236
Abu Dhabi       4
Alaska  6
Asia/Calcutta   21
Asia/Kolkata    5
Time taken: 30.407 seconds, Fetched: 5 row(s)
```



```
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1508603627407_0037
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2017-10-21 16:57:43,696 Stage-2 map = 0%,  reduce = 0%
2017-10-21 16:57:50,239 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.87 sec
2017-10-21 16:57:58,871 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.18 sec
MapReduce Total cumulative CPU time: 2 seconds 180 msec
Ended Job = job_1508603627407_0037
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 4.46 sec   HDFS Read: 11709169 HDFS Write: 2710 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 2.18 sec   HDFS Read: 7200 HDFS Write: 136 SUCCESS
Total MapReduce CPU Time Spent: 6 seconds 640 msec
OK
Mersal  662
GST     535
NULL    170
BNS     73
DiwaliBusiness  58
mersal  27
STRFansWithMersal       26
MersalVsModi    18
BJP     15
[illegible]13
Time taken: 56.267 seconds, Fetched: 10 row(s)
```

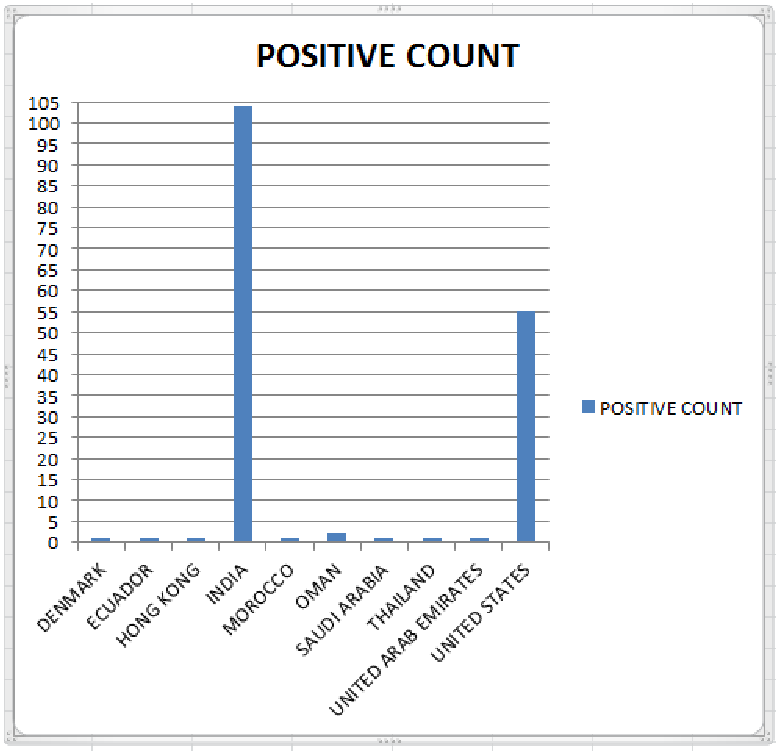


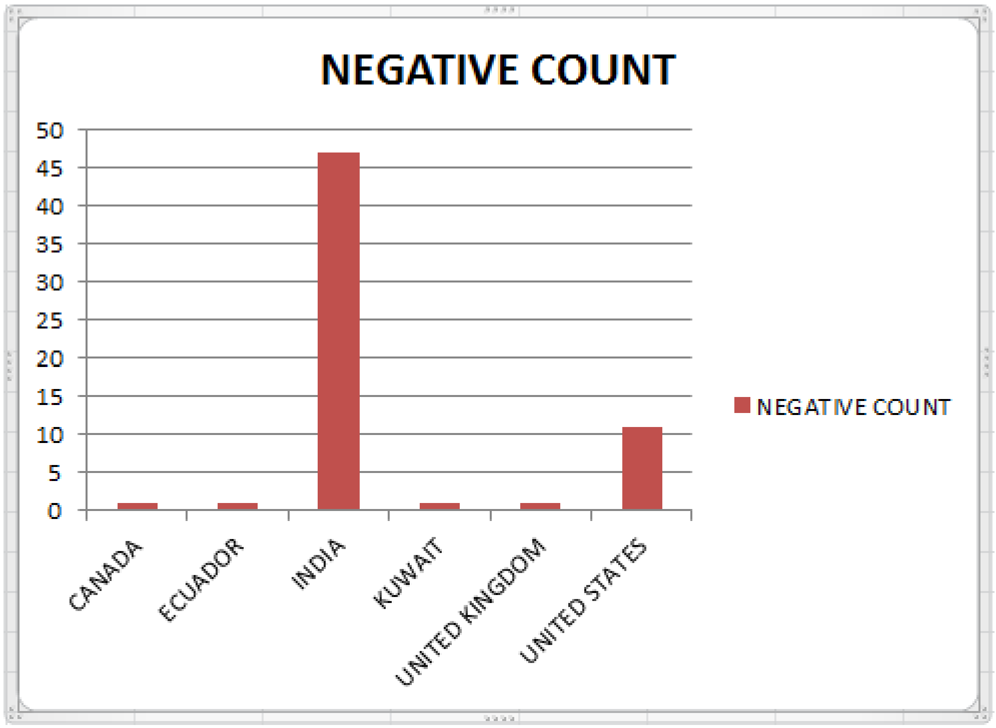
```
New Delhi       NaamTamilarOrg  109
New Delhi       KokkiOfficial   106
Pacific Time (US & Canada)      SirJadeja       3923
Hawaii  ashokgehlot51   310
New Delhi       NaamTamilarOrg  110
Muscat  CNNnews18       69
Pacific Time (US & Canada)      sanjalidias     1
Mumbai  DuaVinod_       102
