Mr. Steal Yo Elections

Divyansh Gupta, Bhaanu Kaul, Zakk Lefkowits, Mikias Alemu. Mr. Steal Yo Elections

Abstract

This project focuses on harnessing the power of a vast microblogging network, Twitter, to gain insight into the political leanings of the American people. Traditionally, polling has been used to achieve the same goal, yet polling has many shortcomings - notably the immense cost to reach a large audience. Twitter bypasses this by providing simple access to billions of users' thoughts, opinions, feelings, etc. By applying modern natural language processing and machine learning techniques to Twitter data, we are able to produce quantitative data about the American people's emotions and how they correlate to voting in presidential elections.

Introduction

The last few elections in the United States have been some of the most polarizing elections in modern history. Polling a representative sample of individuals on their feelings towards political candidates has long been used as a way to gauge how the American public is leaning politically. This is most commonly done through cold calling households, a decreasingly effective technique. The 2016 U.S presidential election was a lesson in the inadequacy of polling and polling metrics as a whole, with Microsoft's Bing prediction topping 91% for the Democrats. The issues encountered with traditional polling include diminishing use of home phones, demographic bias, bias resulting from wording in polling questionnaires, as well as a polling atmosphere where participants may not feel comfortable enough to voice their opinions.Throughout the last decade, the world has unknowingly been provided with an alternative, more effective and reliable way to determine the public's lean in politics - social media.

Social media has created new ways to communicate and spread ideas. This has caused a political uproar and polarization as well us given us a critical tool as an alternative to polling. The analysis of U.S. presidential elections is a complex and historied field. This project involves running political analysis and sentiment analysis on a large number of political election-related tweets to achieve a number of objectives.

Objectives

This project discusses the correlation, if any, between user sentiment and political affiliation based on Twitter tweets around the United States.

- What is the, if any, correlation between a person's emotions and their political leanings?
- How can a correlation between sentiment and political affiliation be used to sway voters toward a certain party or candidate?
- Can this correlation, if any, be used to reproduce the results of previous elections or predict future elections?

Tools

To analyze our dataset, our team used invaluable tools like Python and AWS. One of the libraries our team used was Tweepy, an easy to use Python library for accessing the Twitter API. The natural language processing library for Python, TextBlob, is built off of the popular NLTK library and proved to be an invaluable tool for text processing, political analysis, and sentiment analysis. For storage, computing and scalability Amazon Web Services was used.

Data Collection & Sampling

There are two main sources for the project's raw tweet data: the Harvard dataverse, and Datahub. The Harvard dataverse contains Twitter data from 2016. Datahub contains data from 2012. We originally hoped to get data from the 2008 election; however, there were no convenient sources which provided this data. We wanted 2008 and 2016 data because these two election years marked a change in the presidential party. In 2008, the U.S switched from a Republican to a Democratic president, and in 2016 the U.S switched from a Democratic president to a Republican president. Since we could not find any data from 2008, we chose 2012 since it was the only other presidential election in the Twitter era, and the only other election data we could find.

The 2016 data contains about 280 million twitter status identifiers. These identifiers are used along with the Twitter Search API to get a complete JSON object of each tweet. The 2016 data ranges between July 2016 and November 2016. The data was aggregated using a list of politically relevant hashtags, scraped party affiliate timelines, and scraped data from events such as debates and conventions. Since the 2016 data required calls to the Twitter API, we were heavily rate limited and chose to focus on exclusively processing 2012 data for the scope of this project.

The 2012 data contains over 170 million individual tweets between July 2012 and November 2012. The difference between the 2012 data and 2016 data is that the 2012 data is already in a JSON format, compared to the 2016 where we only have tweet id's. This data was gathered from various search terms relating to the political climate of 2012. This data shares the same concern as the 2016 data. The 2012 data files range between 400 megabytes and 5 gigabytes. These data files, however, were compressed. After decompression, these files were 40GB or larger. Using the linux split command, we divided each 40GB file into smaller ~3GB files of 1 million tweets each. Since this data is already in JSON format, it was a lot easier to work with.

