Sentiment Analysis on Demonetization

Let us find out the views of different people on the demonetization by analysing the tweets from twitter. Here is the dataset where twitter tweets are gathered in CSV format. We will find the positive and negative sentiment on Demonetization based on the Tweets.

MetaData

Id
Text (Tweets)
favorited
favoriteCount
replyToSN
created
truncated
replyToSID
id
replyToUID
statusSource
screenname
retweetCount
isRetweet
retweeted

1) Load the Demonetization tweets file.

```
grunt> load_tweets_fl = LOAD
'/home/cloudera/khasimbabu/PIG_Excercise/CaseStudy1/demonitization_tweets.csv.txt' USING
PigStorage(',');
```

2) Extract ID,TweetText column values from the file.

```
grunt> extract_details = FOREACH load_tweets_fl generate $0 as Id, $1 as TweetText;
```

```
grunt> describe extract_details;
extract_details: {Id: bytearray,TweetText: bytearray}
```

3) Split the TweetText into individual words. tweet_text into words to calculate the sentiment of the whole tweet.

```
grunt> tokenize_text = FOREACH extract_details GENERATE Id,TweetText,
FLATTEN(TOKENIZE(TweetText)) as word;
```

```
grunt> describe tokenize_text;
tokenize_text: {Id: bytearray,TweetText: bytearray,word: chararray}
```

4) Load Dictionary dataset for positive and negative words.

The AFINN is a dictionary which consists of 2500 words which are rated from +5 to -5 depending on their meaning.

```
grunt> load_dictionary = LOAD '/home/cloudera/khasimbabu/PIG_Excercise/CaseStudy1/AFINN.txt'
USING PigStorage('\t') as (word:chararray, rating:int);
```

5) Join the Tweets dataset and Dictionary dataset for differentiate possitive and negative words.

```
grunt> join_tweets_dictionary = JOIN tokenize_text by word LEFT OUTER, load_dictionary by word
USING 'replicated';
```

-> replicated is used for verifying all the tweet words with the dictionary words.

```
grunt> describe join_tweets_dictionary;
join_tweets_dictionary: {tokenize_text::Id: bytearray,tokenize_text::TweetText:
bytearray,tokenize_text::word: chararray,load_dictionary::word: chararray,load_dictionary::rating:
int}
```

6) Get the rating for each word.

```
grunt> rating = FOREACH join_tweets_dictionary GENERATE tokenize_text::Id as
id,tokenize_text::TweetText as text,load_dictionary::rating as rate;
```

7) Group the ID,Text for rating & Get the Average rating for each tweet.

```
grunt> grp_word = GROUP rating by (id,text);
```

```
grunt> average_rate = FOREACH grp_word GENERATE group, AVG(rating.rate) as tweet_rating;
```

8) Classify Positive and Negative tweets.

```
grunt> positive_tweet = FILTER average_rate by tweet_rating>=0;
```

```
grunt> negative_tweet = FILTER average_rate by tweet_rating<0;
```