Chennai ashokgehlot51   316
New Delhi       ashokgehlot51   318
Chennai Ahmedshabbir20  71
Chennai ashokgehlot51   319
New Delhi       Hariadmk        20
Kyiv    AssaulttSethu   676
Kyiv    AssaulttSethu   676
New Delhi       ashokgehlot51   321
New Delhi       pkm370  99
Pacific Time (US & Canada)      ThanthiTV       1373
New Delhi       NULL    NULL
Asia/Calcutta   NULL    NULL
New Delhi       akhileshanandd  7
Chennai ashokgehlot51   327
Eastern Time (US & Canada)      pbhushan1       1142
New Delhi       TrollCinemaOff  371
Chennai Mirchisha       163
New Delhi       akhileshanandd  8
Asia/Calcutta   saket71 34
New Delhi       facelesskrishna 5
Pacific Time (US & Canada)      ashokgehlot51   333
Time taken: 0.85 seconds, Fetched: 559 row(s)
```

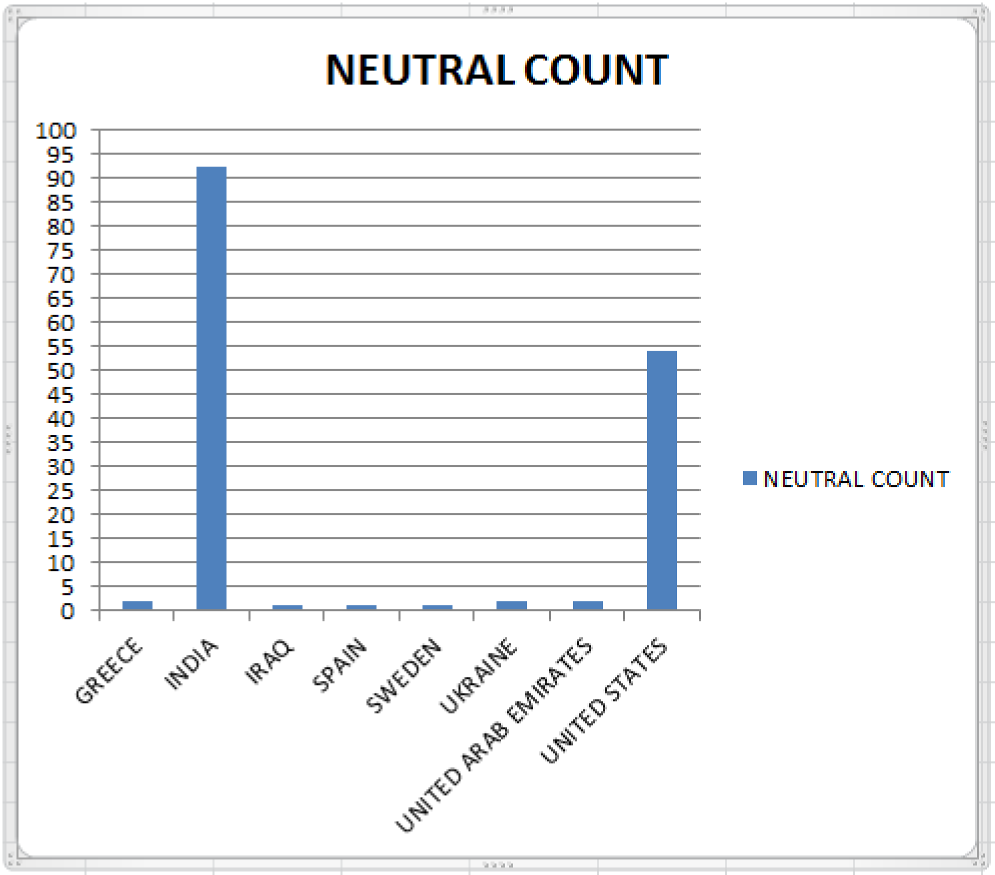
exit.sh

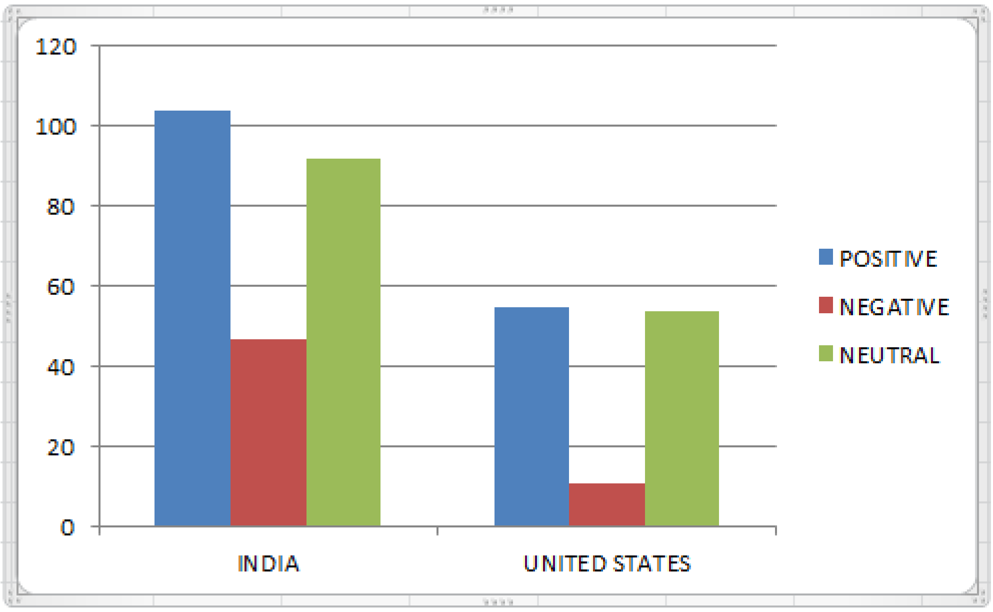
```
**********************************************************************************************
 _____ _   _    _    _   _ _  __ __   _____  _   _
|_   _| | | |  / \  | \ | | |/ / \ \ / / _ \| | | |
  | | | |_| | / _ \ |  \| | ' /   \ V / | | | | | |
  | | |  _  |/ ___ \| |\  | . \    | || |_| | |_| |
  |_| |_| |_/_/   \_\_| \_|_|\_\   |_| \___/ \___/
**********************************************************************************************
```

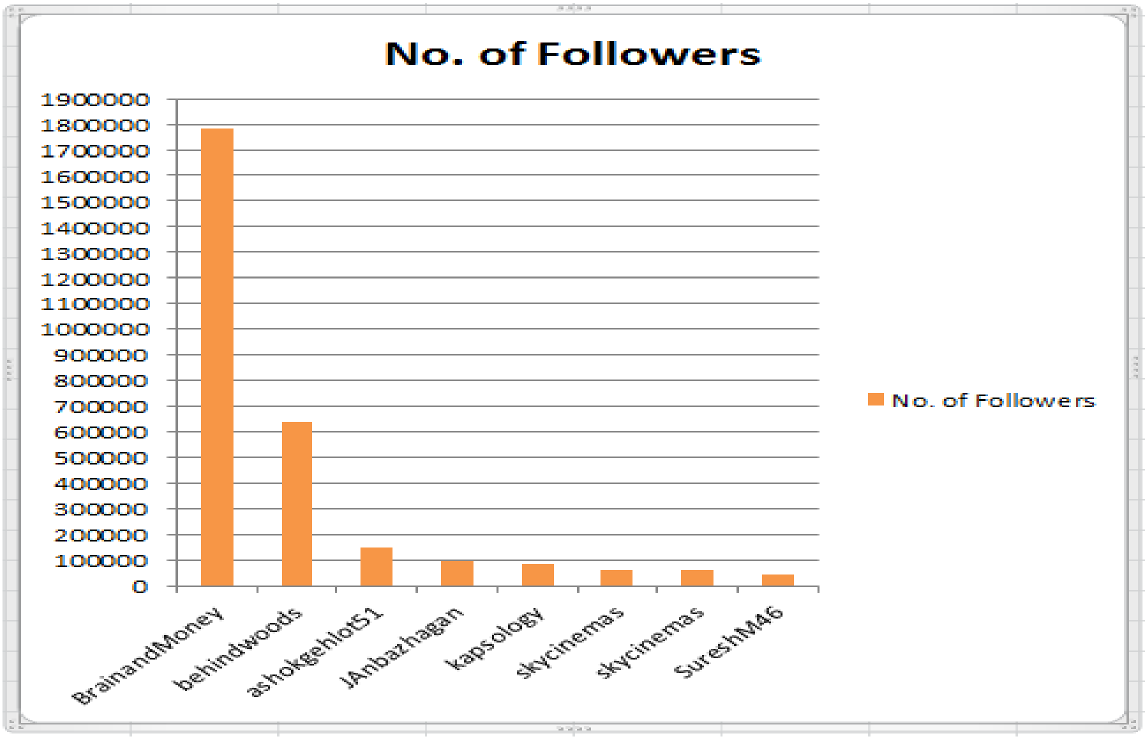
DATA VISUALISATION

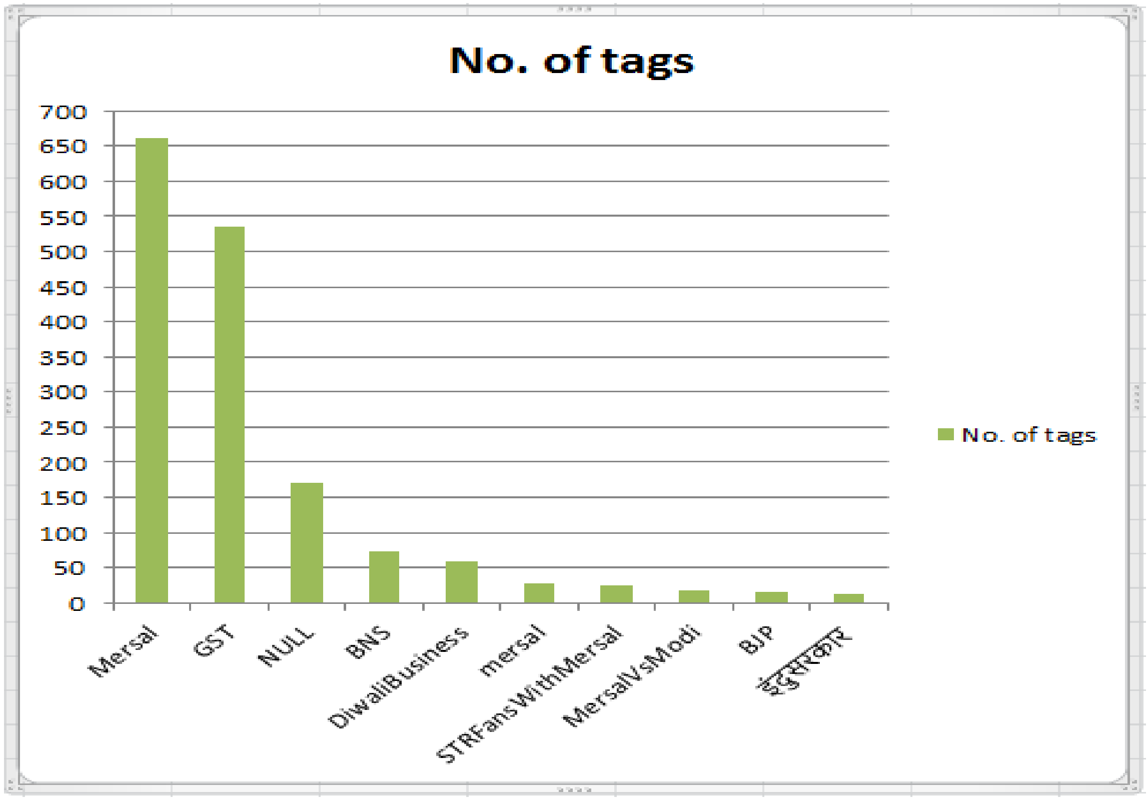