A concern for the data, along with the large size, is the timeframes used during the analysis. With the 2016 data, since there are only have tweet identifiers, the range of the data was unknown until it was processed. This is also a similar issue with the 2012 data. Since there is so much data, the exact ordering of the data is unknown and again the timeframes may not match up with each test data set. The solution for this was to find a way to sort the massive dataset after analysis, which was an expensive operation.

Since the 2012 data set contain a large amount of data in large data files, the data was broken up into smaller pieces. Upon breaking the files up into 1 million tweet file lengths, the pipeline to process the tweets also only processed every 25th tweet. This provided a random sampling of the large data set.

Data Fields

Each tweet object contained many extraneous fields not needed for analysis. The main fields used from each tweet object were: text, hashtags, created date, and retweets. The text field and hashtags were used for both sentiment analysis and political classification. The retweets were used for weighting the tweet. We considered that a tweet with a higher amount of retweets would have a larger impact on the sentiment and political classification than one with no retweets. This is because a retweet will allow the information in a tweet to reach a larger population, causing a larger impact. Along with retweets, a user’s verified status, and follower count were considered for weighing a tweet. The created at date was used for creating a timeline. From each user, we recorded the following fields: follower count, friend count (how many people the user follows), location, tweet count, and verified status.

Data Processing

The format of our raw tweet data were 40 gigabyte files from Harvard’s Dataverse. These huge files had to have the following operations done on them:

- Data cleaning (removal of links, extraneous fields) per tweet
- Political Classification per tweet
- Sentiment Analysis per tweet
- Insertion into SQL Database, as a batch
- Final manual analysis, as a batch

A Python driver was created that ran all these processes in a pipeline on a computer. This proved to be too slow, with an output of fewer than 100 tweets per minute. The team quickly iterated to a multi-threaded design, resulting in slightly less than 400 tweets per minute throughput. With this design, the team ran into limitations with

Python's multi-threading capabilities which are limited to a single physical core. This was too slow and unscalable. A multi-processed approach was tried but ran into slow performance problems as well. A whole new approach was needed, and the Tweet Processing Pipeline (TPP) was created.

Tweet Processing Pipeline (TPP)

