TWITTER ANALYSIS FINAL PROJECT

CIS 490: Big Data Analytics

Abstract

The goal of the experiment was to analyze the data obtained from twitter. There were 974 million existing Twitter accounts in 2014(5). Every second, on average, around 6,000 tweets are tweeted on Twitter (visualize them here), which corresponds to over 350,000 tweets sent per minute, 500 million tweets per day and around 200 billion tweets per year. In this project I attempted to analyze tweets using Naïve Bayes Classifier in the first part with python. In the second part, I attempted to calculate the sentiments of a substantially larger dataset using hive on a Microsoft HDInsight Windows Hadoop server and connected to that server with ODBC Driver from Excel to analyze the data using Power Pivot.

Vladislav Dubrovenski
vladi@ksu.edu

Twitter Analysis Using Various Tool

Introduction

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader). (1)

The initial goal of my experiment was to obtain a large amount of data containing the words BMW, Mercedes, and Audi and then sentimentally analyze it using NLTK tool kit and its NaiveBayesClassifier. I was able to train this classifier with tweets examples provided by NLTK. However, during my experiment I figured out that the method I was using is not very suitable for a large amount of data. Therefore, I decided to use Hive and AFINN sentiment (including both words and emoticons) dictionary found on one GitHub repository.

The Dataset

For the first part of my experiment, I obtain the data directly from Twitter using its API by 3 keywords: Mercedes, BMW, and Audi.

For the second part of my experiment, I used a part of a large datasets that I found at the following URL: https://datahub.io/dataset/twitter-2012-presidential-election. This dataset contains over 170,000,000 tweets that were recorded prior 2012 elections with many various keywords on major political controversies. Unfortunately, that was the only data I was able to obtain free. The start volume of my data is roughly 350GB, which I think is enough to demonstrate the power of Hive, Hadoop and MapReduce. After manipulation with data and removing the tweets without the keywords Obama and Romney and sentiment classification I was left only with roughly 40,000,000 tweets that I upload in excel using ODBC hive driver and Microsoft Power Query.

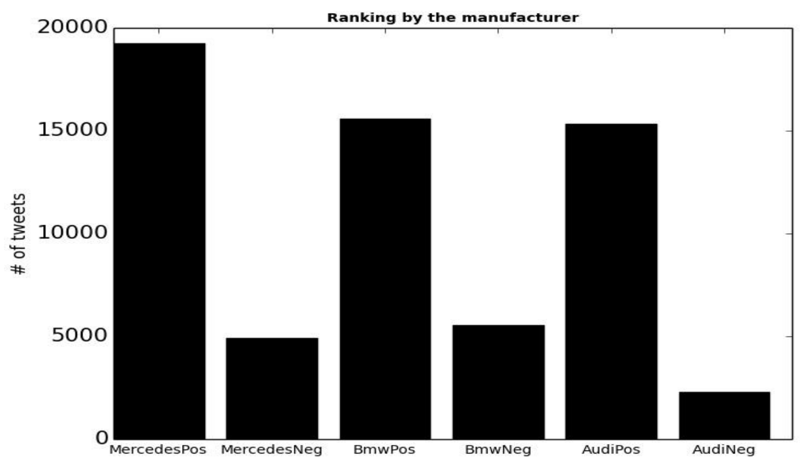
Problem Statement

My main goal for the first part was to be able to distinguish between positive and negative tweets, determine which carmaker has the most positive or negative tweets and compare their popularity. The main goal for the second part of the project was to sentimentally analyze a large dataset of tweets. Since the only large dataset that was obtained was on the topic of presidential elections, my goal became to filter the tweets dataset, distinguish them between negative, neutral, and positive, and try to understand the political views of users on various topics with respect to their date.

The Method (Part 1)

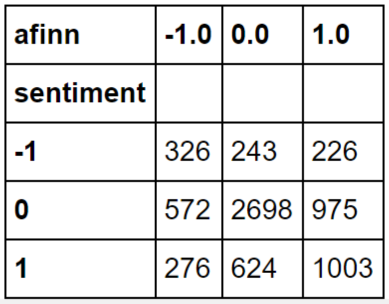
The first step was to create the sentiment dictionaries. I obtained positive and negative tweets as well as English stop words from NLTK Corpus and used them to create 2 separate dictionaries and then combine them into one. After that, I extracted the features from these tweets and used them to train the Naive Bayes classifier that is provided from one of the NLTK libraries. So, what is Nave Bayes classifier and how does it work? I think Sunil Ra gives the best explanation: "It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods."

Another aspect of this part of the project was processing (or cleaning) the input data. Using regex I removed www, http, @, and the hashtags. Then I classified each tweet from the data obtained from twitter as positive or negative with classifier that was trained with tweets from NLTK corpus, and placed these classified tweets into the data frames. Then using "Matplotlib" I received the following graph.

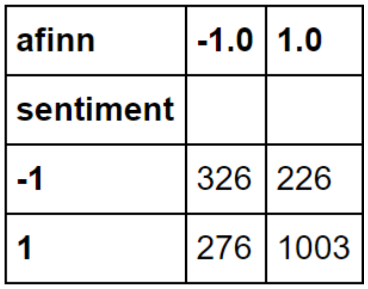


<u>The Method</u> (Part 2)

For the second part of my experiment I decided to use Hive in order to be able to analyze a large amount of tweets. I used AFINN dictionary that contained words and emoji with a polarity score assigned on a scale from -5 to +5. This dictionary was obtained from the following GitHub repository https://github.com/fnielsen/afinn/tree/master/afinn/data and has the following confusion matrices. (2)



