Authors: Oguzhan Ugur, Amir Zamli, Shadman Ahmed

Live Twitter Sentiment Analysis

Introduction

Twitter is a platform where members create and interact with messages called tweets and serves as a platform where individuals express their thoughts through tweets. Tweets are therefore of a great interest to gather information and insight about products, politics, markets etc. This makes the data in twitter valuable since it can reflect the thoughts of society in some part and is therefore used by various companies to conduct analysis.

The type of analysis conducted for online purposes is mostly prediction of people's impressions or feelings, in regard to comments and topic. Grading if the text has positive, negative or natural sentiment. This is one of the applications of sentiment analysis, which uses natural language processing to extract the sentiment content from different kind of texts. Twitter offers huge amounts of data since it approximately publishes 6000 tweets per second [1]. Data-intensive computing makes it possible to handle/scale real time big data and conduct sentiment analysis on the streamed data.

In this project, we will fetch online tweets using the Twitter API with the help of Kafka. Then process the data via a data processing pipeline called Apache Spark. Where we will take the stream of data as small batches and output an average value of a sentiment score of all the tweets contained in the batches and the tweet creation time. The output will be saved and stored in a table in Cassandra where we will extract the average sentiment score and plot it, to see for a certain topic, how the sentiment changes over time.

Methodology and Implementation

Twitter API and Kafka (Dstream)

Twitter exposes its data via an API (Application Programming Interface). This API streams tweet messages in near real-time over a persistent connection. The connection to the API is established by Kafka. Kafka is a data processing model that enables scalable, fault-tolerant stream processing of real time data. The data received from Twitter is provided as a json file with key/value pairs that contains raw tweets. The data contains lots of unnecessary information, we decided to strip of all data except the creation time, and the tweet text itself.

Pre-processing

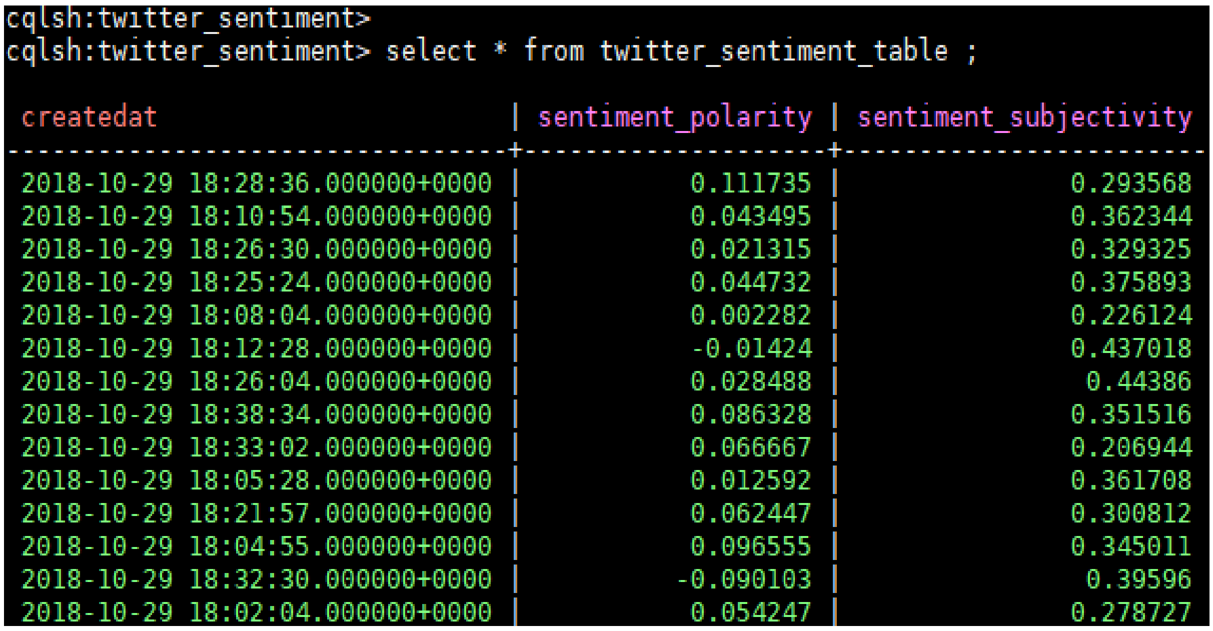
Raw tweets streamed from Twitter usually result in a noisy dataset, due to the nature of people's usage of social medias e.g symbols, URL links, emoticons, usernames etc. Therefore, streamed raw twitter data has to be sanitized and processed into a desired format in order to create data that is analyzable.
The data received from twitter was in json format, where it was tupled into a 2-tuple of time and tweet-text data. The tweets were sanitized from unnecessary information, such as, symbols, URL links, emoticons, usernames etc. These streamed tweets were then analyzed by using a sentiment analysis method.

Sentiment Analysis

In order to do sentiment analysis of the streamed tweets, a library named TextBlob was used. TextBlob is a Python library for processing textual data and it consists of natural language processing tools that makes it possible to utilize sentiment analysis, classification, translation etc. The sentiment analysis returns polarity and subjectivity scores. The polarity scores are digits between -1 and 1, where -1 stands for negative, 0 neutral and 1 positive polarity. The subjectivity scores are digits between 0 and 1, where 0 stands for objectivity and 1 subjectivity. The sentiment analysis of the tweets were conducted over time, the obtained scores were averaged over seconds in order to reduce the data and make it possible to visualize over a long time period. The results obtained by this analysis were stored periodically in a Cassandra table.