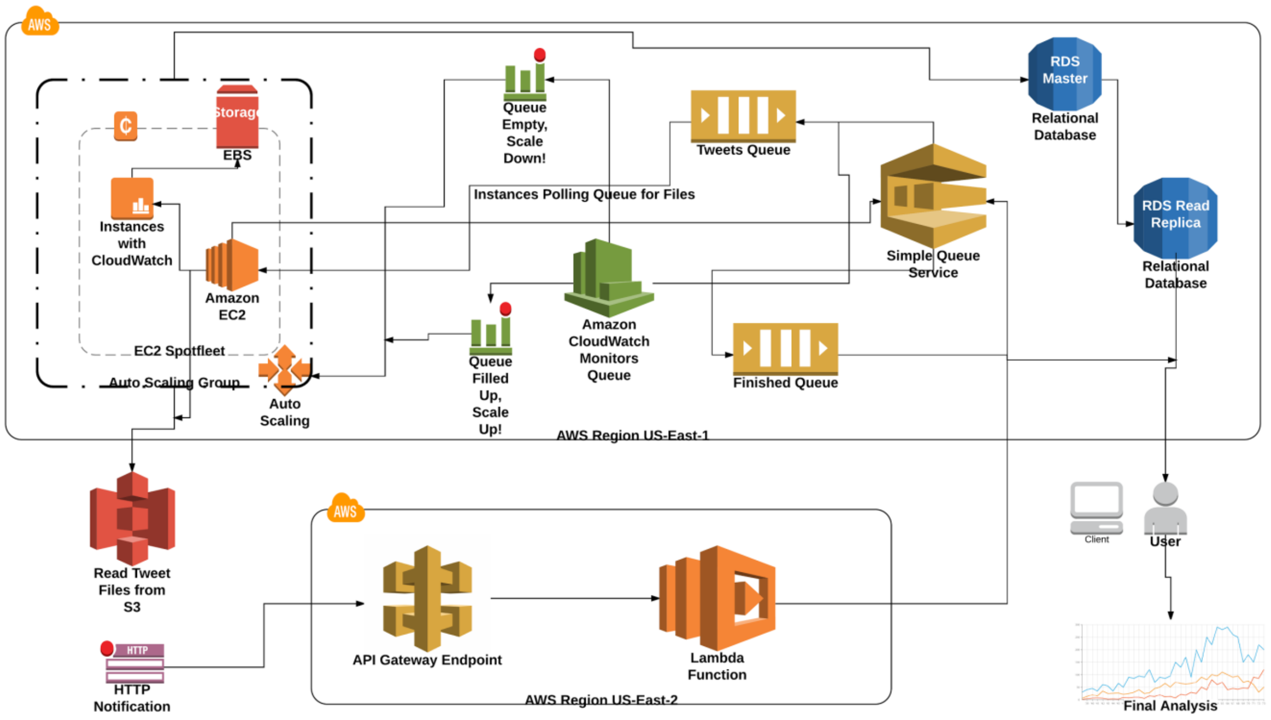


Figure 1: Tweet Processing Pipeline

For large scale (millions) tweet processing, we used Amazon Web Services infrastructure to build our Tweet Processing Pipeline (TPP). TPP, as laid out in Figure 1, enabled us to process a sample of ~3.5 million tweets from our source dataset in a matter of days, with a maximum throughput ceiling that the team did not have the financial resources to test, but theorize is nearly unlimited. The limiting factor in the research quickly went from being time to being financial resources. The pipeline is made up of the following parts:

Pre-requisites: The raw tweet files are stored in S3. Each file is significantly smaller than the amount of RAM available on the EC2 instances on which the analysis will run. Each line in the files represent a single tweet object, in the format that Twitter returns from their Tweet API. There is a pre-trained, serialized classifier (discussed in the following Classification section of the paper) stored in S3 as well.

Step 1: A HTTP Post request is made to a custom endpoint to Amazon's API Gateway. The request contained a file name, representing a file in S3 that contained raw tweets to process. The request also contains parameters, including how many tweets from each file to process.

Step 2: Amazon API Gateway triggers an Amazon Lambda function, which receives the filename from step 1 and passes it to Amazon Simple Queue Service (SQS). SQS queues up the tasks (each task is a file name to process, and all it’s parameters) into the *2012* queue.

Step 3: Amazon CloudWatch is monitoring the number of tasks in the *2012* queue. When the queue contains more than one task that has not been acknowledged by a processing Amazon Elastic Compute (EC2) instance, an alarm is triggered. This alarm, when triggered, will trigger an Amazon CloudWatch Event, starting up an EC2 instance to process the task. If another EC2 instance is sitting idle, it will pick up the next task in the queue. When the queue empties out, Amazon CloudWatch triggers another alarm that will shut down idle EC2 instances.

Step 4: This step deals with EC2 instances. Each EC2 instance is polling the *2012* queue every 20 seconds while sitting idle. The CloudWatch alarms from step 3 ensure no more than 1 EC2 instance is ever sitting idle. When an EC2 instance starts up, it is part of an EC2 Spot Fleet. This means that a bid is placed for the price of the instance, and it only starts up when the bid price falls under that price. The EC2 spot fleet is part of an auto-scaling group, with a pre-configured bootstrap script. This bootstrap shell script installs all the necessary software and pulls down the team’s code from Github and executes it. This ensures that the pipeline is entirely hands-off and requires no manual interaction.

Step 5: The EC2 instance bootstrapped software pulls the task from the queue, pulls the pre-trained classifier from S3, pulls the corresponding tweet raw data file from S3, and begins processing and classifying tweets. Each tweet is cleaned, has sentiment analysis and political analysis run on it, and then is inserted into Amazon Relational Database Service (RDS).

Step 6: When the file is done processing, the finished task is marked as completed by placing it in the *2012_finished* SQS queue. The resulting data is all stored in a master RDS instance. The master RDS instance asynchronously propagates all data to it’s read-replica slave. This allows for minimum locking and maximum throughput for the analysis to be done on the read-replica.
Step 7: A team member manually runs analysis and produces graphs and results from the classified data inside the read-replica RDS instance. This is the result of the pipeline.

Political Classification

Our aim in political classification was to classify text under one of two classes.

