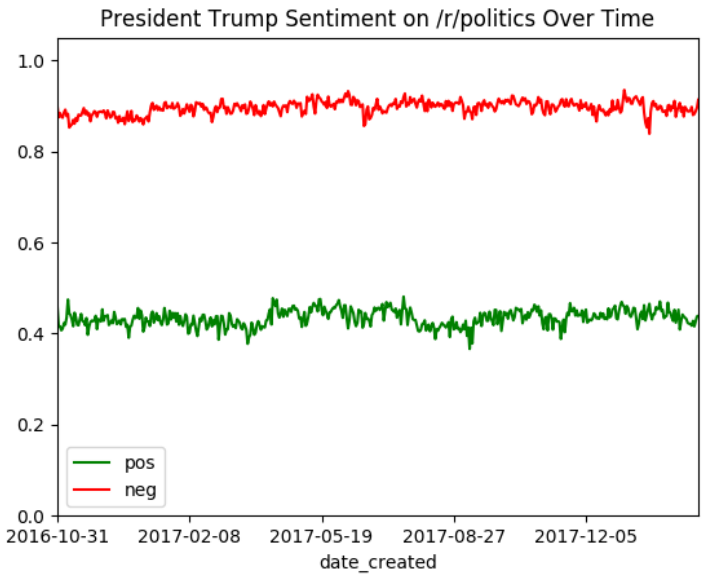
Project 2 Final Report

Arjun Kallapur and Tanmaya Hada

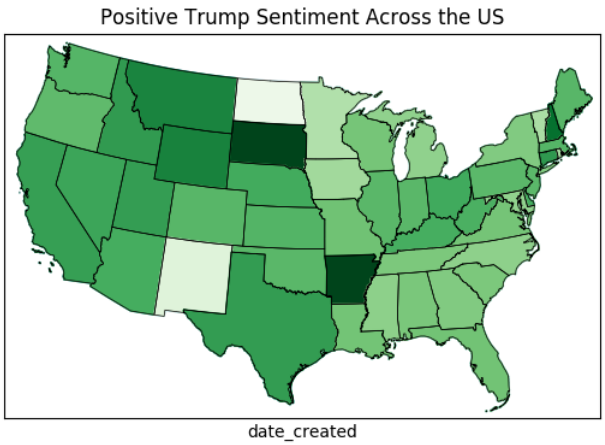
1. Plots

a. Time Series Plot of Positive and Negative Sentiment towards President Donald J. Trump



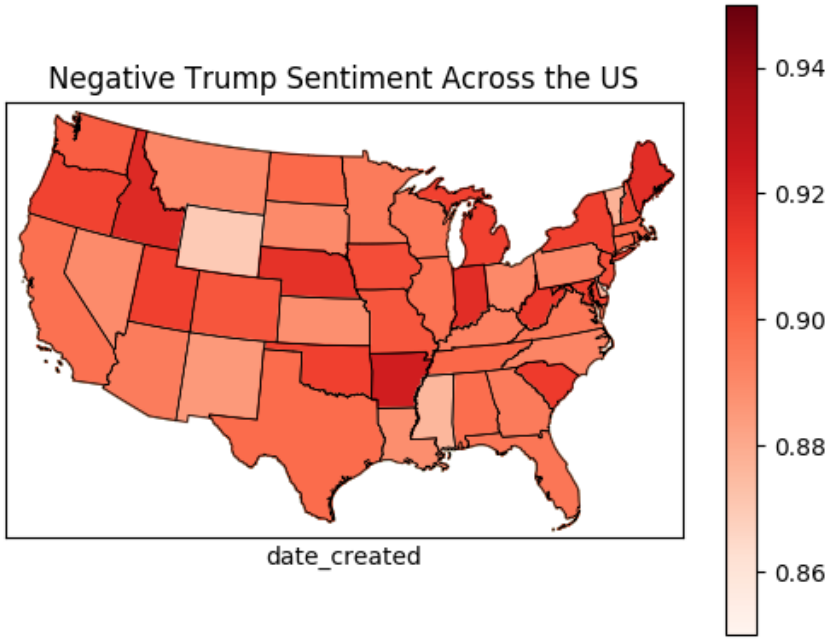
---

b. Map of the United States of America by Positive Sentiment towards President Donald J. Trump

