**Sentiment Analysis of Smartphone Topics From Twitter Data**

Authors: Allan Lo, Eric Yang, Prajakta Pandharkar

**Business Problem**

Develop a data storage/retrieval system to help companies monitor product sentiment via Twitter tweet data. We’ve used Three major Smartphone manufacturers (Apple, Google, Samsung) as a use case for our system.

**Data Storage & Retrieval Issues Addressed**

1. Acquiring Data (Velocity) - Daily query of Twitter Search API to acquire tweets on an ongoing basis.
2. Data Cleaning/Scoring (Variety) - Cleaning of various data types (string, date, etc.) including unstructured tweet data into usable formated data. Create scoring data from tweets to inform sentiment analysis.
3. Storing/Serving Data (Volume) - Develop scalable solution for incoming tweet data storage via Mongo DB/Hive.
4. Consuming Data - Provide easily accessible data to end users. Transform data into information - analysis/business insights.

**Application Description**

The application queries tweets about major smartphone manufacturers (Apple, Google, Samsung) from the Twitter search API. Data is extracted from twitter in JSON format, this raw data is loaded into a Mongo Database as the tweet data repository. From MongoDB the data retrieved, cleaned, parsed, and transformed via a python script and outputted in tab delimited files. The sentiment analysis method utilizes the Vader package in the python Natural Language Toolkit (NLTK) library [1]. The output of the structured analytical result is then loaded into a Hadoop Distributed File System which we use to create a database in Hive. For the data serving layer we connect tableau to Hive Server 2 and load an extract of the data to Tableau Online to make the analysis available for public consumption. In Figure 1, we show the overview of the application architecture and the different layers of workflow.