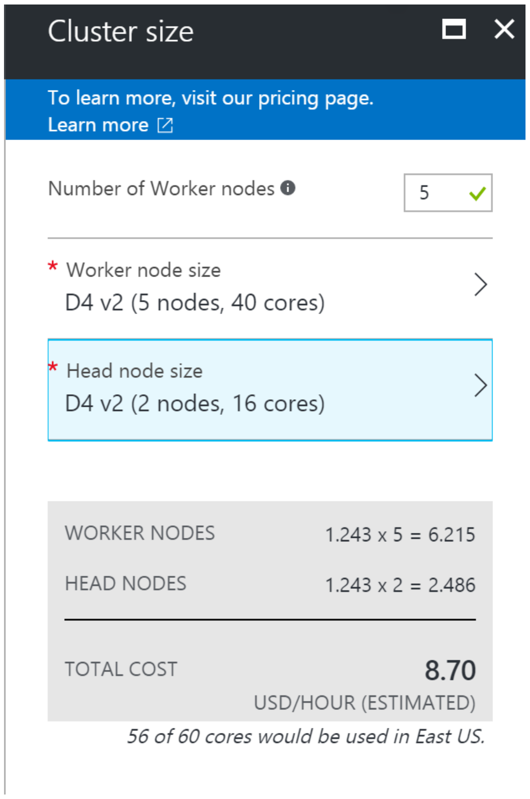
afinn	-1.0	0.0	1.0
sentiment			
-1	326	243	226
0	572	2698	975
1	276	624	1003



afinn	-1.0	1.0
sentiment		
-1	326	226
1	276	1003

Accuracy of this dictionary, according to Finn Årup Nielsen, for positive, neutral, negative separation (the first matrix) is 58.00% and 72.58% for just positive and negative separation.

The data I used for a lot larger than in part 1 and was stored using Microsoft Azure Storage Account, because I was using Hive to perform operations on Gigabytes of twitter data it was my best option to store my data as a blob on their storage. I decided to use Microsoft Azure platform for the part 2 of my project because it provides Hadoop on Windows. The latest version of Hadoop for Windows was not the newest: 2.7.0 (HDI 3.3). Linux, for example, has Hadoop versions 2.7.1 (HDI 3.4) and 2.7.4 (HDI). Also, Azure HDInsight has a preview of Hive with new LLAP technology which makes it even faster (Interactive Hive) on Linux. After trying them all and spending a lot of money (over 400$), I decided to go with Windows, as it seemed to be substantially cheaper than all of the above. It also provided me with capability of connecting to it from Excel via ODBC Hive driver to build some graphs using its power view tools. That literally let me run queries against the tables and views I built during the sentimental analysis directly from HDInsight dashboard. According to the specifications found on support.office.com, Excel 2016-2013 can have a maximum of 1,048,576 rows. And 64-bit (the one that I happened to have on my laptop), doesn't have any hard limits on file size, which makes possible the analysis of large amount of tweets using Data Model Workbooks. It lets you create a PivotTable with aggregated data that is obtained from the Hadoop Windows Server on HDInsight. Before I get to visualization part, I needed to perform the analysis of the tweets using hive. I used method, which includes creating 3 views and 1 table that is provided by one of the Hortonworks articles. I, however, used a different dictionary, different data, and different method of calculating my sentiment since the dictionary was different and did not just include the polarity of -1, 0, and 1 like the one in the Hortonworks article, but it included words and emoji for one column and their polarity in another(from -5 to 5). I decided to use AVG() hive function to calculate the sentiment of each tweet after using LATERAL VIEW EXPLODE twice in views to get each words and then joined the resulting view with AFINN dictionary words to get the polarity value of words in tweet. After that, I created a new table joining the table with just id of a tweet and its


Cluster size

To learn more, visit our pricing page.
Learn more

Number of Worker nodes: 5

* Worker node size
D4 v2 (5 nodes, 40 cores)

* Head node size
D4 v2 (2 nodes, 16 cores)

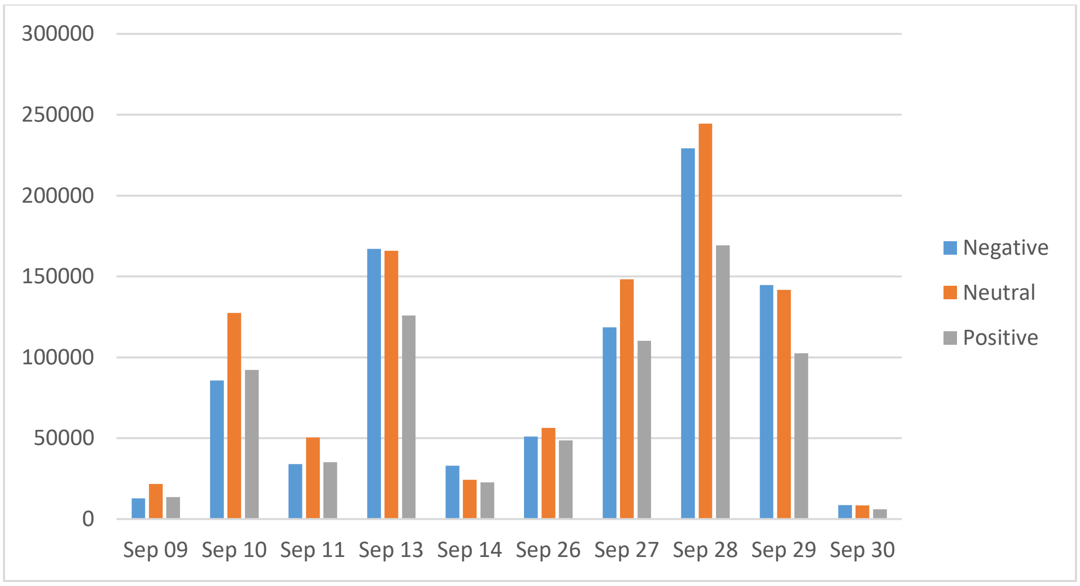
WORKER NODES	1.243 x 5 = 6.215
HEAD NODES	1.243 x 2 = 2.486
TOTAL COST	8.70 USD/HOUR (ESTIMATED)