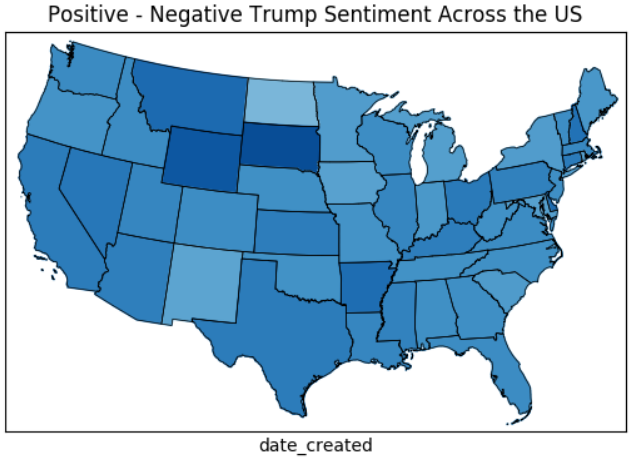


---

c. Map of the United States of America by Negative Sentiment towards President Donald J. Trump



d. Map of the United States of America by Difference in Positive and Negative Sentiment towards President Donald J. Trump



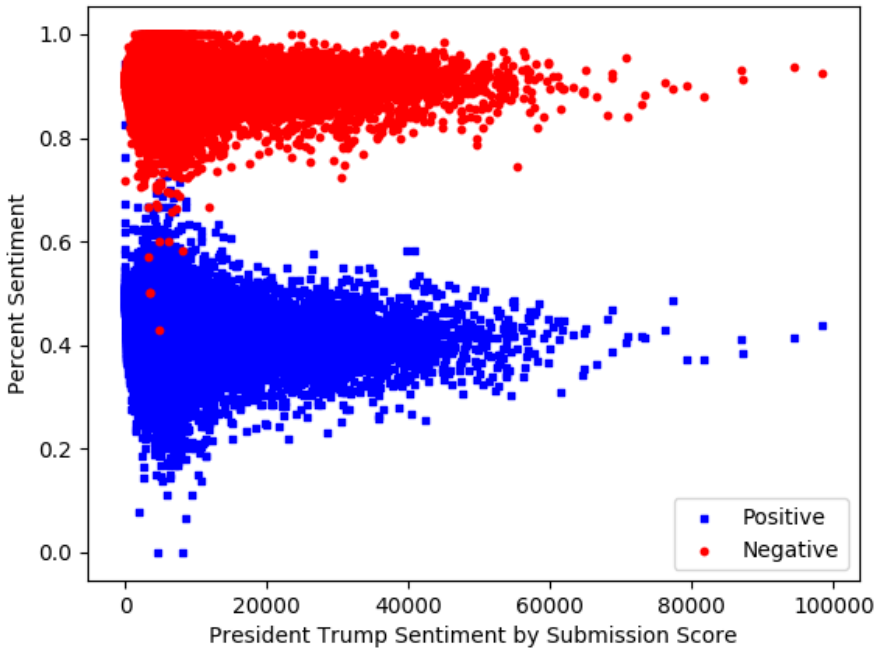
e. Top 10 Positive Stories in favor of President Donald J. Trump

1. Trump Team Knew Flynn Was Under Investigation Before He Came to White House
2. House Inquiry Turns Attention to Trump Campaign Worker With Russia Ties
3. National security adviser Flynn discussed sanctions with Russian ambassador, despite denials, officials say
4. Bob Corker Says Trump's Recklessness Threatens World War III
5. Watergate reporter on Russia: 'I've been saying for a while there's a coverup going on'
6. Bannon Is Subpoenaed in Mueller's Russia Investigation
7. Inside Trump's "anger and impatience" and his sudden decision to fire Comey
8. A Time Magazine with Trump on the cover hangs in his golf clubs. It's fake.
9. Trump had undisclosed hour-long meeting with Putin at G-20 summit
10. Trump Just Sold a $15.8 Million Condo to a Consultant Who Peddles Access to Powerful People