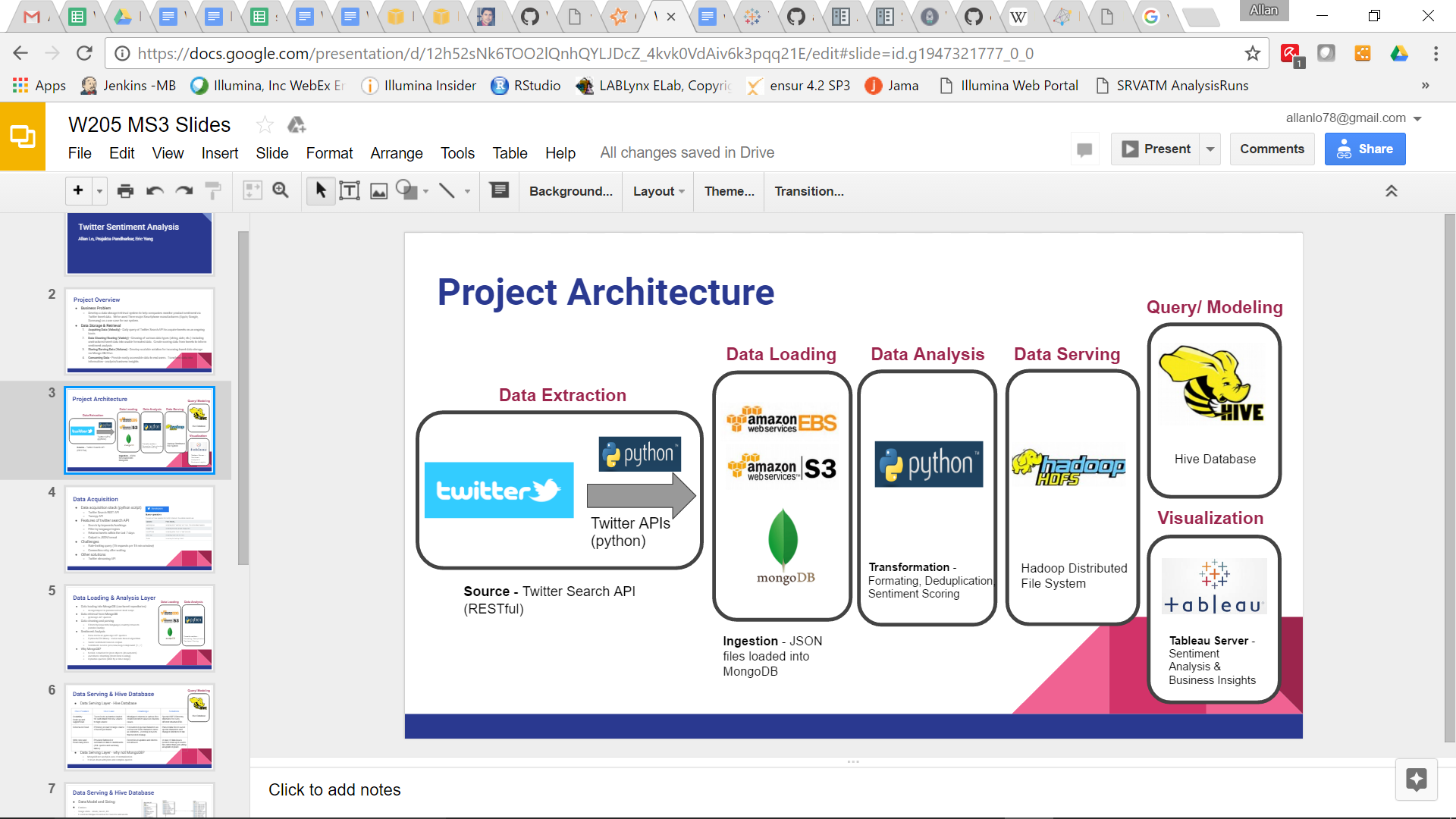


Figure 1. System architecture and application layers of the project

We recommend the UCB-W205 Spring Ex 2 Image (m3.large) for setting up this application.

**Application Components**

1. **Setup Script** - The main src/setup/setup.sh script installs dependencies required python 2.7, OS packages, python packages, and MongoDB.

2. **Twitter Search API** – Data extraction from Twitter source is done through Twitter search API and Tweepy to handle OAuth authentication processes. Twitter search API has a rate limit of 15 queries per 15 minutes window and maximum of 100 tweets per query. We implemented a python script “collectTweets.py” that use the above APIs to retrieve raw tweets based on search terms in Table 1.

The search API performs a case-insensitive query and returns tweets up to a 7 days prior. We used a cron job to collect tweets in JSON files on each brand starting from October 24 up to November 30. Raw tweets were filtered by the occurrence of the keyword “phone” and merged into a single JSON file for data loading in MongoDB.

Table 1. Query strings used in Twitter Search API

|  |  |  |
| --- | --- | --- |
| **Brand** | **Query strings** | **Total Number of Downloaded Tweets (after filtering)** |
| Apple | “apple”, “iphone” | 96,850 |
| Google | “google”, “pixel”, “pixel phone”, “android” | 107,032 |
| Samsung | “samsung”, “galaxy”, “s7” | 16,826 |

3. **Mongo DB** – The install\_mongo.sh file does a yum intall of Mongo DB and starts running the database. The script should be run in root. The script also copies the mongodb.repo file from the cloned github repository to the following location /etc/yum.repos.d/

Load Mongodb using raw json files using a shell script. This scripts reads from a file\_list.txt on

merged json files per brand. It creates the MongoDB and MongoDB collections for the listed

brands and loads them to Mongo.

MongoDB considerations:

MongoDB is a NoSQL solution for json document objects. At present current application considers only three brands. MongoDB is useful for immutable such as the raw tweets data collected from twitter. MongoDB automatically scales horizontally in case additional documents related to more products are needed. MongoDB also supports dynamic queries that can efficiently query document objects by retaining their properties. The application parses the data making use of MongoDB Collections one per brand to produce data for sentiment analysis collections for the listed brands and loads them to Mongo.

Raw data sizing estimation for MongoDB.