- This text indicates the author has positive feelings toward the Republican party
- This text indicates the author has positive feelings toward the Democratic party

Initially the system included 3 classes, Republican, Democratic, and Third Party. These classes were loosely defined without specification of the feeling about the party in the classification. Through the course of the project, we uncovered that we needed to reduce the number of classes and define them in a more concrete manner. This led to the specified classes above; the precise definition of these classes will prove useful in our analysis section.

The political classification system was created by utilizing the Natural Language Toolkit (NLTK) , Sci-Kit Learn, and TextBlob code libraries as part of the Python coding environment. TextBlob provides an abstract text classification model which allows for a specific machine learning classification algorithm to be employed. This allowed for trivial testing of alternate algorithms. The algorithms we tested are as follows: Decision Tree from NLTK, Naive-Bayes from NLTK, and Logistic Regression from Sci-kit Learn.

Building and testing a text classification system requires a set of labeled data. This labeled data consists of a series of objects containing two fields, the label or class of the data (Republican or Democratic in our case), and the text corresponding to the label. Creating this labeled data poses a non-trivial problem - one that affects the accuracy of the resulting system. For our final model, we created the labeled data by using the hashtag table from our processed dataset - this meant we had to perform an ad-hoc re-process of the political classification after our initial processing phase. By having simple access to tweets containing a requested hashtag we were able to build a set of labeled data using campaign hashtags. Our data consists of tweets created during the 2012 presidential election in which the Democratic candidate for president was Barack Obama and the Republican candidate for president was Mitt Romney. Therefore we selected hashtags which supported each campaign and labeled them as the corresponding political party. We found two corollary hashtags for each campaign which provided plenty of data as seen in Table 1.

Republican Classification	**Democratic Classification**
#Romney2012	#Obama2012
#RomneyRyan	#ObamaBiden

Table 1: Twitter Hashtags used for Labeling Training Data

Using these hashtags to identify labels, we selected 20,000 tweets to use as our labeled data, 10,000 republican and 10,000 democratic. Though we were only able to use approx. 12,000 total tweets due to CPU memory constraints.

Once the labeled data was acquired we were able to train and test the classification systems. We split 12,000 labeled data points into a training set and a test set with the training set consisting of a random sample of 80% (9,600 tweets) of the data points and the test set consisting of the remaining 20% (2,400 tweets). Then we cleaned each tweet by removing the following extraneous data: punctuation, stopwords, hyperlinks, and words of length 2 or fewer. Next we built a word set which included all remaining unique words found in the data. We sorted this word set by each word's frequency and removed the bottom 20% least frequent words, all of which were used less than 3 times in 9,600 tweets. A smaller word set helped to reduce a significant amount of CPU resources required by the classification system. With our word set and labeled data we were ready to begin training the classification system. The training portion consists of checking which words in the word set exist in each tweet, the tweet's labeled classification, and applying the specified algorithm - the code required for training is implemented by the TextBlob library. After we had obtained the training classification system, we checked the accuracy by classifying each text entry from our test set and comparing it to the provided label. We then repeated this process for each of the three aforementioned algorithms.

Algorithm	**Accuracy** (0-1)
Decision Tree	.6654
Naive-Bayes	.7238
Logistic Regression	.8864

Table 2: Text Classification Accuracy by Alternate Machine Learning Algorithms

Note: The accuracy metric refers to the normalized amount of labeled text entries that the system accurately predicted divided by the total number of labeled text entries tested.

Logistic Regression showed the highest accuracy by a significant margin according to Table 2. This supports the theoretical finding (Zhu 2007) that Logistic Regression performs better than Naive-Bayes for binary classification when supplied with ample training data. Therefore, we selected the system using Logistic Regression as our political classification model and proceeded to use it in our processing pipeline for all tweets.

Political Classification Notes

The training data we selected implies that the classification system is specific to this data set, it is unlikely to produce accurate results for tweets produced outside of the 2012 campaign season. The process of creating labeled data could be easily recreated to build a political classification system for a different time period of Twitter data.

Our resources restricted us to a classification system that could be trained on a CPU with 16GB of RAM, through trial-and-error testing we determined that this equated to approx. 10,000 data points to be used for training for the Logistic Regression based classifier. With access to machines with more RAM, a more accurate classifier could be produced using the same methods.

