W205 Final Project
Tingwen Bao, Jay Cordes, Alex Jamar, Michelle Liu

Correlating Poll Results with Twitter on Obamacare

Background

Since it was passed in March 2010, Obamacare has remained a controversial topic. Public opinion has remained deeply divided on this health reform law even now, 7 years after it was introduced. In this project, we would like to develop a data infrastructure to hold both ACA poll results from Kaiser and Twitter posts, and analyze the correlation between the result of public polls and sentiment on Twitter. The data infrastructure will also be updated in batch so that people can track the trends in the future.

There are several major sources that conduct polls on Obamacare regularly. We chose poll results tracked by KFF.org (Kaiser Family Foundation) because the results could be easily downloaded with relatively good granularity. In addition, since KFF.org is also doing a lot of other healthcare related polls regularly, choosing it will allow this tool to be further used to track other popular healthcare topics.

We chose to compare poll results with Twitter because it is one of the leading social networks. Twitter is an online news and social networking platform where users globally post and interact with messages. Created in 2006, Twitter has since gained worldwide popularity and has more than 319 million users as of 2016. As one of the leading social networks, Twitter proved to be the largest source of breaking news on the day of the 2016 U.S. presidential election. Twitter claims that the real-time, public orientation of its social network makes it a reliable barometer of the public's constantly changing moods and interests.

Goals

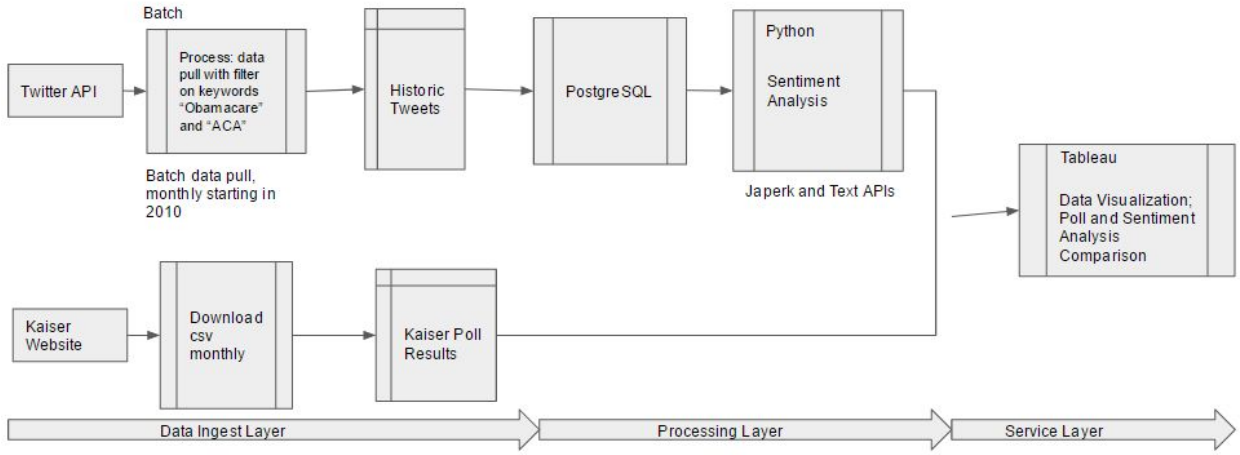
Both Twitter posts and the Kaiser health tracking poll allow us to explore the relative popularity of certain topics in healthcare. This project aims to find the correlation between ACA poll results and its public sentiment on Twitter over time. We chose to use Obamacare, a controversial topic for the past few years, to build a database and analytical infrastructure as an example to illustrate the findings. Since Twitter has emerged as a real-time barometer of public opinion, sentiment analysis on tweets regarding Obamacare will indicate the popularity of the law at a

given time. The goal of this project is to see if there is a correlation between the poll results and the related tweets.

The Process

Our plan is to gather historical tweets about Obamacare and ACA since 2010 and determine the sentiment of each tweet by performing sentiment analysis and assigning them a score (positive, negative, neutral). In order to categorize tweets, we plan to use a list of keywords of hashtags (Obamacare and ACA) and collect only tweets that are of interest to us.

Architecture Graph



Twitter Data

The team's original plan was to collect Twitter data from the Twitter streaming API Tweepy. However, access to the full Twitter history is limited to their commercial product Gnip. The Twitter search API only searches through the last seven days of historical tweets.

Jefferson Henrique's Python library, GetOldTweets, was adopted in the project to retrieve historical tweets. The library calls through a JSON provider and was able to bypass the time constraint limitations of Twitter API and retrieve old tweets. A few changes were made to both the Python wrapper script and the library to search for tweets in English about Obamacare or ACA since its introduction in 2010. The Python wrapper takes in a start date and an end date, and collects 1000 tweets per day every 7 days.

