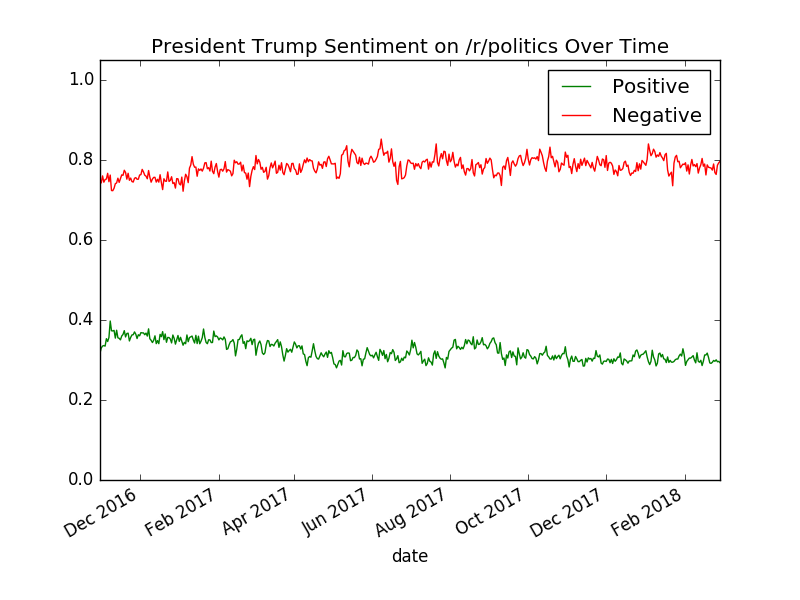
Andrew Ding, Arpit Jasapara, Brittany Hsu, Caleb Chau