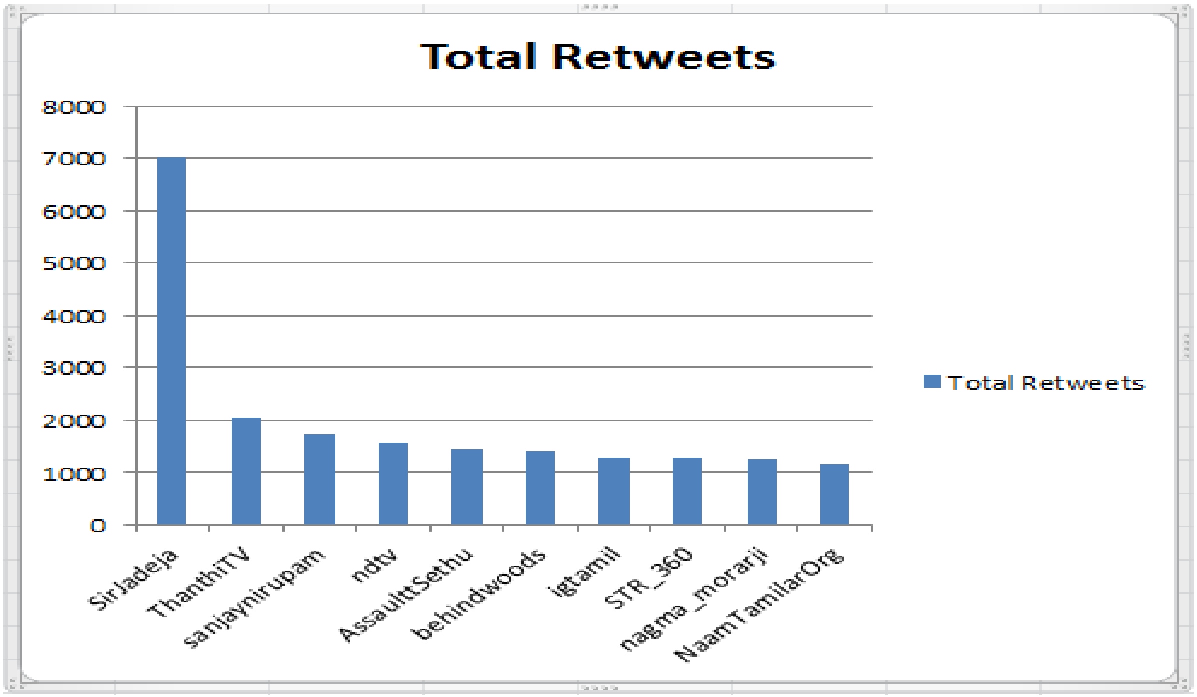


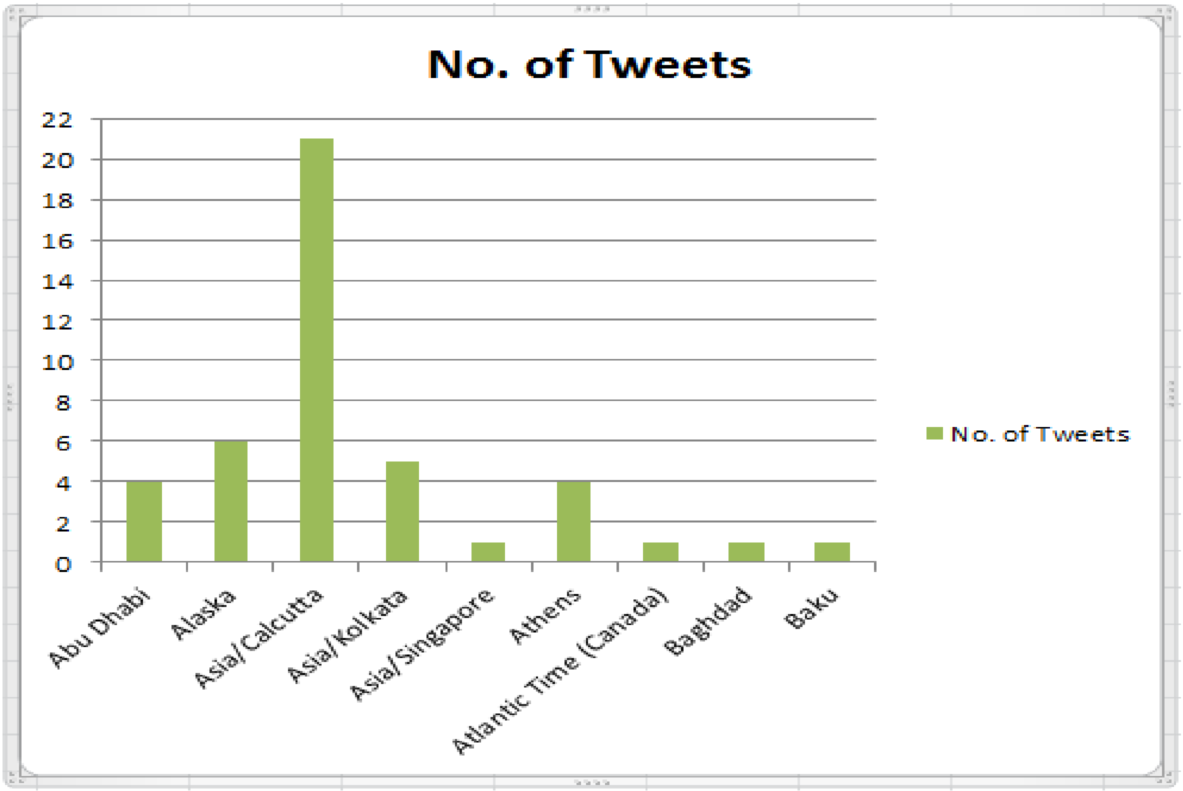

NEUTRAL COUNT
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
GREECE
INDIA
IRAQ
SPAIN
SWEDEN
UKRAINE
UNITED ARAB EMIRATES
UNITED STATES
NEUTRAL COUNT

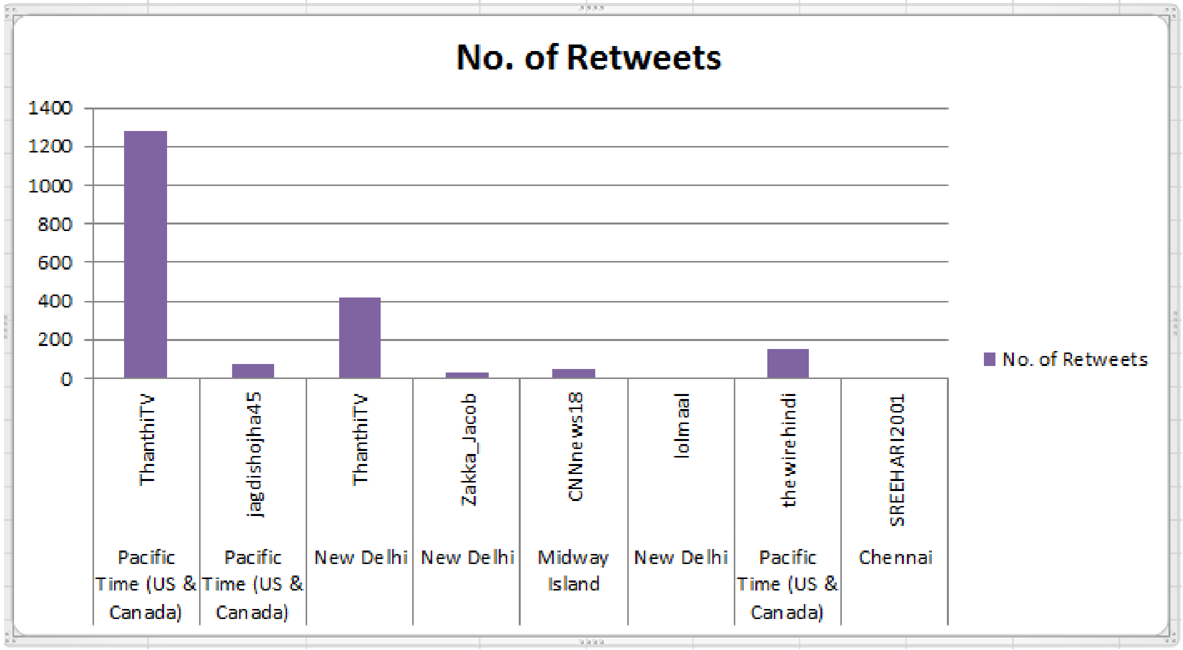

120
100
80
60
40
20
0
INDIA
UNITED STATES
POSITIVE
NEGATIVE
NEUTRAL


No. of Followers
1900000
1800000
1700000
1600000
1500000
1400000
1300000
1200000
1100000
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
BrainandMoney
behindwoods
ashokgehlot51
JAnbazhagan
kapsology
skycinemas
skycinemas
SureshM46
🟧 No. of Followers


No. of tags
700
650
600
550
500
450
400
350
300
250
200
150
100
50
0
Mersal
GST
NULL
BNS