Assuming around 10,000 tweets a day. Estimated Size: (MongoDB and Hive)

* 1 tweet ~ 280 bytes => 280 \* 10000 \* 3 = 8400000 bytes
* => 8.4 MB per day \* 30 days = 252 MB (detail stage table)
* => 252 \* 2 tables + 45 \*3 small tables = 639 MB.

Hence, considered 1GB per brand for allocation.

4. **Data Transformation and Sentiment Scoring** – The analysis layer of this application consists of three parts: data retrieval from MongoDB, data cleaning and transformation into structured data, and applying a sentiment scoring method. Data retrieval from MongoDB is done by pymongo API where the python script initiates a connection to MongoDB and retrieves the document collection (eg. “twitter\_apple\_collection”) in a pandas dataframe. Data parsing is done on “user”, “place”, and “geo” fields to extract relevant information. The resultant dataframe is then filtered by several predefined rules:

1. “English-only” tweets
2. “US-only” tweets
3. Removal of retweets
4. Removal of redundant tweets by the same user (the latest tweet is kept)
5. Keyword filters: words that may indicate an irrelevant tweet (eg. “fruit”, “food”, “clearance”, “phone case”, “LA galaxy”)

The sentiment analysis is implemented using the Vader package of python NLTK library. The Vader algorithm is a lexicon and rule-based sentiment scoring system where each token in a sentence is matched with the lexicon corpus that have associated sentiment polarity scores. We chose this algorithm because it is relatively fast and it requires no training. Furthermore, the algorithm has been benchmarked against other ML methods in several domains including twitter data and outperforms them.

Each tweet receives four sentiment scores - “pos”, “neg”, “neu”, and “compound”; “pos” is the score for positive sentiment, “neg” is the score for negative sentiment; “neu” is the score for neutral sentiment, each of the above range from [0, 1]; “compound” score is the normalized score ranged from [-1, 1] that can be used as the single metric for overall sentiment scoring. The output of the analysis is a structured format (tab delimited file) with extracted fields for data loading in Hive.

5. **HDFS:** The serving layer uses HDFS to store the files required for analytical layer provisioned by HIVE. Using bash script analytics\_dataload.sh, required directory structure is created in /user/w205/. For every brand a separate folder structure is created. The output files of analysis data is stored in the form of tab delimited files.

HDFS considerations:

Application is set up with hadoop distributed files system which stores data at present in AWS instance for only required brands. However, this setup is scalable for additional brands in future. As the data increases, the HDFS can be pushed to AWS-S3 system for scalability.

6. **Hive-ERD-DDL -**– At the end the bash script prepares normalized and analytical tables by calling analytics\_ddl.sql. ER Data model is as shown in the diagram below.

Apache Hive Database considerations:

To render the sentiment analysis using Tableau dashboard it is necessary to precalculate data from HDFS files. Hive provides various features that allow sql queries to reorganize data fed to Tableau. Here is the summary of Hive features used for application and challenges faced while using them.

Table 2. Hive Database Feature Summary

|  |  |  |  |
| --- | --- | --- | --- |
| Hive Feature | Use Case | Challenge | Solutions |
| Scalability:  Scale up and support load | Tweet-feeds as batches loaded for each brand from low volume to high volume | Misaligned columns in various files created incorrect values in columns issues. | Special HDFS Directory structures for every different structured file. |
| Schema on Read | Efficiency in load for large volume of tweets per brand. | Encountered special characters as well as text fields characters same as delimiters - (comma) in tweets that needed cleanup | Parsed data files to avoid special characters and changed delimiters to tab |
| Write once and Read many times | Provision flattened or summarized data to dashboards. (SQL queries and summary tables) | Record level updates and deletes not allowed. | In case of data issues needed clean up in source files rather than just writing an update in place. |

Summary of Entity and purpose:

Hive database instance - product\_analytics

HDFS location - 'hdfs://localhost:8020/user/hive/warehouse/analyticsdb/'