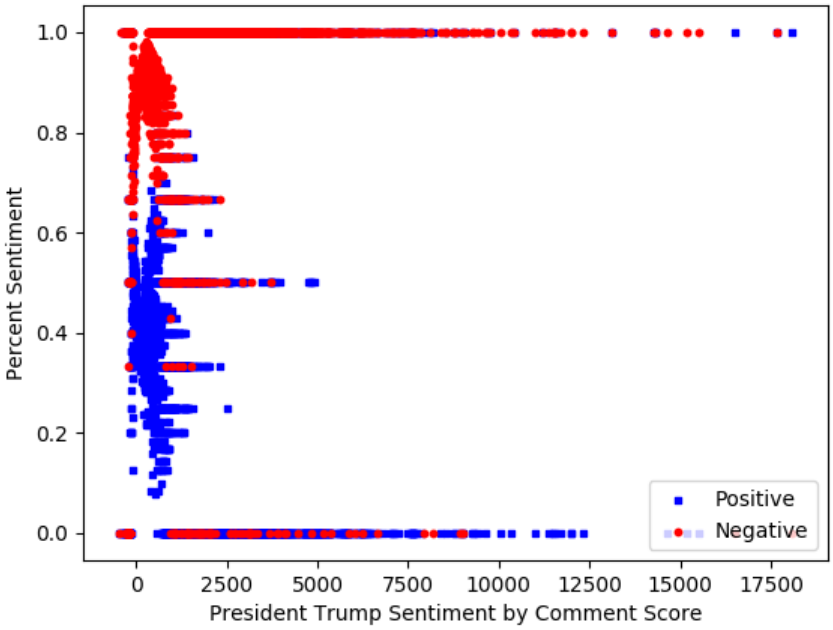
f. Top 10 Negative Stories Against President Donald J. Trump

1. WH doctor declares Trump in 'excellent' health, mentally fit for office
2. Poll: 82 percent think Trump should testify under oath to Mueller
3. President Trump pretended to know Japanese during prime minister's visit
4. Trump Said to Not Have Recordings of Conversations With Comey
5. Donald Trump recounts eating "the most beautiful piece of chocolate cake" with Xi Jinping while launching missiles heading to Iraq
6. Trump plans to declare that Iran nuclear deal is not in the national interest
7. Scarborough to Trump: 'your own lawyers think you are too stupid' to talk to Mueller
8. Neo-Nazi Website Praises Trump's Charlottesville Reaction: 'He Loves Us All'
9. Mattis goes where Trump won't: US-NATO bond 'unshakeable'
10. Michael Wolff's 'Fire and Fury' to Become TV Series

g. Submission Score vs Percentage Sentiment Scatterplot



h. Comment Score vs Percentage Sentiment Scatterplot



2. Analysis of the Above Plots

Overall, we can conclude that r/politics does not have a favorable opinion of President Trump. We note from Plot 1(a) that the negative sentiment expressed towards the President is almost twice the positive sentiment expressed towards him, in a time period that did not contain an election cycle. We note that there is a variation in positive and negative sentiments towards the President, but there is not a large variation state to state. The positive sentiments all lie within a range of (0.3, 0.45), and the negative sentiments lie within a range of (0.86, 0.94). Therefore, from the above plots it is concluded that the national sentiment is fairly uniform, and there is no large state to state variation. Perhaps Americans have more in common than was thought. Over time as well, the sentiment remained largely constant. There is no large shift in either positive or negative sentiment towards the President. By submission score and comment score, we see there is no overwhelming positive sentiment towards the President, but there are a large number of submissions and comments that are overwhelmingly negative towards the President.

3. Answers to Questions:

Question 1:

The following functional dependencies are implied by the data:

$$Input\_id \rightarrow labeldem$$
$$Input\_id \rightarrow labelgop$$
$$Input\_id \rightarrow labeldjt$$

Question 2:

The schema for the comments is not fully normalized. We note the existence of both $subreddit$ and $subreddit\_id$, even though $subreddit$ can be determined from $subreddit\_id$ and vice versa. We also note that $subreddit\_id$ is a foreign key, and if a subreddit is deleted, then the comment will still have a subreddit, which is a referential integrity issue. We also note that if the $subreddit\_id$ is changed, we will also have to update the $subreddit$ everywhere. At a scale as large as that of Reddit, this could prove to be costly.

We would decompose the comments in the following way, with two separate relations:

$$\{id, author, subreddit\_id, subreddit, stickied, score, retrieved\_on, \dots\},$$

$\{subreddit\_id, subreddit\}$

This removes the redundancy. However, note that Reddit persists with this "redundancy", likely with good reason. One possible reason could be that while a comment is being displayed, the name of the parent subreddit must be visible. In our "normalized" design, in order to display the name of the subreddit, a join would have to be computed, or a lookup would have to be performed. Once again, this is a costly operation, particularly at a large scale. Thus, the collectors of the data have kept in this "redundancy" in favor of a quick access to get the name of the subreddit a comment is on.