Sentiment Analysis

In addition to politically classifying each tweet, we needed a sentiment score to gauge the emotions of the text. We accomplished this by using an out-of-the-box sentiment analysis implementation provided by the TextBlob library. This sentiment analysis model used by TextBlob is a Naive-Bayes based analyzer trained on a movie review corpus from IMDB. The analyzer produced one of three states for a given input: positive, negative, and neutral.

Storage

The final step of our processing pipeline was to store the original Tweet objects, the political classification, and the sentiment analysis scores in a SQL database. This provided an easy way to index, aggregate, and represent our data. We used Amazon's Relational Database Server (RDS) to store our data. The database consisted of 5 tables: User, Tweet, TweetSentiment, TweetPolitical, and Hashtag.

Sentiment and Political Classification

One of our core goals was to look at the relationship between emotion, political affiliation, and the change in political affiliation. The following chart shows two lines, the first is the percentage of tweets classified as negative per day, the second shows the change in political affiliation. The change in political affiliation is defined as the difference between the percentage of Republican tweets between the listed day and the prior day, since every tweet is defined in one of two classes this represents the change for both parties.

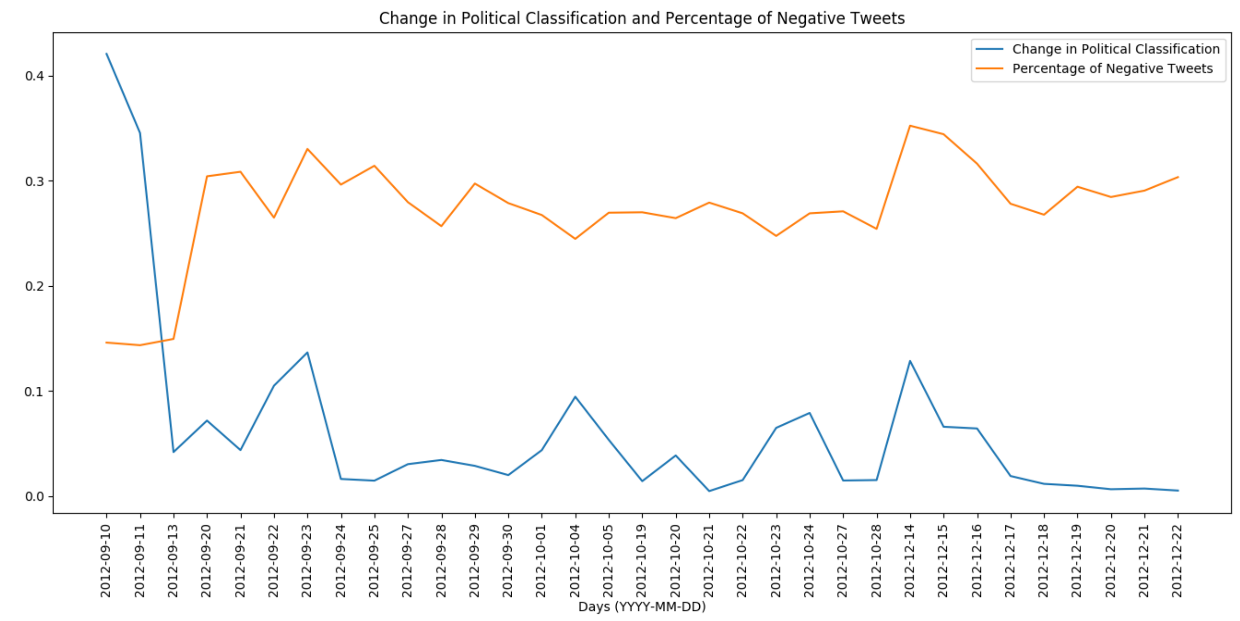


Figure 2: Change in Political Classification and Percentage of Negative Tweets over Time

Figure 2 shows a positive correlation between the two lines. September 9th, 2012 and December 14th in particular show corresponding spikes in both. This supports the hypothesis that Americans are more likely to change political affiliation when negative emotions are high. This goes hand-in-hand with the conventional wisdom that political races are won through attacking the opponent, this is evidenced by the high proportion of "attack ads" or negative advertisements meant to hurt the opponents image.