Table 3. Entities in Hive Database

|  |  |
| --- | --- |
| Entity | Purpose and Description |
| clean\_tweet\_dtl | This entity stores the filtered and clean (free from special characters) tweet sentiment analytics output. This entity contains statistical attributes and brands as well.  Since all the brands have same structure, all the data files are staged in this entity. This entity serves as a detail table for Tableau. |
| location | This entity is loaded from staged data in clean\_tweet\_dtl. It is normalized locations for twitter users. |
| users | This entity is loaded from staged data in clean\_tweet\_dtl and it is normalized data about the users. |
| sentiment\_check | Derived table.  This entity represents data required for Tableau summary dashboard. This entity is loaded from staged data in clean\_tweet\_dtl. |
| features | This entity is loaded from a flat file”features.txt”. This contains important keywords and features we consider users may be discussing over twitter. This entity is used for tokenization to avoid common english prepositional words and verbs and allows analysis on mobile brand related keywords. |
| wordcloud | Derived table.  This entity calculates number of times a word is seen in the “text” attribute of a tweet. The count of each word in text is stored per brand. |

Since the twitter data contains string date format which does not match easy date conversion, complex derived attribute for dates are defined in all normalized and derived tables.

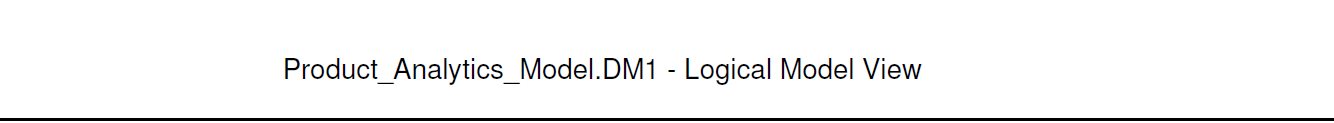
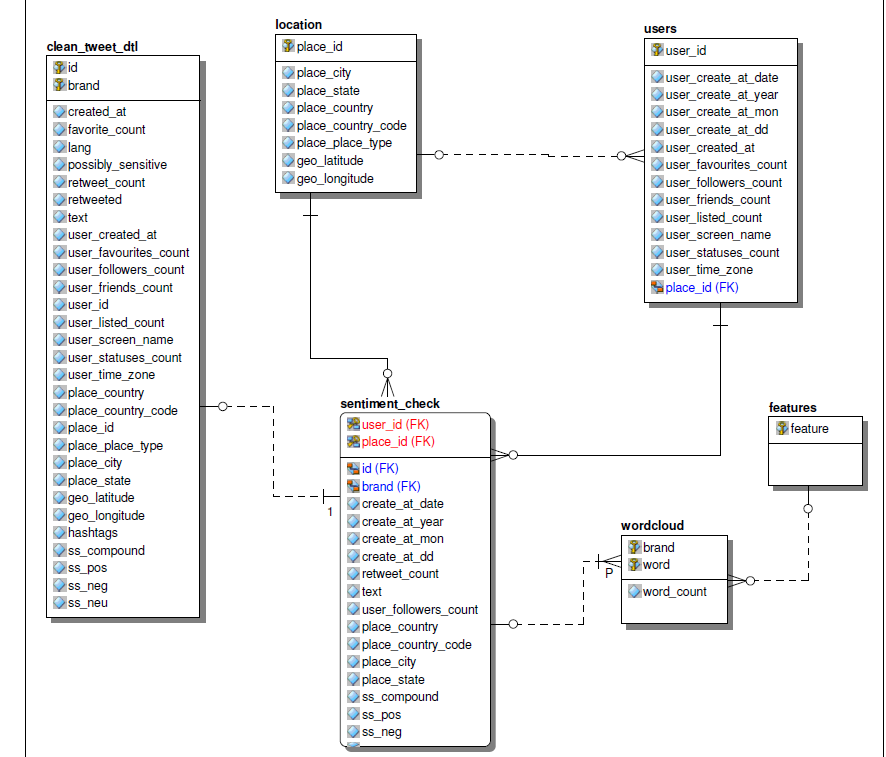


Figure 2. Entity relationship diagram of product analytics model

7. Other considerations:

At present the application collects the tweets on a daily basis in the form of batches. These files are then merged to load to MongoDB and collection per brand. For future usage, considering the growing size of the tweet files, the application can be modified to insert new batch data incrementally. The application stages clean tweets in a single table considering future growth The current location of the files from AWS EBS volume can be changed to AWS S3.

