Text Data analysis on Twitter data.

**Technologies**: Hadoop-1.2.x, Flume-1.4, Hive-0.12.

We have two phases in this process.

**Phase** **1:** Downloading twitter data using Flume and store in HDFS

**Phase** **2:** Loading data to Hive and develop some adhoc queries to do analysis on text data.

**Phase 1:**

1. Create a developer account in <https://dev.twitter.com/user/login>and login with valid credentials.
2. Go to MyApplications and create a new app
3. Enter below details to create your application.

Name : MyApp

Description : This is my first application

Website : Give your website address(http://myapp.com)

Callback URL: http://myapp.com

Agree the Licence and click on create application.

1. Now go to API keys tab and create your access token by clicking on "Create my access token" to authorize your application created above.
2. Last step is to test OAuth and then we could see OAuth signature for your request, we need below parameters in flume configuration.

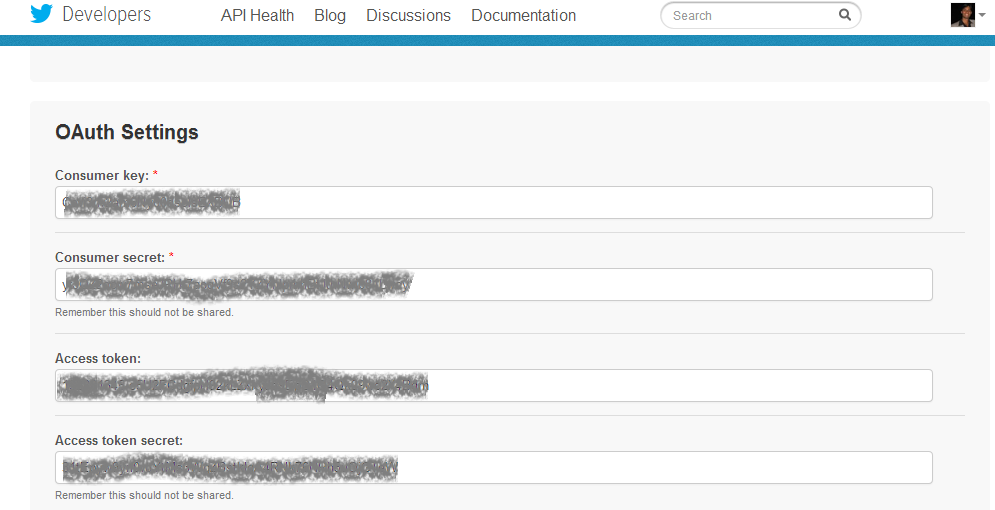
**Consumer key**: <some random generated key>

**Consumer secret**: <some random generated key>

**Access token**: <some random generated key>

**Access token secret**:<some random generated key>

Keep noted these values for future configuration in flume.



1. Assuming Hadoop is already installed and configured. Next step is to download and extract FLUME to installation directory.
2. Download the [flume-sources-1.0-SNAPSHOT.jar](http://files.cloudera.com/samples/flume-sources-1.0-SNAPSHOT.jar) (http://files.cloudera.com/samples/flume-sources-1.0-SNAPSHOT.jar) and add it to the flume class path or add the jar file to flume/lib

This jar contains the java classes to pull the Tweets and save them into HDFS.

1. The $FLUME\_HOME/conf/flume.conf should have all the agents (flume, memory and hdfs) defined as below
2. The **consumerKey**, **consumerSecret**, **accessToken** and **accessTokenSecret** have to be replaced with those obtained from above.

And, **TwitterAgent.sinks.HDFS.hdfs.path** should point to the NameNode and the location in HDFS where the tweets will go to.

The **TwitterAgent.sources.Twitter.keywords** value can be modified to get the tweets for some other topic like football, movies,politics etc.

Add below lines to flume.conf file

$nano $FLUME\_HOME/conf/flume.conf

TwitterAgent.sources = Twitter

TwitterAgent.channels = MemChannel

TwitterAgent.sinks = HDFS

TwitterAgent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource

TwitterAgent.sources.Twitter.channels = MemChannel

TwitterAgent.sources.Twitter.consumerKey = <consumerKey generated in step 5 >

TwitterAgent.sources.Twitter.consumerSecret = <consumerKey generated in step 5>

TwitterAgent.sources.Twitter.accessToken = <consumerKey generated in step 5>

TwitterAgent.sources.Twitter.accessTokenSecret = <consumerKey generated in step 5>

TwitterAgent.sources.Twitter.keywords = hadoop, big data, analytics, bigdata, cloudera, data science, data scientiest, business intelligence, mapreduce, data warehouse, data warehousing, mahout, hbase, nosql, newsql, businessintelligence, cloudcomputing

TwitterAgent.sinks.HDFS.channel = MemChannel

TwitterAgent.sinks.HDFS.type = hdfs

TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/user/flume/tweets/

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.channels.MemChannel.type = memory

TwitterAgent.channels.MemChannel.capacity = 10000

TwitterAgent.channels.MemChannel.transactionCapacity = 100

1. Now Start flume agent using below command

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

1. After a couple of minutes the Tweets should appear in HDFS.

**Phase 2:**

1. Twitter returns Tweets in the JSON format. Now we develop schema for twitter data and load to hive to perform text data analysis.
2. Assuming Hive is installed and configured.
3. Download [**hive-serdes-1.0-SNAPSHOT.jar**](http://files.cloudera.com/samples/hive-serdes-1.0-SNAPSHOT.jar) to the lib directory in Hive and this library will help Hive understand the JSON format.
4. Create a hive external table for tweets.