Question 3:

We attach the output for our explain below. This explain was calculated on the $JOIN$ carried out on the $comments$ and $labeled\_data$.

The query for the explain was the following:

$SELECT\ labels.Input\_id, labels.labeldem, labels.labelgop, labels.labeldjt, body$
$FROM\ comments$
$JOIN\ labels$
$ON\ id = Input\_id$

```
== Physical Plan ==
*(2) Project [Input_id#50, labeldem#51,
labelgop#52, labeldjt#53, body#4]
+- *(2) BroadcastHashJoin [id#14], [Input_id#50],
Inner, BuildRight
   :- *(2) Project [body#4, id#14]
   :  +- *(2) Filter isnotnull(id#14)
   :     +- *(2) FileScan parquet [body#4,id#14]
Batched: true, Format: Parquet, Location:
InMemoryFileIndex[file:/media/sf_vm-
shared/P2B/comments.parquet], PartitionFilters: [],
PushedFilters: [IsNotNull(id)], ReadSchema:
struct<body:string,id:string>
   +- BroadcastExchange
HashedRelationBroadcastMode(List(input[0, string,
true]))
```

```
      +- *(1) Project [Input_id#50, labeldem#51,
labelgop#52, labeldjt#53]
         +- *(1) Filter isnotnull(Input_id#50)
            +- *(1) FileScan parquet
[Input_id#50,labeldem#51,labelgop#52,labeldjt#53]
Batched: true, Format: Parquet, Location:
InMemoryFileIndex[file:/media/sf_vm-
shared/P2B/labels.parquet], PartitionFilters: [],
PushedFilters: [IsNotNull(Input_id)], ReadSchema:
struct<Input_id:string,labeldem:int,labelgop:int,la
beldjt:int>
```

We note the following about the above explain output:
We see keywords like project, filter, and FileScan in the output statement. Project represents the select statements we used in our query, filter is the where clause and FileScan is scanning the parquet file for the data. The instructions are evaluated in the nested order so the inner nested commands are done first. The join algorithm used is a BroadcastHashJoin which is an optimization of a Hash Join when one side of the data is below a certain threshold. We also see some additional information at the end which shows that batched processing is enabled and the file format it is reading from is parquet.

4. Design Decisions and Citations

The most important design decision to be pointed out is that we did not use all of the data. In Tasks 8 and 9, while we were joining the comments to the submissions, we sampled only 20% of the comments. This sped up the process greatly. Other design decisions for each task are detailed below:

1. Tasks 4 and 5:
   We referred to the User Defined Function documentation at the following link (https://docs.databricks.com/spark/latest/spark-sql/udf-python.html). Note for the wrapper function for sanitize, I referred to the following stack overflow post to use the extend() function on a list: (https://www.geeksforgeeks.org/append-extend-python/).
2. Task 6
   For count vectorizer, the following resource was used:
   (https://spark.apache.org/docs/latest/ml-

features.html#countvectorizer). In Task 6B, there is a function if() to replace the t3_ at the beginning of links. That was found by searching for replacing text in SQL and using the following link: (https://www.w3schools.com/sql/func_mysql_if.asp).

3. In Task 9, a UDF was used to extract the second element of a probability vector. we got this idea from high-level discussions with friends in the class. To avoid a join when fitting the model to the unseen data, the positive probabilities were calculated, then the column was renamed, then the negative model was run. This was posted on Piazza.
4. In Task 10, for the Top 10 post, we got the idea of using PySpark native functions from Piazza, and subsequently had high-level discussions with other students in the class.
5. Other sources used were the following:
   (https://stackoverflow.com/questions/2670871/mysql-group-results-by-day-using-timestamp) -> for grouping results by timestamp
   (https://www.w3schools.com/sql/func_sqlserver_replace.asp) -> For the replace function in SQL
   (https://stackoverflow.com/questions/7571635/fastest-way-to-check-if-a-value-exist-in-a-list) for fastest way to check if an element is in list of states
   Apart from this, many Piazza posts were used for inspiration and reference.