To support near real time sentiment analysis, the data can be ingested using near real time streaming

8. **Tableau -** The Tableau\_Sentiment\_Final.twb file is a Tableau workbook file that is used to create a Tableau dashboard. Tableau is connected to Hive Server 2 via a Cloudera Hadoop ODBC Driver. Tableau is also used to extract data from Hive and load the data to Tableau Online. This allows us to publish our dashboard online and access data with lower latency.

[Link to Cloudera Hadoop ODBC Driver](http://www.cloudera.com/downloads/connectors/hive/odbc/2-5-12.htm)

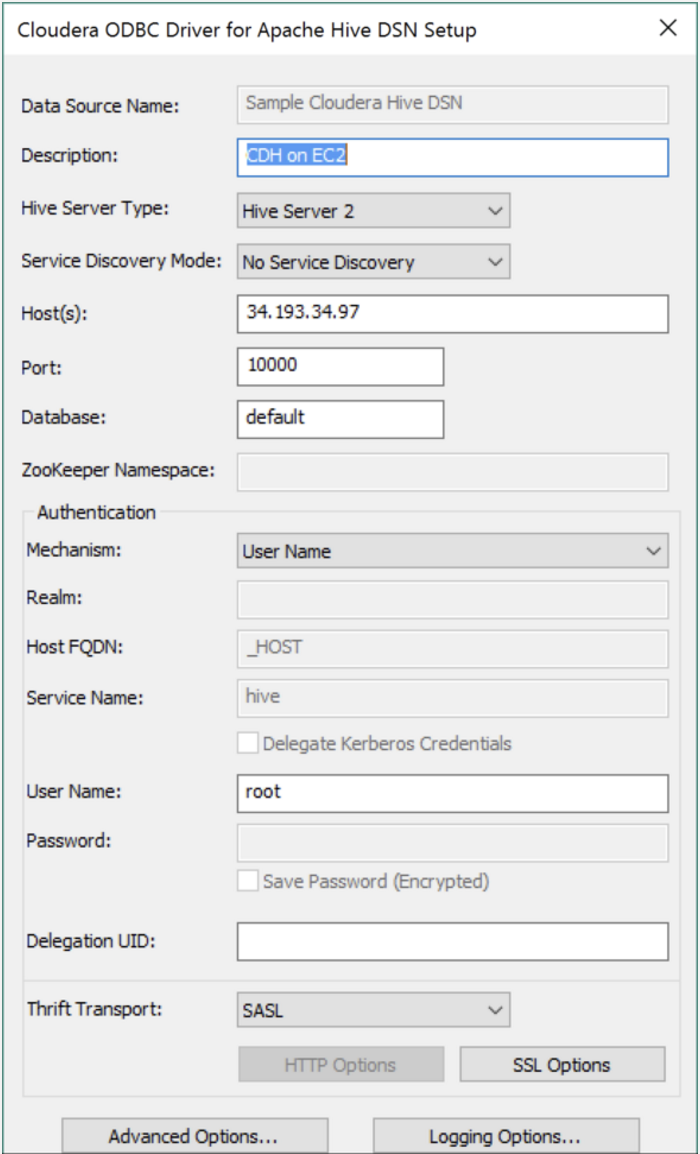


Figure 3. Windows ODBC Setup (Change Host IP)

**Analysis Results**

**Sentiment Overview**

Overall tweets were mostly neutral. We are not surprised by this result since our sentiment analysis is rules based. We are looking for a set of positive or negative words if these words do not appear the tweet will be scored as neutral. It’s also likely that many tweets are not expressing positive or negative sentiment. Tweets tend to have higher proportions of “moderately positive” or “moderately negative” if they carry non-neutral sentiment.