Persistent Storage (Cassandra)

Cassandra is a high-performance distributed database that is known to be highly scalable and fault tolerant [2]. This database was used in order to store and update the the averaged scores over time. The result was stored in a table with the columns: time, subjectivity score and polarity score. See table depicted on Fig 1.



```
cqlsh:twitter_sentiment>
cqlsh:twitter_sentiment> select * from twitter_sentiment_table ;

 createdat                       | sentiment_polarity | sentiment_subjectivity
---------------------------------+--------------------+------------------------
 2018-10-29 18:28:36.000000+0000 |           0.111735 |               0.293568
 2018-10-29 18:10:54.000000+0000 |           0.043495 |               0.362344
 2018-10-29 18:26:30.000000+0000 |           0.021315 |               0.329325
 2018-10-29 18:25:24.000000+0000 |           0.044732 |               0.375893
 2018-10-29 18:08:04.000000+0000 |           0.002282 |               0.226124
 2018-10-29 18:12:28.000000+0000 |           -0.01424 |               0.437018
 2018-10-29 18:26:04.000000+0000 |           0.028488 |                0.44386
 2018-10-29 18:38:34.000000+0000 |           0.086328 |               0.351516
 2018-10-29 18:33:02.000000+0000 |           0.066667 |               0.206944
 2018-10-29 18:05:28.000000+0000 |           0.012592 |               0.361708
 2018-10-29 18:21:57.000000+0000 |           0.062447 |               0.300812
 2018-10-29 18:04:55.000000+0000 |           0.096555 |               0.345011
 2018-10-29 18:32:30.000000+0000 |          -0.090103 |                0.39596
 2018-10-29 18:02:04.000000+0000 |           0.054247 |               0.278727
```

Figure 1, Table in Cassandra consisting sentiment score for each tweet

Results

In order to get the plots depicted in Fig 2, the data had to be pre-processed before using bokeh visualization library. The data was processed so that negative, positive and neutral sentiment averages was calculated over larger batches of time, but separately so that it was possible to achieve the desired visualization of sentiment shown below in figure 2.

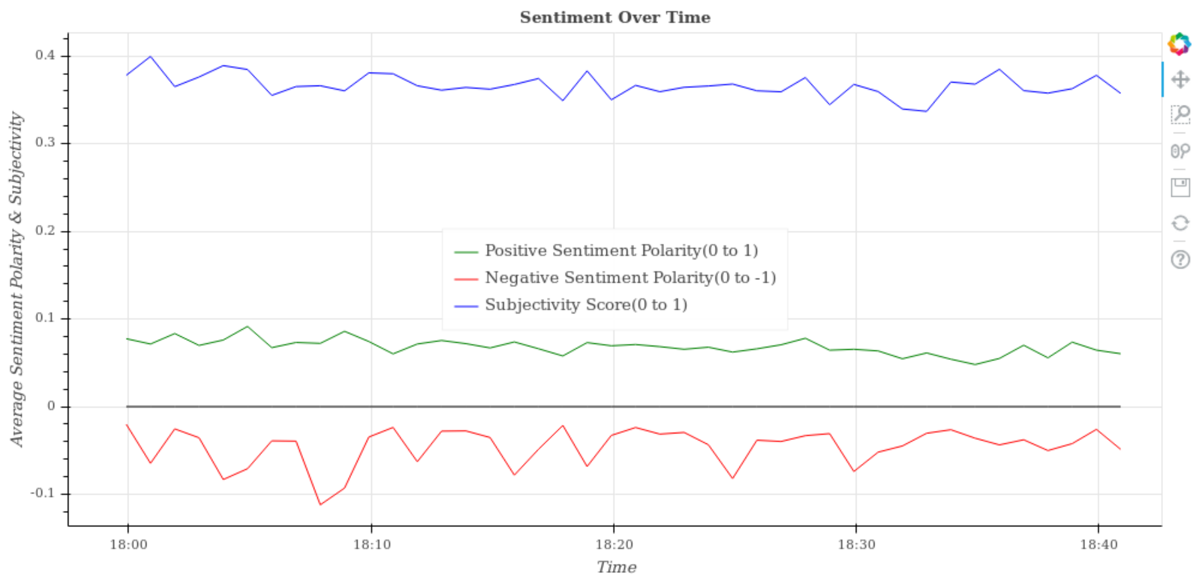


Figure 2, The Trump topic in Twitter gave a polarity score between -0.1 and 0.1 for negativity and positivity respectively.

Conclusion

Working with Spark and Kafka turned out well, we found out that it was easy to expand new features and process the data in a functional manner, this allows us to expand our program to support multiple languages given that we have sentiment analysis for each supported language and further scalability in the case of handling multiple topics and streaming sources, having these capabilities can be very important.

Sentiment analysis can be a good way to find out the sentiment of a specific topic. We chose to do sentiment analysis by using Python and Spark. Spark gave us a lot of opportunities to process incoming tweets and analyze them in a distributed and fault tolerant manner. The problem we encountered were to integrate Cassandra and Spark via Python, we solved this using a work around that can cost us unnecessary overhead time on each saving to the Cassandra table. This problem could be resolved by either making it work properly in python or switching to Scala, where the integration is easier. On the other hand, Cassandra worked well and provided us with scalability and fault tolerance for storage of real time data. The sentiment analysis tool was quite effective and easy to use. The streaming process from Twitter worked well, the only problem was that a non-trending topic didn’t give enough data to analyze. This problem could be solved by having access to historic data.

References

[1] http://www.internetlivestats.com/twitter-statistics/, last updated: 1/11-18.
[2] https://www.tutorialspoint.com/cassandra/cassandra_introduction.htm, last updated: 28/10-18