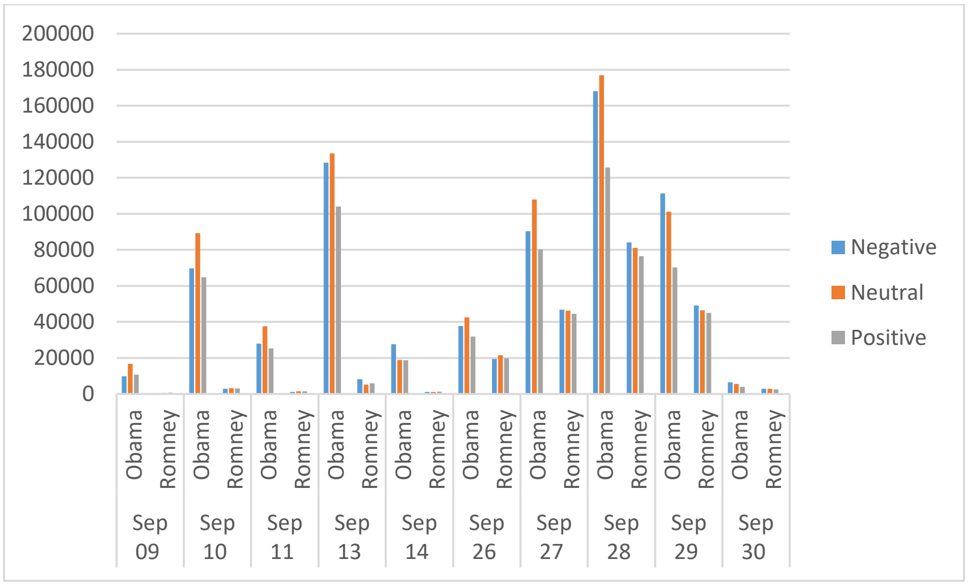
56 of 60 cores would be used in East US.

average value of polarity and the initial table on id to get the date and user location fields in order to be able to graph some data. Then I connected to HDInsight with the method I described above and got the graphs. For my head nodes I used the following specs:

My final solution used 56 cores and with 5 working nodes with the specs above. Price was $7.55 an hour. My concern was whether excel would be able to obtain a large amount of rows from Microsoft Azure HDInsight server. I was very impressed when my laptop was able to load more than 24,000,000 records from one view during the tests. I was also able to graph this data, even though I could not get anything meaningful because that was an intermediate view and I did that just to make sure Excel will be good enough for graphing my data.

The following graphs shows the sentiment results for the word Obama for 10 days. (Sample, actual data can be found in a separate file)



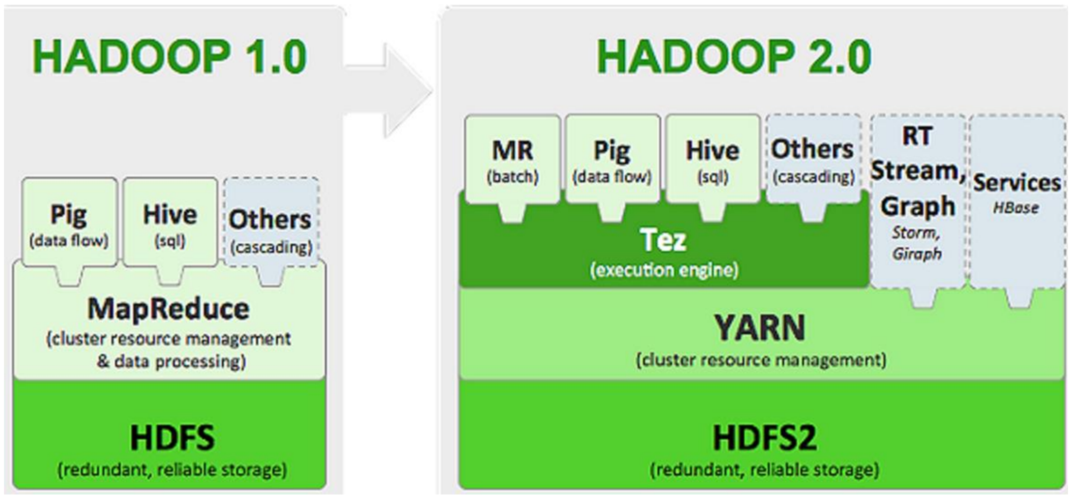


The above graph shows twitter sentiment analysis for Obama and Romney in September. (Sample, actual data can be found in a separate file)

<u>Optimizations</u>

Before executing queries, I used multiple optimizations to be able to work on a large dataset.

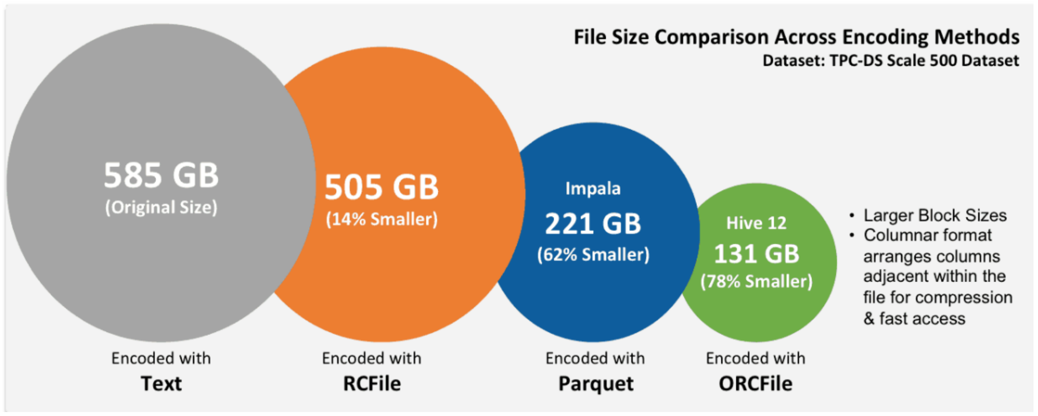
Those optimizations were found (3) and made my queries run a lot fasters. My hive optimizations included: Using TEZ, Storing the tables created as ORC files, Using Vectorization, and using cost-based query optimization. Let's start with TEZ, according to Roopesh Shenoy (4) this engine