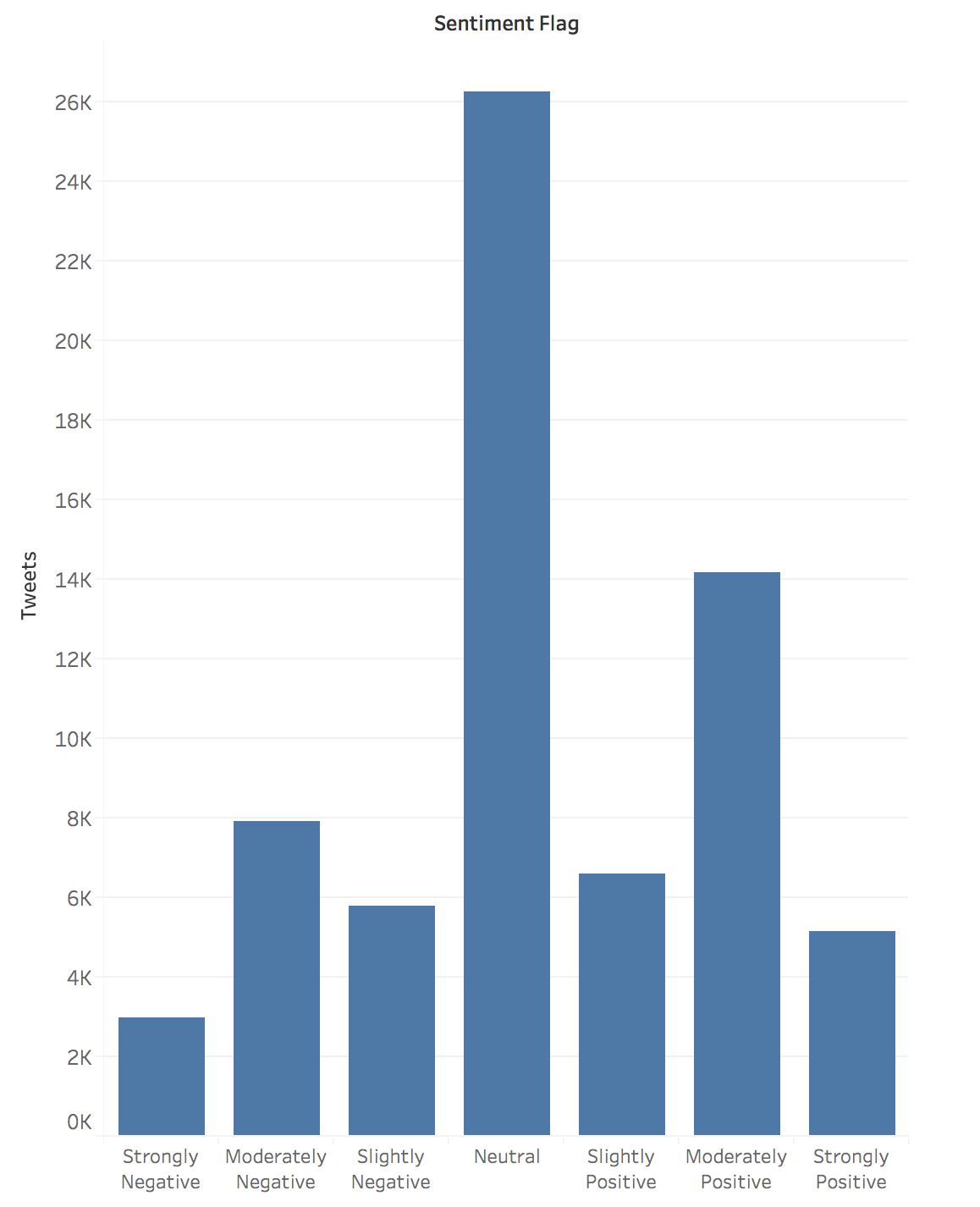


Figure 4. Histogram of sentiment categories of tweets

**Sentiment Over Time**

Looking at average sentiment over time we see that sentiment is largely stable for apple and is relatively neutral. In the last week of October we see that sentiment for Google is relatively more positive. Google launched it’s pixel phone on 10/20/2016, there could potentially be a lot of initial excitement from the launch which eventually subsides over the initial couple weeks. In this same period Samsung has relatively negative sentiment which could be residual negativity from their exploding battery issues which began in late August.

