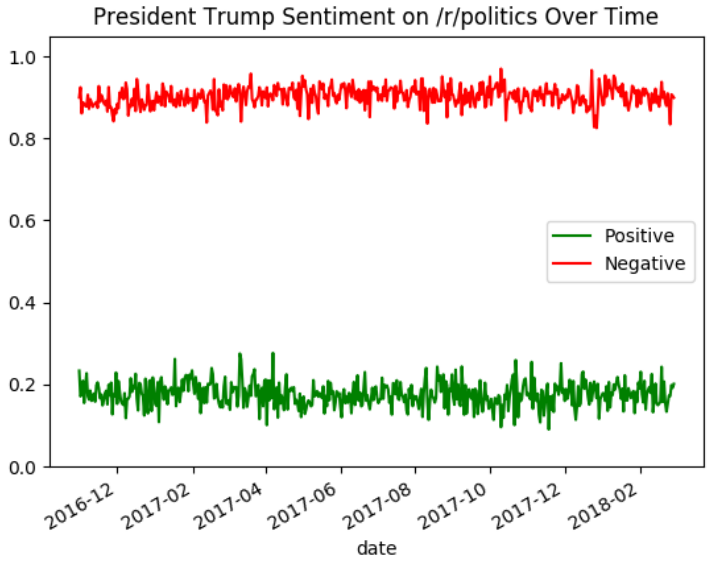
We sampled about 1% of the total comments, which took about 20 minutes (without training model) to generate all required csv files for the deliverable.

1. Time series plot (by day) of positive and negative sentiments.

Figure 1: Time series plot



2. 2 maps of the United States: one for positive sentiment and one for negative sentiment. Color the states by the percentage.
   Extra credit: we have successfully added Alaska and Hawaii to the map.

Figure 2: Map of positive sentiment

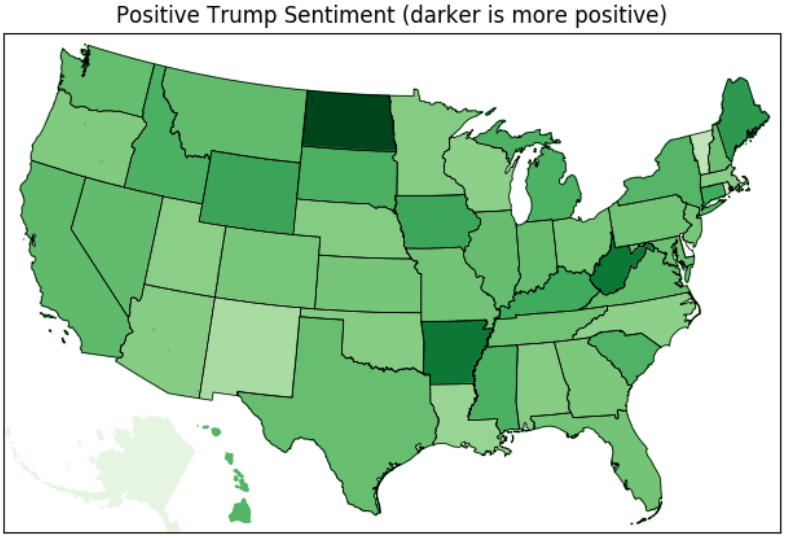
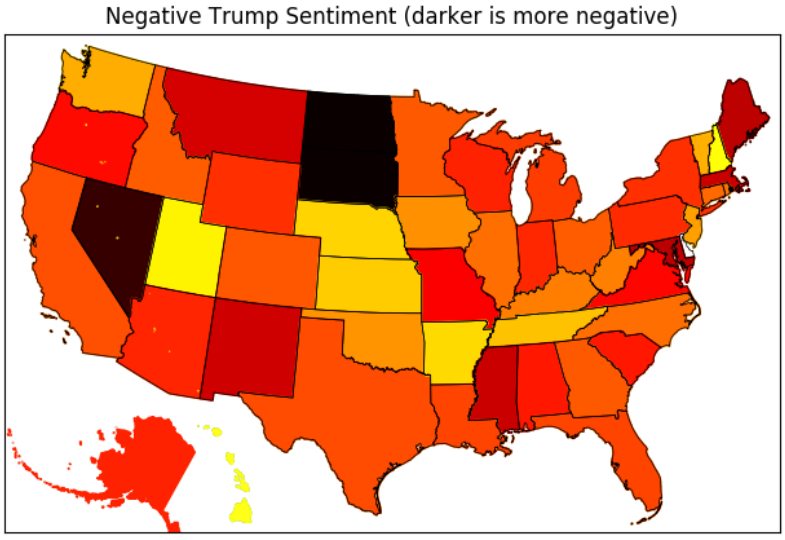
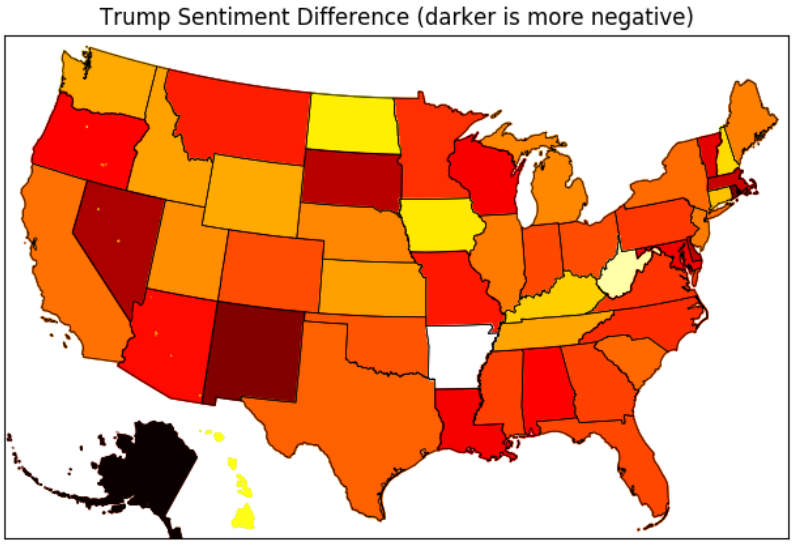


