**Performing sentiment analysis on the tweets**

The tweets of trending topics stored in HDFS are used for sentiment analysis and processed using HiveQL. The steps are discussed in the following section.

1. *Removal of stopwords and cleaning*

Removal of Stopwords and cleaning is used to remove unwanted or useless data in relation to our analysis process. There are Hive queries to remove the stopwords and also clean the data.

hive> CREATE VIEW tweets\_simple AS SELECT

id,

`user`.screen\_name, source, retweeted\_status.retweet\_coun t, entities.hashtags,

cast ( from\_unixtime( unix\_timestamp(concat( '2016 ', substring(created\_at,5,15)), 'yyyy MMM dd hh:mm:ss')) as timestamp) ts,

text,

`user`.statuses\_count,

`user`.friends\_count,

`user`.followers\_count,

`user`.time\_zone

FROM raw\_f;

1. *Tokenizing the tweets*

In order to find the sentiment words, the tweet is split into words using one of the Hive UDF functions. A Hive table is created to store the tweet id and the array of words present in each tweet. As multiple words are present in an array, we used some built in UDTF function to extract each word from an array and created a new row for each left outer join operation on a table that multiple words are present in an array, used some built in UDTF function to extract each word from an array and created a new row for each left outer join operation on a table that contains id, word and dictionary table if the word matches with the sentiment word in the dictionary.

**Queries used for this purpose is:**

hive> CREATE VIEW L1 AS SELECT

id,

words

FROM raw\_f LATERAL VIEW

EXPLODE(sentences(lower(text))) dummy

AS words;

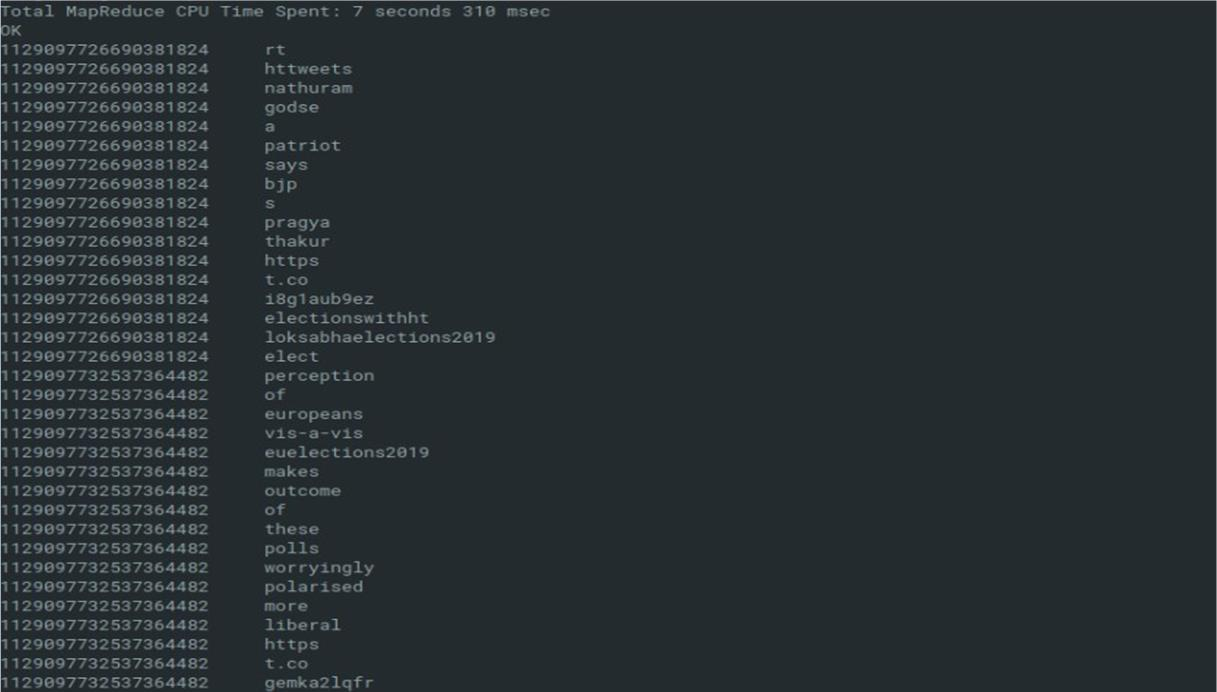
hive> CREATE VIEW L2 AS SELECT

id,

word

FROM L1 LATERA VIEW

EXPLODE(words) dummy AS word;



*Fig: Tokenization of each word in tweets.*

1. *Sentiment word detection*

Sentiment analysis is done using dictionary-based method. A table is created to store the contents present in the dictionary. In order to rate the tokenized words, the tokenized words have to be mapped with the loaded dictionary. We performed left outer join operation on a table that contains id, word and dictionary table if the word matches with the sentiment word in the dictionary, then a rating is given to the matched word or else NULL value is assigned. A hive table is created to store id, word and then rating.

## **Creating external table for dictionary**

hive> CREATE EXTERNAL TABLE dictionary (

type STRING,

length INT,

word STRING,

pos STRING,

polarity STRING

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' STORED AS TEXTFILE;

**Load dictionary into HDFS**

hadoop dfs -put /home/mayank/dictionary.csv hdfs://localhost”9000/

LOAD DATA INPATH ‘/dictionary.csv’ INTO TABLE dictionary

**Converting polarity into numeric value**

hive> CREATE VIEW L3 AS SELECT

id,

L2.word,

CASE d.polarity

WHEN ‘negative’ THEN 0

WHEN ‘positive’ THEN 5

ELSE 2.5 END AS polarity

FROM L2 LEFT OUTER JOIN dictionary d on L2.word = d.word;

**Classification of words into positive, negative or neutral**

hive> CREATE TABLE tweets\_sentiment AS SELECT

id,

CASE

WHEN AVG (polarity) > 2.5 THEN ‘positive’

WHEN AVG (polarity) < 2.5 THEN ‘negative’

ELSE ‘neutral’ END AS sentiment

FROM L3 GROUP BY id;

**Classification of tweets**

After performing all the above steps, we have id, word and rating. Then “group by” operation is performed on id to group all the words belonging to one tweet after which average operation is performed on the ratings given to each word in a tweet. Then “group by” operation is performed on id to group all the words belonging to one tweet after which average operation is performed on the ratings given to each word in a tweet. Based on the average ratings, tweets are classified into positive and negative.

hive> CREATE VIEW L8 AS SELECT

sentiment,

COUNT (id)

FROM tweets\_sentiment AS ts

GROUP BY sentiment;



*Fig: Tweets classification*