Political Classification Accuracy

Our key method of determining the accuracy of political classification takes the form of an Electoral College Vote Model (ECVM) based purely on the political classifications determined by our classification system. The ECVM picks out all tweets in which the user's location indicates they reside in a specific US state including Washington, D.C. The model then takes all classifications of tweets per state and determines a classification for the state by whichever political class held the majority of classifications. We then compared these state classifications to each state's actual electoral college vote. The results are as follows:

	Correctly Predicted	Total	Accuracy
Electoral College Votes	335	538	.6227
States	31	51	.6078

Table 3: ECVM Accuracy

As shown by Table 3 the accuracy of the ECVM is higher than simply choosing a random class for each state. This shows that our political classification system is accurate, albeit with a margin of error. It is interesting to note that each incorrect prediction was a state that voted Republican yet was classified as Democratic. This could be due to a skewed demographic that Twitter represents.

Future Work

This research provides a starting point into a complex subject with myriad variables and influencing factors. Analyzing more campaign seasons, particularly the 2016 campaign season (REFERENCE), and using the results to refine the classification would improve the accuracy of the political classification and therefore the results. More CPU resources would enable one to build a larger and more powerful political classification system which would further improve the accuracy.

The project accomplished the goal of using Twitter as an indicator of political leanings, yet Twitter does not represent an accurate demographic of the American public. Integrating more social media platforms, namely Facebook, would allow for more of the country to be represented. Additionally, determining which demographics are not well-represented in social media and incorporating an adjustment would improve the overall accuracy. Analyzing social media posts provides an excellent insight into a younger demographic’s political feelings, but is not particularly useful for tackling the entire American voting population.

References

Zhu, X. (2007). Text Categorization with Logistic Regression. *Text Categorization with Logistic Regression*. Retrieved May 1, 2017.

Yaniv Altshuler, Wei Pan, Alex (Sandy) Pentland (2012) Trends Prediction Using Social Diffusion Models [PDF]. *MIT Media Lab.* Retrieved from http://web.media.mit.edu/~yanival/SBP-Behavior-shaping.pdf

Jungherr, A. (2015). Twitter as Political Communication Space: Publics, Prominent Users,
and Politicians. *Analyzing Political Communication with Digital Trace Data Contributions to Political Science,* 69-106. doi:10.1007/978-3-319-20319-5_4

Ringsquandl, M., & Petković, D. (2013). *Analyzing Political Sentiment on Twitter* [Scholarly project]. In *University of Applied Sciences Rosenheim.*

"Twitter 2012 Presidential Election." *The Datahub.* Kingmolnar, 08 July 2015. Web. 1 Apr. 2017. <https://datahub.io/dataset/twitter-2012-presidential-election>.

Littman, Justin; Wrubel, Laura; Kerchner, Daniel, 2016, "2016 United States Presidential Election Tweet Ids", doi:10.7910/DVN/PDI7IN, Harvard Dataverse, V3

Appendix

Database Schema

User

Field	**Type**
ID	VARCHAR(100)
user_id	INT
retweet_count	INT
tweet_text	VARCHAR(100)
created_at	DATETIME

Tweet

Field	**Type**
ID	VARCHAR(100)
Follower_count	INT
Friends_count	INT
Statuses_count	INT
Screen_name	VARCHAR(100)
Name	VARCHAR(100)
Verified	BOOLEAN
Location	VARCHAR(250)