The process was run on an Amazon Web Services (AWS) Elastic Cloud Compute (EC2) virtual machine on the Berkeley AMI. The files were stored on the AWS instance, and held more than 400,000 Tweets totaling 90 MB of CSV data.

The Database

Once we had the Twitter data, we needed to store it in a way that made it easily accessible. We opted to use a PostgreSQL database because of its simplicity to use and maintain and the fact that our language of choice, Python, had a convenient driver (psycopg2) which gave us access to all of the functionality we needed.

A python script “dbscript.py” was created which made a new PostgreSQL table called Tweets (if it didn’t already exist) and took in a command line argument of a filename and imported that file into the table. Having a self-contained script like this with limited scope and well-defined purpose allowed us to avoid overly complex or large code files that could be difficult to maintain later.

The data structure of the table is as follows…

Tweets

Id	BIGSERIAL (PK)	Surrogate ID
Tweet_id	BIGINT	ID from Twitter
Tweet	VARCHAR(250)	Tweet text
User_handle	VARCHAR(25)	User name
Mentions	VARCHAR(250)	Users mentioned
Hashtags	VARCHAR(250)	Hashtags mentioned
Retweets	INTEGER	Number of times retweeted
Favorites	INTEGER	Number of times favorited
Permalink	VARCHAR(250)	URL for the tweet
Japerk	INTEGER	Sentiment analysis score
TextAnalysis	INTEGER	Sentiment analysis score
SentimentAPI	INTEGER	Sentiment analysis score
Created_At	TIMESTAMP	When the tweet was written
Inserted_At	TIMESTAMP	When the tweets was stored in the database

Even though we had a Tweet ID we could have used as a primary key, we chose to generate our own ID instead. This would maintain uniqueness and would keep us out of trouble even if tweets were accidentally imported twice. In general, we think it’s good practice to have a meaningless surrogate primary key. We also added indexes to the tweet and created_at fields to speed up query times later. The bottleneck in our process was going to be in the sentiment analysis piece, not the data import, so the slight overhead of the indexes was considered to be worth it.

The ID field had a datatype of “bigserial”, which is an auto-incrementing integer and the inserted_at field had a default of “now()”. These automatically populating fields would simplify things since we wouldn’t need to even mention them in the data import SQL queries and they would track what they were supposed to without any risk of error.

Something else worth mentioning is that the code script tries to create the finalprojecttweets database and the tweets table and handles the exceptions if they already exist. It also skips the use of transactions, since no commands are bundled together and there was no concern that an error would leave a task half-completed. When errors do occur during the import, the script prints the exception info to the screen and continues on to the next record. Our goal was to create relatively robust piece of code that could reliably get through a large number of records. In some cases, some data may be lost (for example, truncated to 250 characters), but for our purposes, the resulting table was useful and helped facilitate the later sentiment analysis and data visualization steps.

Sentiment Analysis

With our database and tweets in place, the next step in our project involved conducting sentiment analysis on our tweets. After reviewing the programming libraries and APIs available for sentiment analysis, we elected to use two different APIs and compare the results from each. We knew going in that sentiment analysis is notoriously difficult, so we chose two APIs to see if one performed better than the other.

Initially, we attempted to conduct sentiment analysis on all of our tweets, however, we realized mid trial run that it would have taken nearly 5 days to complete. As such, we decided instead to take a random sample of ~10% of our data (around 35,000 tweets) and conduct our sentiment analysis on only those.

To do so, we created a script Sentiment_Functions.py, which first randomly selects a certain number of tweets from our Tweets database, then creates a POST request to each of the sentiment analysis APIs containing the tweet text. After it parses the response string, it maps the values positive, neutral, and negative to 1, 0, and -1, respectively, then updates the appropriate rows in the database with those values.

Poll Data

Kaiser Health Tracking Poll:

Kaiser health tracking poll has public poll results on ACA since the month after it was enacted. The data is updated monthly and could be downloaded as csv files or pulled from links.

Kaiser's poll data is the best source of public poll result we could find that is relatively complete in time and could be downloaded as a flat file. However, it stores the spreadsheet in Google Doc which means we would not be able to wget and store it in PostgreSQL directly from instance. To accommodate this, we decide to simply store it as flat file and have Tableau open it directly without saving it to our database. In addition, the table format it provided is not suitable for us to map with our sentiment analysis table in PostgreSQL. So we reverse pivoted it so that it could map to PostgreSQL table by time.

Evaluating and Visualizing the Results

With the sentiment score assigned to the tweets and saved to the PostgreSQL table, we could start comparing the scores and seeing how they align with the public poll results. For this project, we used Tableau as the tool for the exploration, evaluation, and visualization.

