# CS143, Spring 2019
# Project2CS143 Report

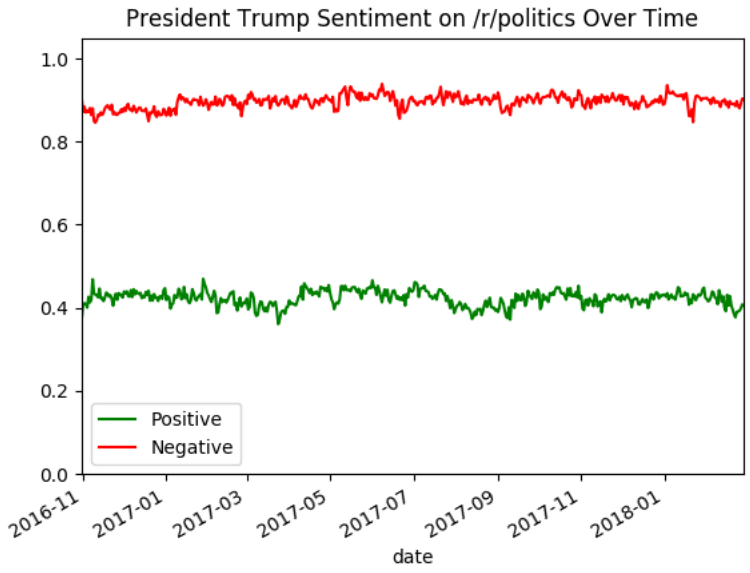
Justin Chang, Quentin Troung

06/1/2019

## 1 Graphs and Explanations

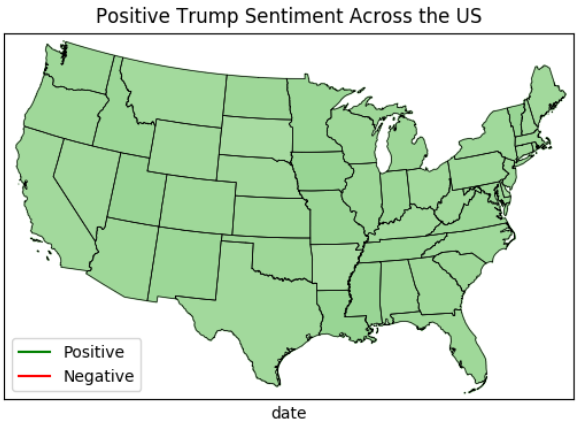
(a) Graph 1. Create a time series plot (by day) of positive and negative sentiment. This plot should contain two lines, one for positive and one for negative. It must have data as an X axis and the percentage of comments classified as each sentiment on the Y axis.

**Solution:** Below is the graph requested




(b) Graph 2 and 3. Create 2 maps of the United States: one for positive sentiment and one for negative sentiment. Color the states by the percentage.

**Solution:** Below are the 2 graphs requested



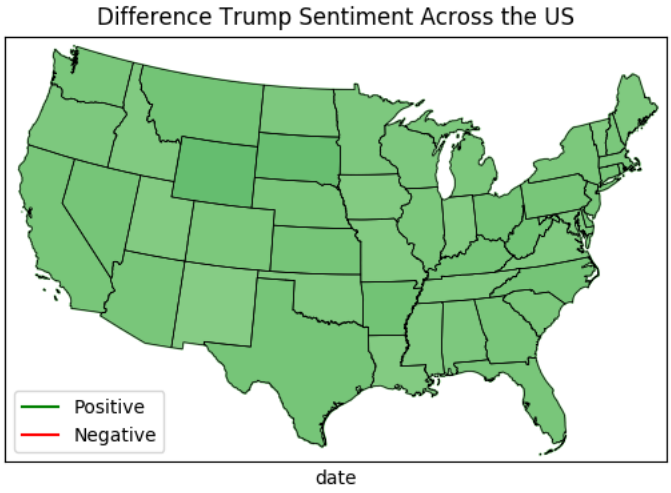


(c) Graph 4. Create a third map of the United States that computes the difference: Positive - Negative percentage.

**Solution:** Below is the graph requested

(d) Graph 5. Give a list of the top 10 positive stories (have the highest percentage of positive comments) and the top 10 negative stories (have the highest percentage of negative comments). This is easier to do in Spark.

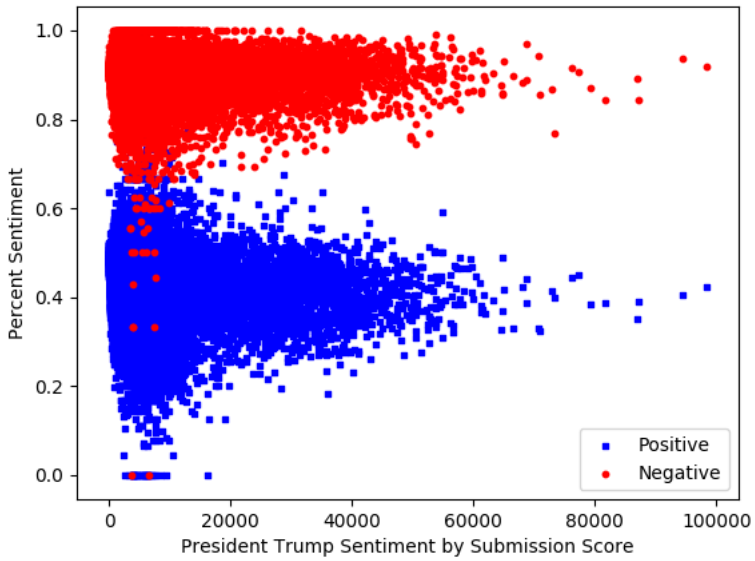
**Solution:** Below are the 2 csv lists requested. These were generated in spark.