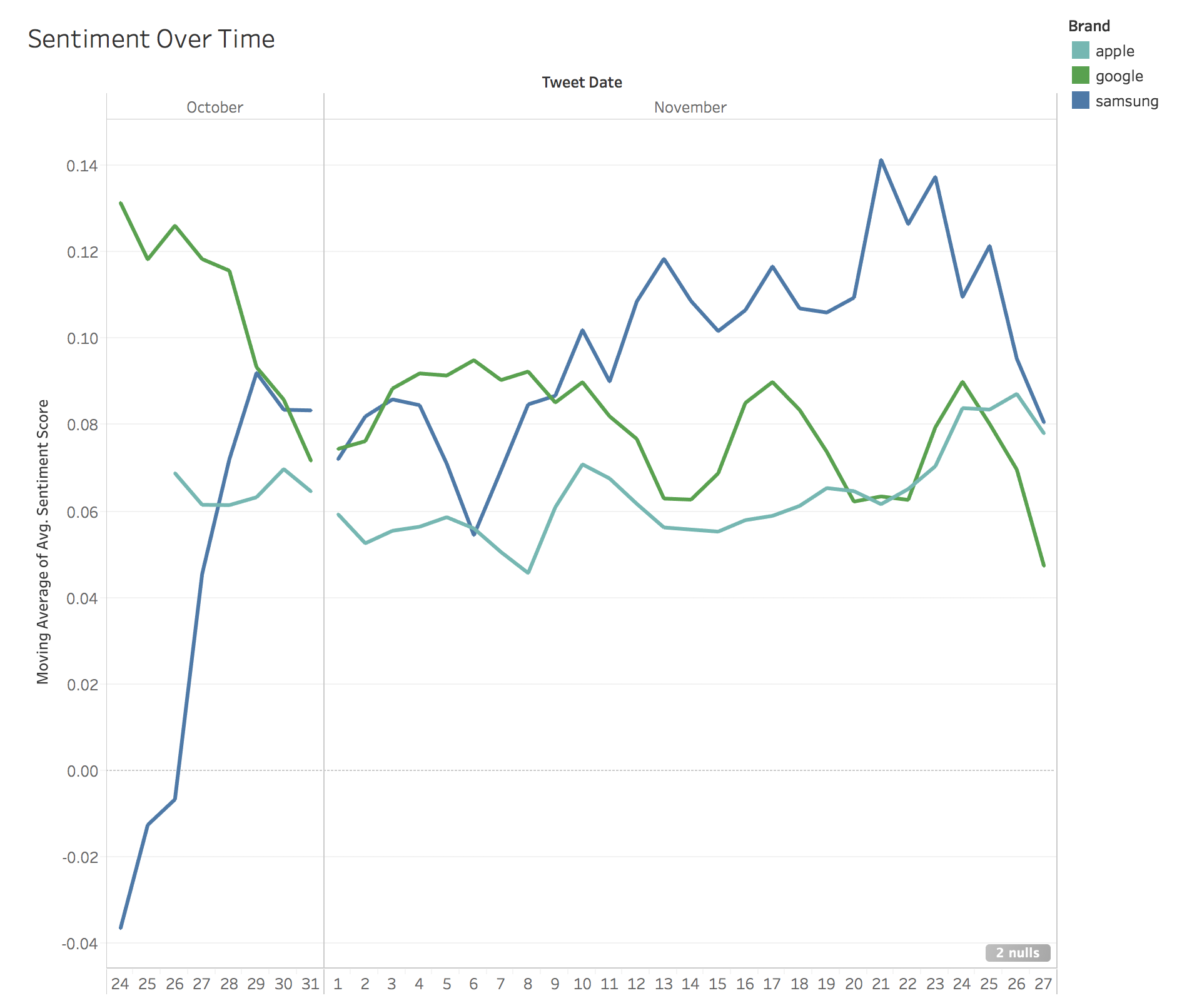


Figure 5. Sentiment score of brands over time

**Word Cloud**

The word cloud below shows the most common words associated with our Samsung tweets. We can see there are many popular words, but both explode and battery show up as top words. This is more confirmation that people on twitter are still interested in Samsung’s latest recall



Figure 6. Word cloud analysis of Samsung smartphone features

**Tableau Dashboard**

Additional analysis and data visualizations are hosted on our Tableau Server:

* + [**Link to w205 Twitter Smartphone Sentiment Dashboard**](https://10az.online.tableau.com/#/site/midsw205project/views/TwitterSmartphoneSentimentAnalysis-Final/SmartphoneTweetSentiment?:iid=4)
  + **UserName: ericy330@gmail.com**
  + **Password: midsw205**

**Files, Descriptions, and Locations**

|  |  |  |
| --- | --- | --- |
| **File Name** | **Description** | **Location** |
| setup.sh | Main setup script for python2.7, system packages, python package, and MongoDB installation; calls install\_mongo.sh | src/setup/setup.sh |
| install\_mongo.sh | Installs Mongo and starts the database | src/setup/install\_mongo.sh |
| mongodb.repo | Mongo install file | src/setup/mongodb.repo |
| collectTweets.py | Collects tweets by a query string and output json files | src/scripts/collectTweets.py |
| features.txt | This has static list of keywords and features for MongoDB load | src/scripts/features.txt |
| file\_list.txt | List of merged files (parameters) for each brand for mongodb\_load.sh | src/scripts/file\_list.txt |
| mergeJson.sh | Applies a keyword search filter and merges the raw tweets in a directory in a single json file | src/scripts/mergeJson.sh |