Hashtag

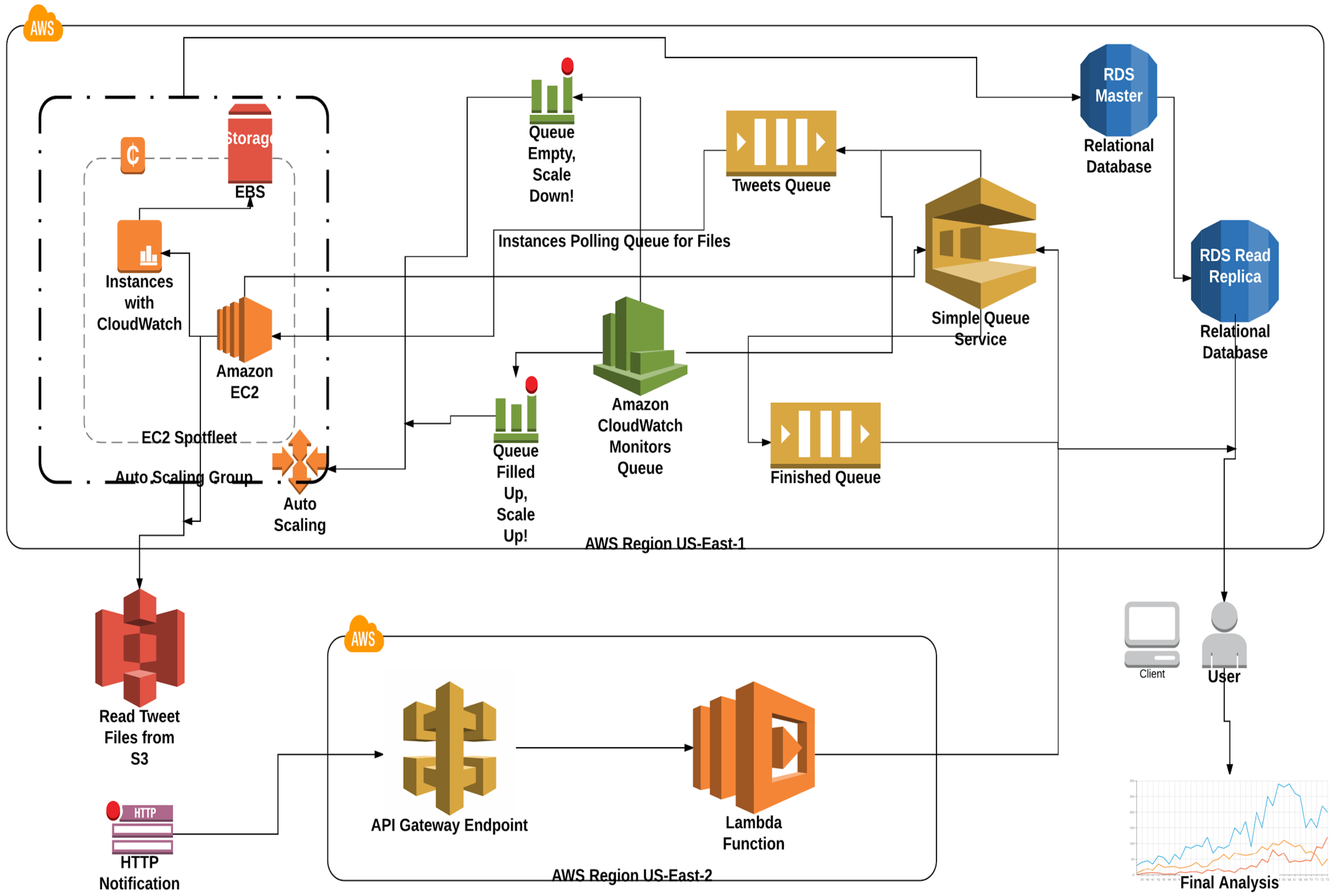
Field	**Type**
tweet_id	VARCHAR(100)
hashtag	VARCHAR(150)

Tweet Political

Field	**Type**
tweet_id	VARCHAR(100)
democrat _prob	DECIMAL(6, 5)
republican_prob	DECIMAL(6, 5)
thirdd_prob	DECIMAL(6, 5)
classification	VARCHAR(100)

Tweet Sentiment

Field	**Type**
tweet_id	VARCHAR(100)
polarity	DECIMAL(6, 5)
classification	VARCHAR(100)


AWS
Storage
EBS
Instances with CloudWatch
Amazon EC2
EC2 Spotfleet
Auto Scaling Group
Auto Scaling
Queue Empty, Scale Down!
Instances Polling Queue for Files
Queue Filled Up, Scale Up!
Amazon CloudWatch Monitors Queue
Tweets Queue
Finished Queue
Simple Queue Service
RDS Master
Relational Database
RDS Read Replica
Relational Database
AWS Region US-East-1
Read Tweet Files from S3
HTTP
HTTP Notification
AWS
API Gateway Endpoint
Lambda Function
AWS Region US-East-2
Client
User
Final Analysis