HADOOP 1.0
Pig (data flow)
Hive (sql)
Others (cascading)
MapReduce (cluster resource management & data processing)
HDFS (redundant, reliable storage)
HADOOP 2.0
MR (batch)
Pig (data flow)
Hive (sql)
Others (cascading)
RT Stream, Graph Storm, Giraph
Services HBase
Tez (execution engine)
YARN (cluster resource management)
HDFS2 (redundant, reliable storage)

makes the queries run a lot faster. The following paragraph explains why very well. "The main reason for Tez to exist is to get around limitations imposed by MapReduce. Other than being limited to writing mappers and reducers, there are other inefficiencies in force-fitting all kinds of computations into this paradigm – for e.g. HDFS is used to store temporary data between multiple MR jobs, which is an overhead. (In Hive, this is common when queries require multiple shuffles on keys without correlation, such as with join - grp by - window function - order by.)"

The next optimization I used was vectorization. According to the same article, vectorized query execution improves performance of many operations by performing them by bathes of **1024** rows at once instead of a **single** row each time. Another optimization was Cost-based query optimization, which I found extremely useful for almost every query I used. Using this optimization I was able to improve the execution time a lot. For example, I will use the SELECT query to a table that contain 18000000 rows. The results were amazing, time taken decreased from time taken: 443.555 seconds to 113.402 seconds.

And the last optimization was storing the tables as orc files because Hive supports it and according to the article mentioned above provides a faster response time for the hive queries to the table created as an Orc file. Orc file uses predicate push-down, compression and more techniques. (More at 8). Without the above optimization running the queries is very slow and some of them would not even run. In the code I used all of them multiple times to get the best results. Also, I noted, that excel works faster too after the cluster settings were optimized because excel basically executes queries against a table stored at HDInsight server.

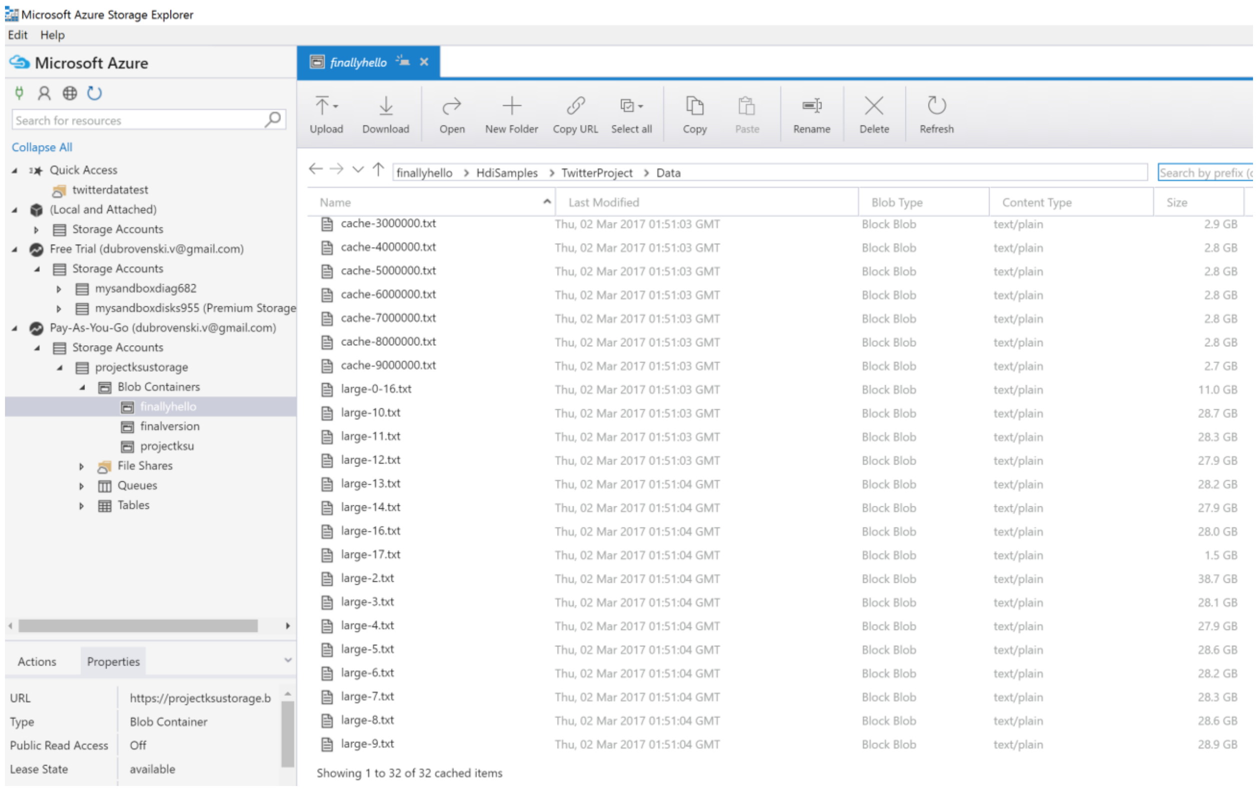


This picture explains why I used orc file as a method for storing my tables.

Conclusion

In this project I tried my best to perform a sentiment analysis on various twitter data. I learned more about Hive, Microsoft cloud services and Excel. I also learned a lot about the optimization, different way of writing queries. I also learned ways to upload the data in data cloud. During the course of this project I improved my knowledge of using Hadoop and Tez. I learned more about Microsoft Azure capabilities and especially about storage explorer while copying my data to azure containers. Overall I enjoyed this project a lot, however the prices of cloud computing are pretty high and I have spent a lot of US dollars trying everything out before even starting working on my project.