| mongodb\_load.sh | Loads mongodb and collections per brand | src/scripts/mongodb\_load.sh |
| runCollectTweets.sh | Bash script called by the cron job to collect tweets | src/scripts/runCollectTweets.sh |
| sentimentAnalysis.py | Sentiment analysis python script | src/scripts/sentimentAnalysis.py |
| analytics\_dataload.sh | Creates hdfs file locations. Creates Hive tables and loads them | src/scripts/analytics\_dataload.sh |
| analytics\_ddl.sql | This sql defines the hive tables and is called through analytics\_dataload.sh | src/scripts/analytics\_ddl.sql |
| Tableau\_Sentiment\_Final.twb | Tableau Workbook | src/tableau/tableau\_sentiment\_final.twb |
| apple\_result.tsv | Analysis output of Apple | data/analysis\_outpu/apple\_result.tsv |
| google\_result.tsv | Analysis output of Google | data/analysis\_output/google\_result.tsv |
| samsung\_result.tsv | Analysis output of Samsung | data/analysis\_output/samsung\_result.tsv |
| Smart\_Phones\_Tweets\_iPhone\_11\_01\_2016.json | Sample smart phone tweets for apple | data/rawdata/apple |
| vader\_lexicon.zip | Lexicon corpus required by Vader sentiment analysis package | data/vader\_corpus/vader\_lexicon.zip |

**Files Dependencies**

|  |  |
| --- | --- |
| **File Name** | **Dependencies** |
| NLTK vader package | data/vader\_corpus/vader\_lexicon.zip |
| install\_mongo.sh | mongodb.repo |
| mongodb\_load.sh | install\_mongo.sh  file\_list.txt |
| analytics\_dataload.sh | sentimentAnalysis.py  analytics\_ddl.sql  features.txt |
| Tableau\_Sentiment\_Final.twb | Cloudera ODBC, Hive Server 2 |

**Reference**

1. Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.