204805602, 504742401, 904765400, 504748356

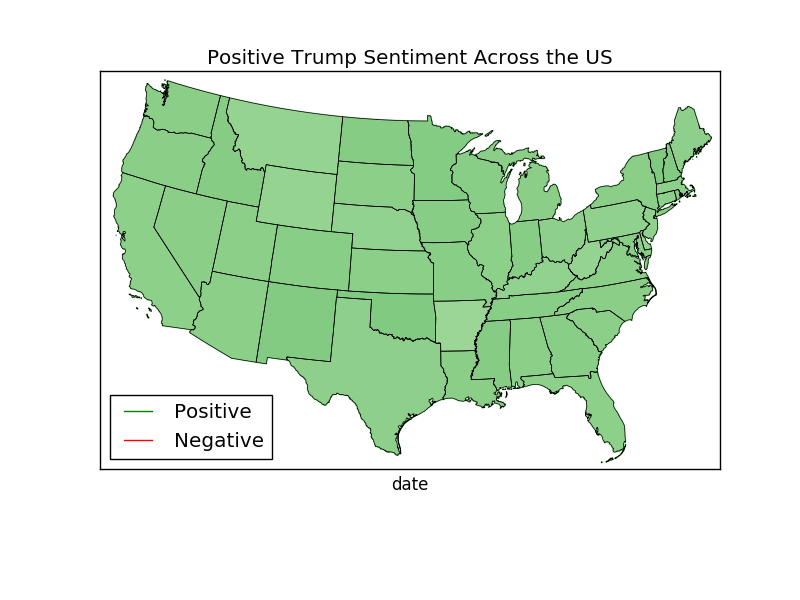
Project 2b

/r/politics thoughts on President Trump

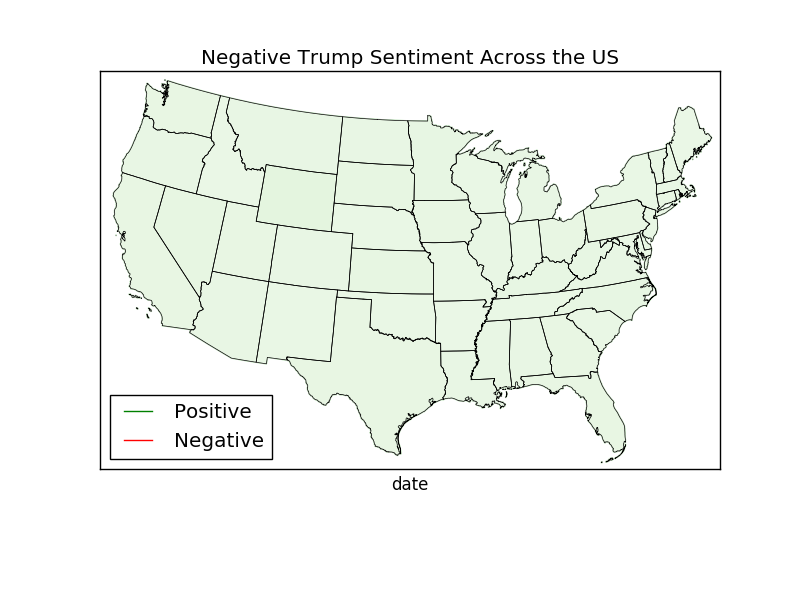
1. Create a time series plot (by day) of positive and negative sentiment. This plot should contain two lines, one for positive and one for negative. It must have data as an X axis and the percentage of comments classified as each sentiment on the Y axis.



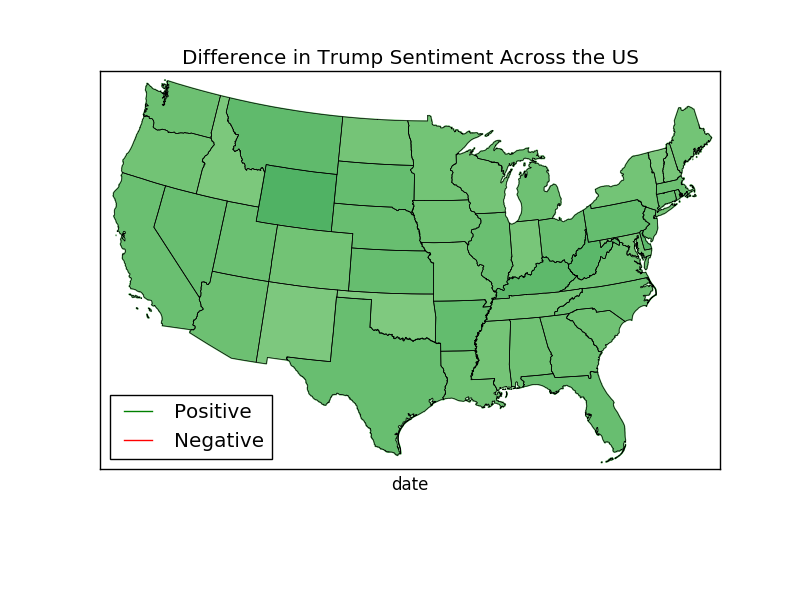
1. Create 2 maps of the United States: one for positive sentiment and one for negative sentiment. Color the states by the percentage.
   1. Positive