Figures:



Microsoft Azure Storage Explorer

Edit Help

Microsoft Azure

- Quick Access
  - twitterdatatest
- (Local and Attached)
  - Storage Accounts
- Free Trial (dubrovenski.v@gmail.com)
  - Storage Accounts
    - mysandboxdiag682
    - mysandboxdisks955 (Premium Storage
- Pay-As-You-Go (dubrovenski.v@gmail.com)
  - Storage Accounts
    - projectksustorage
      - Blob Containers
        - finallyhello
        - finalversion
        - projectksu
      - File Shares
      - Queues
      - Tables

URL	https://projectksustorage.b
Type	Blob Container
Public Read Access	Off
Lease State	available

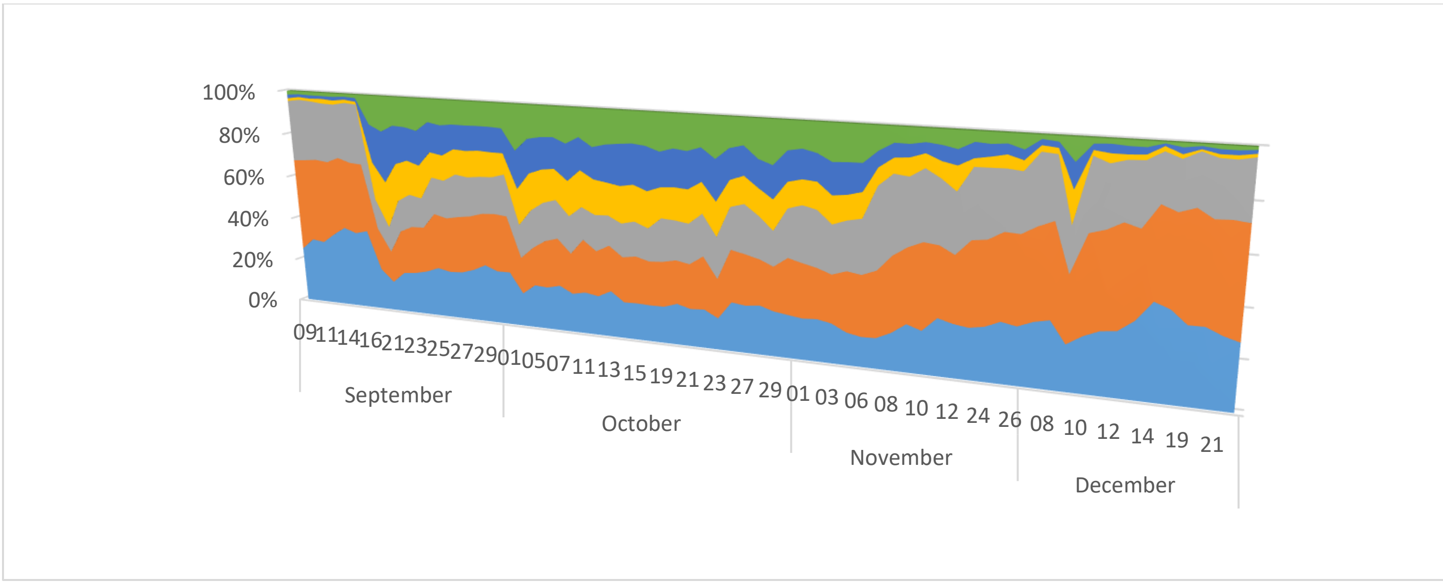
finallyhello

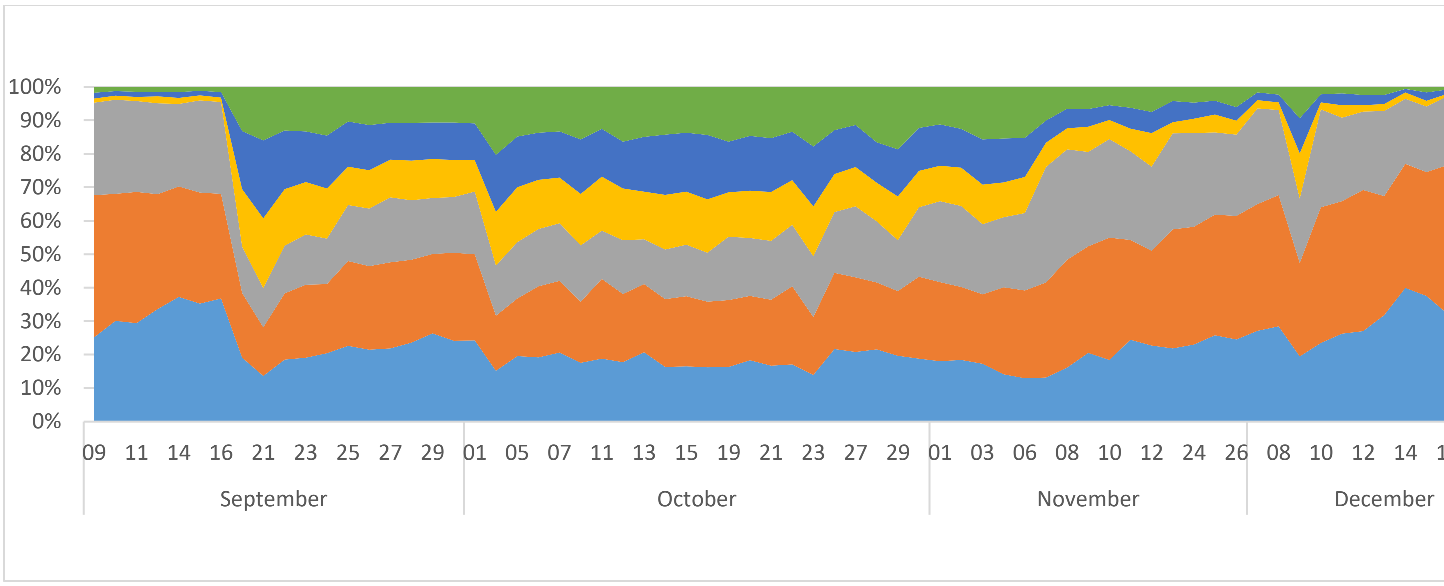
finallyhello > HdiSamples > TwitterProject > Data