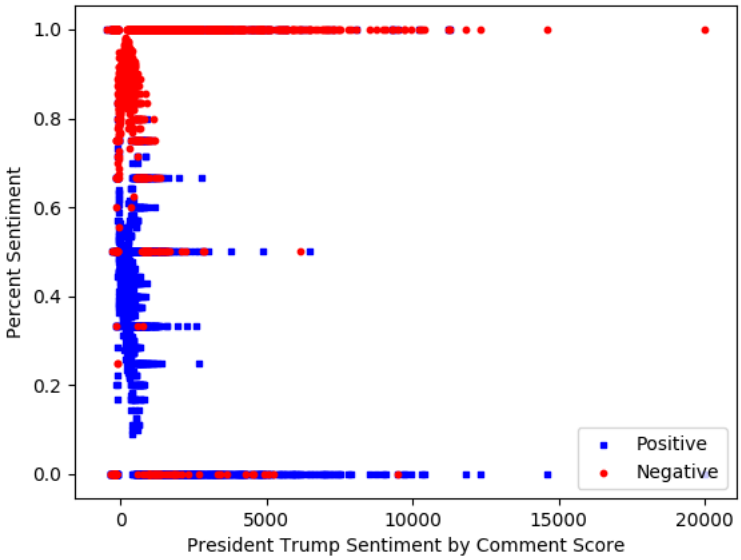



`fifth.png`

`sixth.png`

(e) Graph 6. Create TWO scatterplots where the X axis is the submission score, and a second where the X axis is the comment score, and the Y access is the percentage positive and negative. Use two different colors for positive and negative. This allows us to determine if submission score, or comment score can be used as a feature. **Solution:** Below are the 2 scatter plots requested






**Paragraph Explanation of Observations:**

All of the plots produced above involved using all of the dataset after applying a 0.25 threshold in our model creation in task 7. It is worth noting that the sentiments seem to be more negative than positive when looking at the data in each plot that assumes supporting vs against trump. This makes sense since there are usually more liberals and democrats on reddit, so sentiment analysis from dataset indicated that there were more sentiments in the politics subreddit that are against trump. We can also see on the maps that there are way more people that are negative against trump than positive across the US in general, and the data from our other graphs indicate that the sentiments remain consistently in that favor.

## 2 Questions and Answers

(a) Part 1

**Solution:** Solution to Part 1

Functional Dependencies in labeled_data.csv:
Input_id − >labeldem
Input_id − >labelgop
Input_id − >labeldjt

(b) Part 2

**Solution:** Solution to Part 2

No the data is not fully normalized. We could decompose it further by moving all attributes relating to the author into another relation and using an author_id. Also, we could decompose it further by moving all attributes relating to the subreddit into another relation and only using the subreddit_id. I think the data collector stored the data like this to make it easier to perform statistics on the data (A OLAP point of view, this is better to do aggregation functions). It would be easier because we don't need to perform joins. Also, it may be that the data was made available in this format and so it was just collected as is. The data is also thus, going to take up a lot more space.

(c) Part 3

**Solution:** Solution to Part 3

Question 3:

Using explain on this line of code in reddit_model.py:

labeled_comments = labels.join(comments, comments.id == labels.Input_id).select(['Input_id', 'labeldem', 'labelgop', 'labeldjt', 'body']).explain()

Gave the output:

```
Gave the output:
== Physical Plan ==
*(2) Project [Input_id#170, labeldem#171, labelgop#172, labeldjt#173, body#4]
+- *(2) BroadcastHashJoin [Input_id#170], [id#14], Inner, BuildLeft
   :- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, false]))
   :  +- *(1) Filter isnotnull(Input_id#170)
   :     +- *(1) Sample 0.0, 0.1, false, -4372746341945787401
   :        +- *(1) FileScan parquet
   [Input_id#170,labeldem#171,labelgop#172,labeldjt#173] Batched: true, Format:
   Parquet, Location: InMemoryFileIndex[file:/media/sf_vm-shared/labels.parquet],
   PartitionFilters: [], PushedFilters: [], Rea dSchema:
   struct<Input_id:string,labeldem:int,labelgop:int,labeldjt:int>
   +- *(2) Filter isnotnull(id#14)
      +- *(2) Sample 0.0, 0.1, false, 7407911338482665907
         +- *(2) FileScan parquet [body#4,id#14] Batched: true, Format: Parquet,
         Location: InMemoryFileIndex[file:/media/sf_vm-shared/comments.parquet],
         PartitionFilters: [], PushedFilters: [], ReadSchema:
         struct<body:string,id:string>
-
```

The join algorithms used by spark seem to be a broadcast hash join. Hash joins are generally useful for equality joins on large tables because it computes the join condition based on a hash index. It seems like it is also joining on the hash keys denoted by the number next to the attribute (input_id is 170 and id is 14) and does an inner left join. It is also worth noting that we only used 10% of the data to sampling just to speed things up a bit. The project on the first line after the physical plan shows SQL select being used or in relational algebra terms, the project statement to select the following 5 columns.