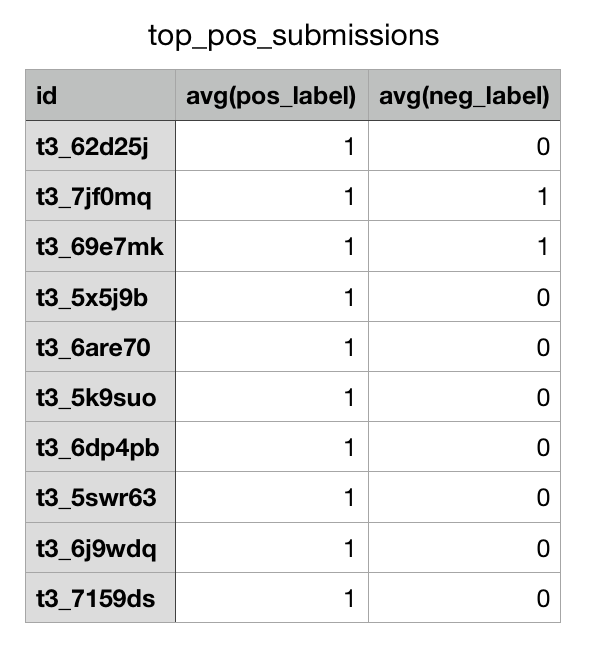
* 1. Negative



1. Create a third map of the United States that computes the difference: %Positive - %Negative.

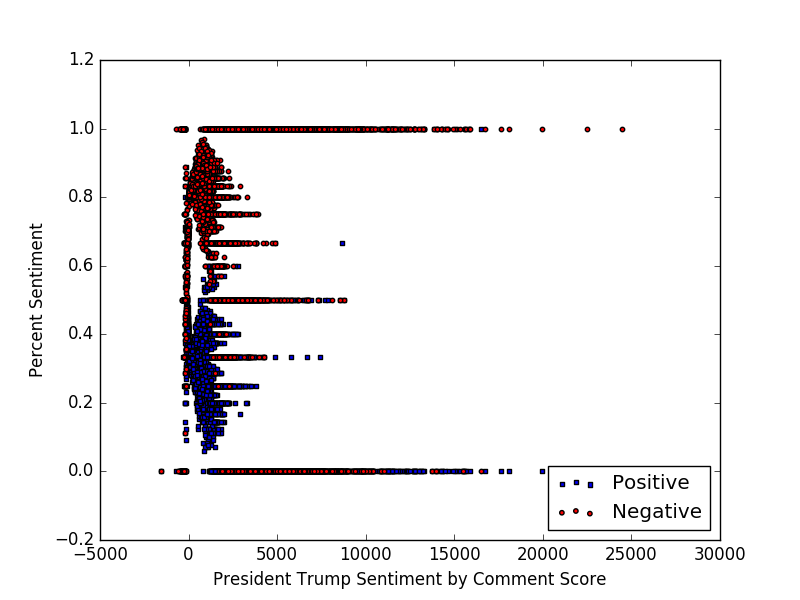


1. Give a list of the top 10 positive stories (have the highest percentage of positive comments) and the top 10 negative stories (have the highest percentage of negative comments).

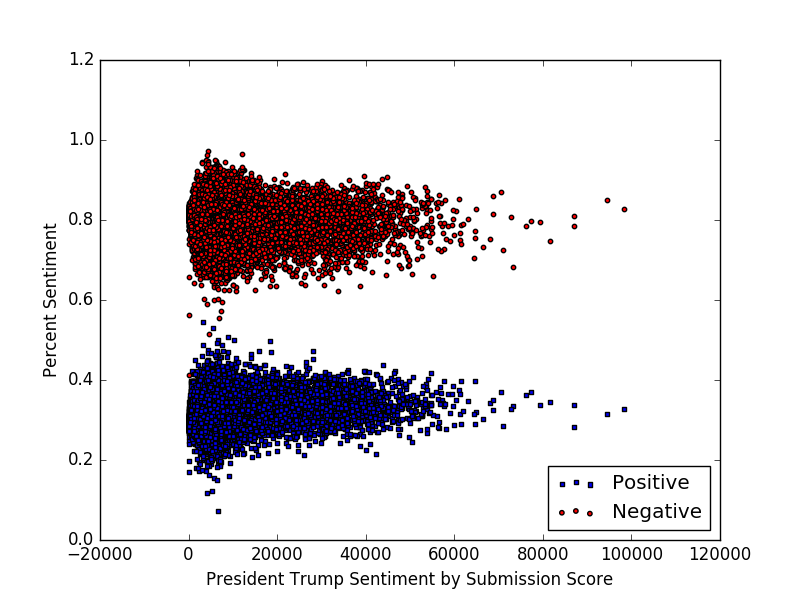




1. Create TWO scatterplots where the X axis is the submission score, and a second where the X axis is the comment score, and the Y access is the percentage positive and negative. Use two different colors for positive and negative.
   1. Comment



* 1. Submission



Paragraph Summary:

Overall, /r/politics does not seem to look favorably upon President Trump. Our first plot indicates that negative sentiment towards Trump was always high, and positive was fairly low. While it seems to fluctuate up and down plus/minus 0.1, it is interesting to see that the plot ended with the negative sentiment increasing from around 0.75 in December 2016 to 0.8 in February 2018. Similarly, the positive sentiment seemed to dip from around 0.35 to almost 0.3. The statewide graphs paint a similar picture. The negative sentiment by state is much higher averaging around 0.8 and the positive sentiment averaged around 0.3. The difference graph illustrates an expected picture where the difference is less in conservative, Republican-stronghold states such as Arkansas, and much more pronounced in liberal, Democratic states such as Washington. A few states were surprising such as Oklahoma having a large difference, indicating a high negative sentiment and a low positive sentiment. While there is slight variance by state, /r/politics still seems to hold an unfavorable view towards Trump. By story, there was a very split picture, but still seemed to hold a slightly unfavorable view towards Trump. Out of the top 10 positive stories, 2 of them strongly indicated negative sentiment towards Trump, whereas out of the top 10 negative stories, 1 strongly indicated positive sentiment and 3 indicated weak positive sentiment. This plot doesn’t reveal too much except the evident split. Lastly, by comment score and submission, /r/politics still held a relatively unfavorable view of Trump. While the negative sentiment was clearly stronger than positive sentiment in both graphs, comment score definitely seemed to be more related to negative sentiment. The highest comment scores are mostly red. Near the left, indicating a lower comment score, the blue positive sentiments coalesce, and as you move rightwards indicating a higher comment score, the red negative sentiments increase. So it seems that higher comment scores are correlated with a negative view of Trump. For submission score, for both positive and negative, most sentiments were clustered around the middle-left, not indicating any clear pattern between submission score and sentiment. Overall, in each of these plots, /r/politics clearly seems to have a negative view about President Trump.

Questions:

QUESTION 1: Take a look at labeled\_data.csv. Write the functional dependencies implied by the data.

Input\_id-->labeldem

Input\_id-->labelgop

Input\_id-->labeldjt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

QUESTION 2: Take a look at the schema for the comments dataframe. Forget BCNF and 3NF. Does the data frame look normalized? In other words, is the data frame free of redundancies that might affect insert/update integrity? If not, how would we decompose it? Why do you believe the collector of the data stored it in this way?

The data frame does not look normalized because it has redundant columns like subreddit and subreddit\_id. These are redundant because they already exist in the submissions table. If a comment is part of a submission, it obviously has the same subreddit. We would decompose it by getting rid of those two columns. We think the collector of the data stored it that way because they may have been looking at comments purely to analyze data without considering the actual submission.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

QUESTION 3: Pick one of the joins that you executed for this project. Rerun the join with .explain() attached to it. Include the output. What do you notice? Explain what Spark SQL is doing during the join. Which join algorithm does Spark seem to be using?

JOIN: joined\_comments = context.sql("select labels.Input\_id, labels.labeldem, labels.labelgop, labels.labeldjt, body from comments join labels on id=Input\_id").explain()

OUTPUT:

== Physical Plan ==

\*(2) Project [Input\_id#172, labeldem#173, labelgop#174, labeldjt#175, body#4]

+- \*(2) BroadcastHashJoin [id#14], [Input\_id#172], Inner, BuildRight

:- \*(2) Filter isnotnull(id#14)

: +- \*(2) Sample 0.0, 0.2, false, -2488598004471025273

: +- \*(2) FileScan parquet [body#4,id#14] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/home/cs143/data/comments.parquet], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<body:string,id:string>

+- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, false]))

+- \*(1) Filter isnotnull(Input\_id#172)

+- \*(1) Sample 0.0, 0.2, false, -8665745127085591957

+- \*(1) FileScan parquet [Input\_id#172,labeldem#173,labelgop#174,labeldjt#175] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/home/cs143/data/labels.parquet], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<Input\_id:string,labeldem:int,labelgop:int,labeldjt:int>

Based on the plan, we can see that Spark SQL is using a broadcast hash join to join the tables, which is more efficient when dealing with large tables (1). "BroadcastHashJoin [id#14], [Input\_id#172]" shows how it is joining the tables based on the two keys we provided. We can also see how Project shows the attributes that were "projected" (in relational algebra terms) or "selected" (in SQL terms).

(1): https://jaceklaskowski.gitbooks.io/mastering-spark-sql/spark-sql-joins-broadcast.html