$ nano twitterdata.q

USE gsk;

ADD JAR /usr/local/hadoop/hive-0.12.0-bin/lib/json-serde-1.1.9.3-SNAPSHOT-jar-with-dependencies.jar;

DROP TABLE IF EXISTS TWITTERDATA;

CREATE EXTERNAL TABLE IF NOT EXISTS TWITTERDATA (

filter\_level string,

retweeted\_status struct<

contributors: string,

text: string,

geo: string,

retweeted: string,

in\_reply\_to\_screen\_name: string,

truncated: string,

lang: string,

entities:struct<

symbols: array<string>,

urls:array<string>,

hashtags:array<string>,

user\_mentions:array<string>

>,

in\_reply\_to\_status\_id\_str:string,

id:string,

source:string,

in\_reply\_to\_user\_id\_str: string,

favorited: string,

in\_reply\_to\_status\_id: string,

retweet\_count: int,

created\_at: string,

in\_reply\_to\_user\_id: string,

favorite\_count: string,

id\_str: string,

place: string,

user: struct<

location: string,

default\_profile: string,

profile\_background\_tile: string,

statuses\_count: int,

lang: string,

profile\_link\_color: string,

id: string,

followings: string,

favourites\_count: int,

protected: string,

profile\_text\_color: string,

description: string,

verified: string,

contributors\_enabled: string,

profile\_sidebar\_border\_color: string,

name: string,

profile\_background\_color: string,

created\_at: string,

is\_translation\_enabled: string,

default\_profile\_image: string,

followers\_count: int,

profile\_image\_url\_https: string,

geo\_enabled: string,

profile\_background\_image\_url: string,

profile\_background\_image\_url\_https: string,

follow\_request\_sent: string,

url: string,

utc\_offset: int,

time\_zone: string,

notifications: string,

profile\_use\_background\_image: string,

friends\_count: string,

profile\_sidebar\_fill\_color: string,

screen\_name: string,

id\_str: string,

profile\_image\_url: string,

listed\_count: int,

is\_translator: string

>,

coordinates: string

>,

contributors string,

text string,

geo string,

retweeted string,

in\_reply\_to\_screen\_name string,

truncated string,

lang string,

entities struct<

symbols:array<string>,

urls:array<string>,

hashtags:array<string>,

user\_mentions:array<string>

>,

in\_reply\_to\_status\_id\_str string,

id string,

source string,

in\_reply\_to\_user\_id\_str string,

favorited string,

in\_reply\_to\_status\_id string,

retweet\_count int,

created\_at string,

in\_reply\_to\_user\_id string,

favorite\_count int,

id\_str string,

place string,

user struct<

location: string,

default\_profile: string,

profile\_background\_tile: string,

statuses\_count: int,

lang: string,

profile\_link\_color: string,

id: string,

followings: string,

favourites\_count: int,

protected: string,

profile\_text\_color: string,

description: string,

verified: string,

contributors\_enabled: string,

profile\_sidebar\_border\_color: string,

name: string,

profile\_background\_color: string,

created\_at: string,

is\_translation\_enabled: string,

default\_profile\_image: string,

followers\_count: int,

profile\_image\_url\_https: string,

geo\_enabled: string,

profile\_background\_image\_url: string,

profile\_background\_image\_url\_https: string,

follow\_request\_sent: string,

url: string,

utc\_offset: int,

time\_zone: string,

notifications: string,

profile\_use\_background\_image: string,

friends\_count: string,

profile\_sidebar\_fill\_color: string,

screen\_name: string,

id\_str: string,

profile\_image\_url: string,

listed\_count: int,

is\_translator: string

>,

coordinates string

)

ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'

LOCATION '/user/flume/tweets/';

Now that we have the data in HDFS and the table created in Hive

1. We start with simple queries to check if data is loaded correctly.

hive>select count(\*) from twiiterdata ;

// Sample tweets

hive> SELECT text from twitterdata limit 5

1. Below are few queries on twitter data

//check out the tweets that are RTs.

SELECT retweeted\_status.user.screen\_name, COUNT(\*) as total

FROM twitterdata

WHERE retweeted\_status.user is not null

GROUP BY retweeted\_status.user.screen\_name

ORDER BY total desc

LIMIT 20;

//To know which user has the most number of followers

select user.screen\_name, user.followers\_count c from twitterdata order by c desc;

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

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(retweet\_count) as retweets FROM twitterdata GROUP BY retweeted\_status.user.screen\_name, retweeted\_status.text) t GROUP BY t.retweeted\_screen\_name ORDER BY total\_retweets DESC LIMIT 10;

1. Advanced queries

// n-gram analysis:

//In twitter data, selecting 50 most frequently 3-word pairs called tri-gram (3-word triplets). We also try di-gram, four-gram.

SELECT ngrams(sentences(lower(text)), 3, 50) FROM twitterdata;

//di-gram

SELECT ngrams(sentences(lower(text)), 2, 50) FROM twitterdata;

// explode function will take input array/map of elements outputs array elements as individual records

SELECT explode(ngrams(sentences(lower(text)), 2, 50)) as ngs FROM twitterdata;

//Select the 100 most common words that appear after "I need"

SELECT explode(context\_ngrams(sentences(lower(text)), array("i", "need", null), 100))

AS ngs FROM twitterdata;