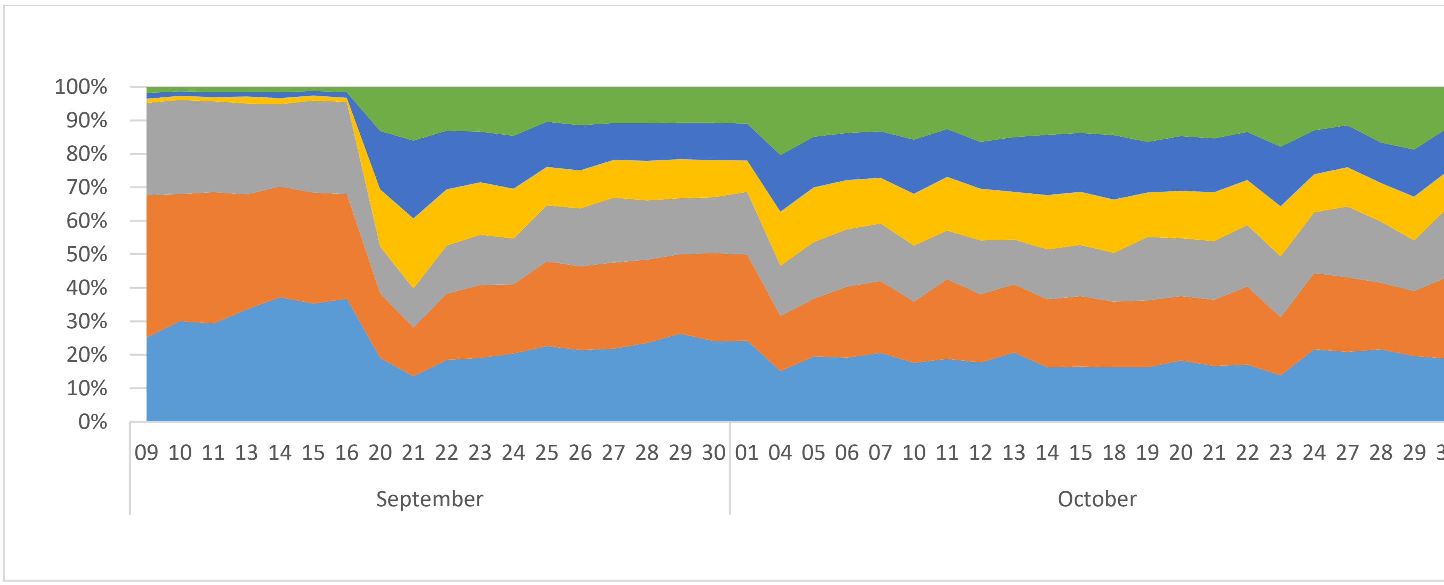
Name	Last Modified	Blob Type	Content Type	Size
cache-3000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.9 GB
cache-4000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.8 GB
cache-5000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.8 GB
cache-6000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.8 GB
cache-7000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.8 GB
cache-8000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.8 GB
cache-9000000.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	2.7 GB
large-0-16.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	11.0 GB
large-10.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	28.7 GB
large-11.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	28.3 GB
large-12.txt	Thu, 02 Mar 2017 01:51:03 GMT	Block Blob	text/plain	27.9 GB
large-13.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.2 GB
large-14.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	27.9 GB
large-16.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.0 GB
large-17.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	1.5 GB
large-2.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	38.7 GB
large-3.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.1 GB
large-4.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	27.9 GB
large-5.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.6 GB
large-6.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.2 GB
large-7.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.3 GB
large-8.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.6 GB
large-9.txt	Thu, 02 Mar 2017 01:51:04 GMT	Block Blob	text/plain	28.9 GB