Tableau, as a widely used visualization tool, is able to easily connect to a PostgreSQL database as well as directly import flat files. We chose it as our visualization tool because it easily met our needs of connecting multiple data source and refresh it regularly. It has a relatively strong visualization ability and is easily accessible by publishing the dashboard to a public/private server. The team could also control the level of access and interaction each single user could have. One of the drawbacks we found is that Tableau does not have really strong statistical functions. This makes comparison across multiple data sources/tables harder.

To evaluate and visualize the result, we created a dashboard with 3 tabs:

- **Background:** a general introduction of the background and objective of this project
- **Comparison:** visuals used to compare the 2 different sentiment analysis methods and understand their correlation with public poll data
- **Explore Tweets:** word cloud of top tweets by favorites and its trend by time with interactive filters

Some of the conclusions we draw from evaluation and visualizations are:

1. Using different sentiment analysis method yields different conclusion of Obamacare related tweets. Japerk's score indicates that the positive and negative tweets of Obamacare are close to a tie. Textanalysis's score indicates that it thinks there are about twice as many negative tweets as positive.
2. KPP poll shows public has a nearly even spread on their opinion on Obamacare. It is more positive at the start and then more negative, and most recently, almost even.
3. There is no significant correlation between sentiment analysis score and poll results
4. We only pulled a portion of tweets by time. It is surprised to see that the tweets with most favorites and retweets happened in 2016 rather than when it was first introduced. It was right before election when people mentioned Obamacare to express their political stands.

With future scale-up in mind, we had created a data extract for all the data sources we used. This will help significantly decrease the calculation time applying filters in real time. But since Tableau is not that good at dealing with really large data in nature, it might become slower if we extract all tweets in the future or if we want to display more details. To solve this, one option could be to create aggregated data tables in PostgreSQL and have Tableau only run on top of the aggregated tables.

Ideas for Future Improvement

In order to evolve the project for future improvement, we can think of a few ideas that we would incorporate if given more time. These ideas include improving the ways in acquiring the data, storing and processing it more efficiently, and overall, scaling up our solution.

- Replace all the tabs in the tweets with spaces before writing it to CSV. The original comma delimiter caused us trouble during the import, so we switched the file format to tab-delimited. This allowed us to import almost all of the tweets without issues, but there were still some lost records due to unexpected tabs. If we were to strip out all tab characters before saving the tweets to the text file in the first place, we'd be able to import all of them.
- We are currently pulling the same amount of tweets every month. But it could be that there are more tweets from when the ACA was first announced or before the election than other times. If people see more news/tweets about its negatives, they may react more negatively when they are asked during the poll. Or it is also possible that unless they just read a negative tweet yesterday, it won't have any effect on their opinion in poll. So instead of pulling the same amount of tweets every week, pulling in in proportion with the total number of related tweets might improve the analysis.
- When creating the PostgreSQL table, we should default the sentiment scores to null instead of 0. With our current setup, it's difficult to tell whether both of the sentiment scores for a given tweet are neutral (since 0 = neutral) or whether they just weren't in the subset of sampled tweets.
- Find a way around the technical difficulties to automate and import the polling data into a PostgreSQL table as well. The more data we have in easy-to-use database tables, the better. That would also allow us to do part of the stat calculation and aggregation in PostgreSQL rather than in Tableau.
- Instead of connecting directly to raw tweets data in Postgres, we could do some aggregation and calculation in the serving layer to reduce the processing time of Tableau.

Conclusion

Overall, we learned that text sentiment analysis is nowhere near perfect, and even though we did not find correlation between poll results and Twitter sentiment, we still learned a great deal

while implementing our end-to-end solution. The 3Vs that define big data are important challenges in our project in that the long term needs of the solution will have to control the sheer *volume* of the data coming in from Twitter, and to increase the *velocity* of doing sentiment analysis and processing of the data. Our solution is business oriented since it would be valuable for both Twitter and its end users to use the Twitter platform as a real-time barometer of public opinion. This application could be used to test any new sentiment analysis approaches that are developed and presumably, their conclusions will become closer and closer to poll results as progress is made. If we could demonstrate a correlation between Twitter sentiment scores and polling data, we would have provided evidence for a powerful new tool, essentially the ability to do real-time polling on any subject.

AWS Snapshot
Region - N. Virginia
Name - UCB_W205_Final_Project_Tweets
ID - ami-3f20bd29

References

http://kff.org/interactive/kaiser-health-tracking-poll-the-publics-views-on-the-aca/#?aRange=twoYear

https://github.com/Jefferson-Henrique/GetOldTweets-python

https://market.mashape.com/textanalysis/sentiment-analysis

https://market.mashape.com/japerk/text-processing