DiwaliBusiness
mersal
STRFansWithMersal
MersalVsModi
BJP
इंदुसरकार
🟩 No. of tags


Total Retweets
8000
7000
6000
5000
4000
3000
2000
1000
0
SirJadeja
ThanthiTV
sanjaynirupam
ndtv
AssaultSethu
behindwoods
igtamil
STR_360
nagma_morarji
NaamTamilarOrg
Total Retweets


No. of Tweets
22
20
18
16
14
12
10
8
6
4
2
0
Abu Dhabi
Alaska
Asia/Calcutta
Asia/Kolkata
Asia/Singapore
Athens
Atlantic Time (Canada)
Baghdad
Baku
No. of Tweets


No. of Retweets
1400
1200
1000
800
600
400
200
0
ThanthiTV
jagdishojha45
ThanthiTV
Zakka_Jacob
CNNnews18
lolmaal
thewirehindi
SREEHARI2001
Pacific Time (US & Canada)
Pacific Time (US & Canada)
New Delhi
New Delhi
Midway Island
New Delhi
Pacific Time (US & Canada)
Chennai
No. of Retweets

REFERENCES

[1] Changbo Wang, Zhao Xiao, Yuhua Liu, Yanru Xu, Aoying Zhou, and Kang Zhang, ―SentiView: Sentiment Analysis and Visualization for Internet Popular Topics‖, IEEE Transactions On Human-Machine Systems, Vol. 43, No. 6, November 2013

[2] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow and Rebecca Passonneau, ―Sentiment Analysis of Twitter Data‖, Department of Computer Science, Columbia University (http://www.cs.columbia.edu/~apoorv/Homepage/Publications_files/emnlp2011_full.pdf)

[3] Jianshu Weng, Ee-Peng Lim, Jing Jiang, Qi He, ―TwitterRank: Finding Topic-sensitive Influential Twitterers‖, WSDM'10, February 4–6, 2010, New York City, New York, USA Copyright 2010 ACM