Showing 1 to 32 of 32 cached items

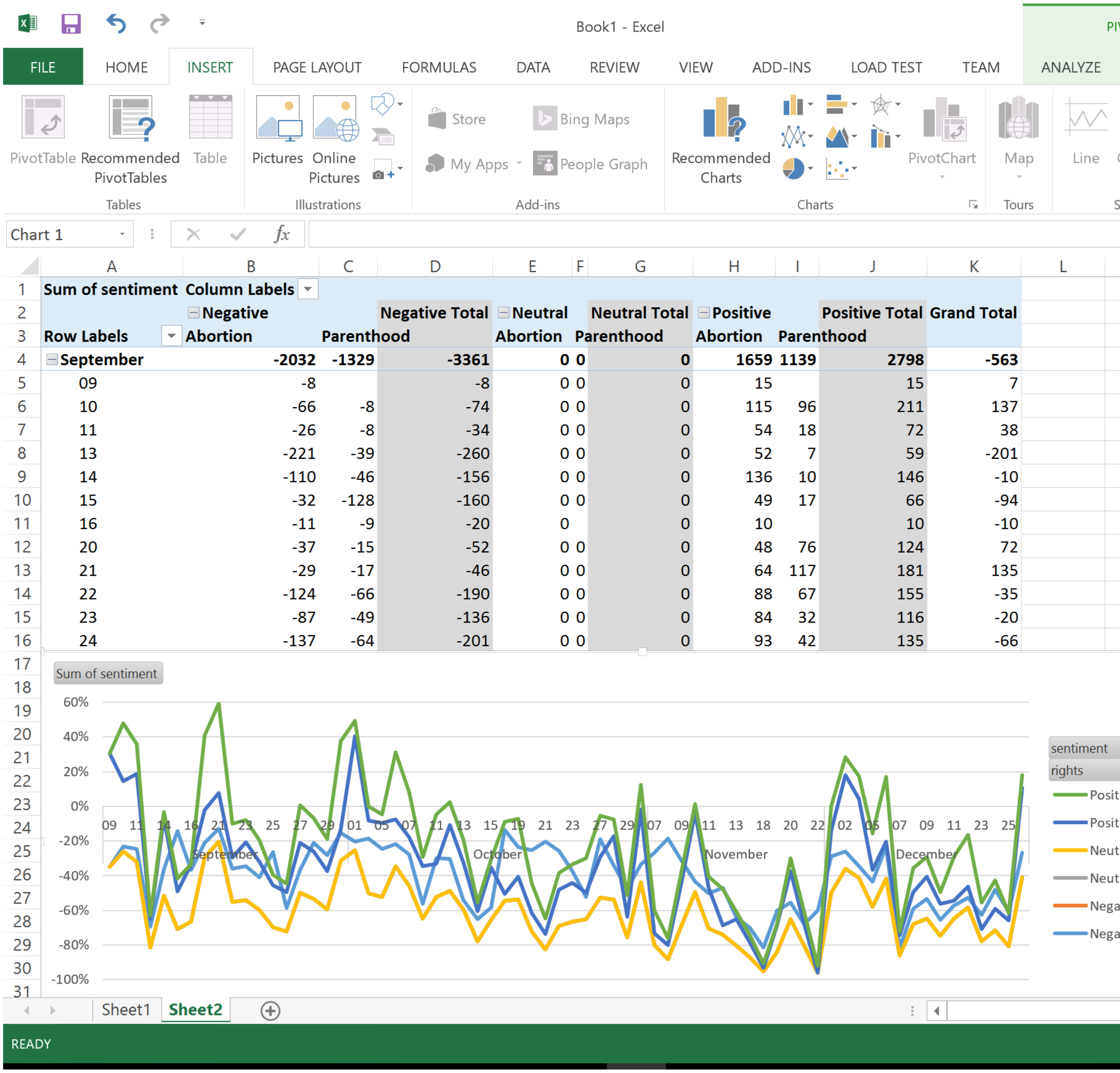
1(Uploaded data)





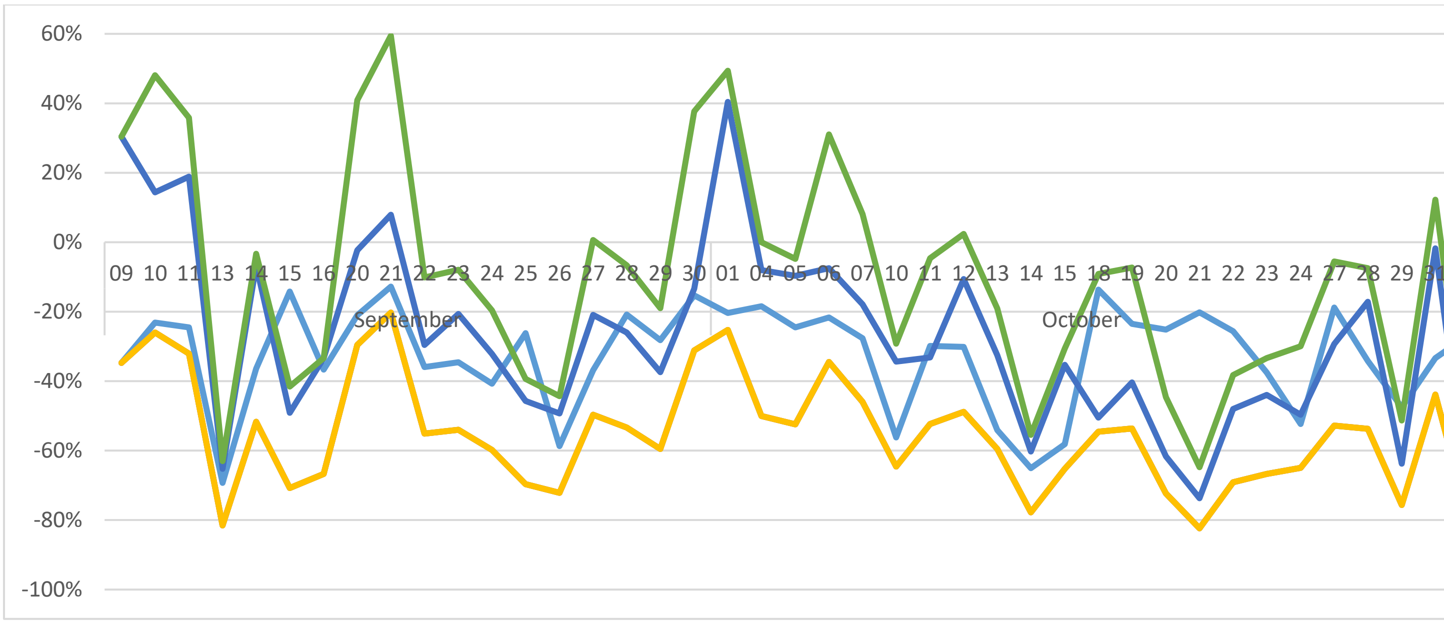


Graphs 1.1, 1.2, 1.3(Obama versus Romney area charts(3D and 2D))

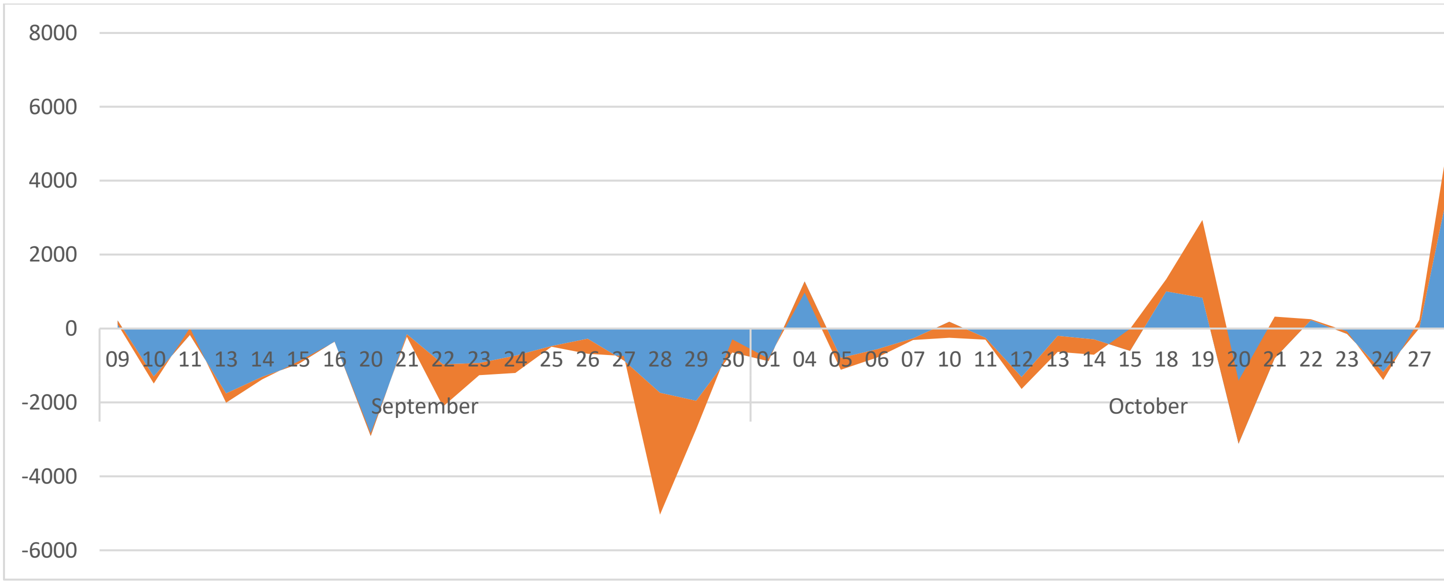


Sum of sentiment	Column Labels									
	Negative		Negative Total	Neutral		Neutral Total	Positive		Positive Total	Grand Total
Row Labels	Abortion	Parenthood		Abortion	Parenthood		Abortion	Parenthood		
September	**-2032**	**-1329**	**-3361**	**0**	**0**	**0**	**1659**	**1139**	**2798**	**-563**
09	-8		-8	0	0	0	15		15	7
10	-66	-8	-74	0	0	0	115	96	211	137
11	-26	-8	-34	0	0	0	54	18	72	38
13	-221	-39	-260	0	0	0	52	7	59	-201
14	-110	-46	-156	0	0	0	136	10	146	-10
15	-32	-128	-160	0	0	0	49	17	66	-94
16	-11	-9	-20	0		0	10		10	-10
20	-37	-15	-52	0	0	0	48	76	124	72
21	-29	-17	-46	0	0	0	64	117	181	135
22	-124	-66	-190	0	0	0	88	67	155	-35
23	-87	-49	-136	0	0	0	84	32	116	-20
24	-137	-64	-201	0	0	0	93	42	135	-66

Graph 2.1(Abortion vs Parenthood Excel View)



Graph 2.2(Abortion vs Parenthood Full Area Chart View)



Graph 3.1(Republicans vs Democrats graph [sum sentiment])

Works Cited:

(1) "Sentiment Analysis." *Wikipedia*. Wikimedia Foundation, 25 Feb. 2017. Web. 01 Mar. 2017. <https://en.wikipedia.org/wiki/Sentiment_analysis>.

(2) Fnielsen. "Afinn." *GitHub*. N.p., n.d. Web. 01 Mar. 2017. <https://github.com/fnielsen/afinn/blob/master/notebooks/CrowdFlower%20Twitter%20sentiment%20analysis%20with%20emoticons.ipynb>.http://www.internetlivestats.com/twitter-statistics/

(3) "How to Make Your Hive Queries Run Faster on Hadoop." *Hortonworks*. N.p., 17 May 2016. Web. 01 Mar. 2017. <https://hortonworks.com/blog/5-ways-make-hive-queries-run-faster/>.

(4) Shenoy, Roopesh. "What Is Apache Tez?" *InfoQ*. Infoq, n.d. Web. 01 Mar. 2017. <https://www.infoq.com/articles/apache-tez-saha-murthy>.

(5) Sherman, Erik. "Many Twitter Users Don't Tweet, Finds Report." *CBS News*. CBS Interactive, 14 Apr. 2014. Web. 01 Mar. 2017. <http://www.cbsnews.com/news/many-twitter-users-dont-tweet-finds-report/>.

(6) Moujahid, Adil. "An Introduction to Text Mining Using Twitter Streaming API and Python." *An Introduction to Text Mining Using Twitter Streaming API and Python // Adil Moujahid // Data Analytics and More*. N.p., 21 July 2014. Web. 01 Mar. 2017. <http://adilmoujahid.com/posts/2014/07/twitter-analytics/>.

(7) Luce, Laurent. "Laurent Luce's Blog." *Twitter Sentiment Analysis Using Python and NLTK | Laurent Luce's Blog*. N.p., 2 Jan. 2012. Web. 01 Mar. 2017. <http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/>.

(8) Shanklin, Carter. "ORCFile in HDP 2: Better Compression, Better Performance." *Hortonworks*. N.p., 06 Sept. 2013. Web. 01 Mar. 2017. <https://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/>.