Figure 3: Map of negative sentiment



3. Map of United States that computes Positive percentage − Negative percentage



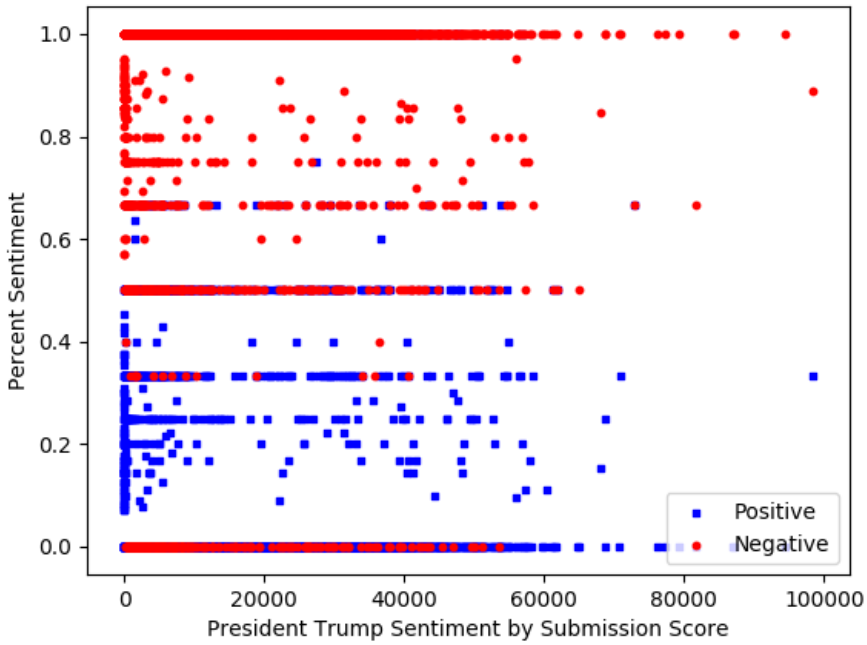
4. For the 1% of the data that we sampled, the list of "most positive" scores are:

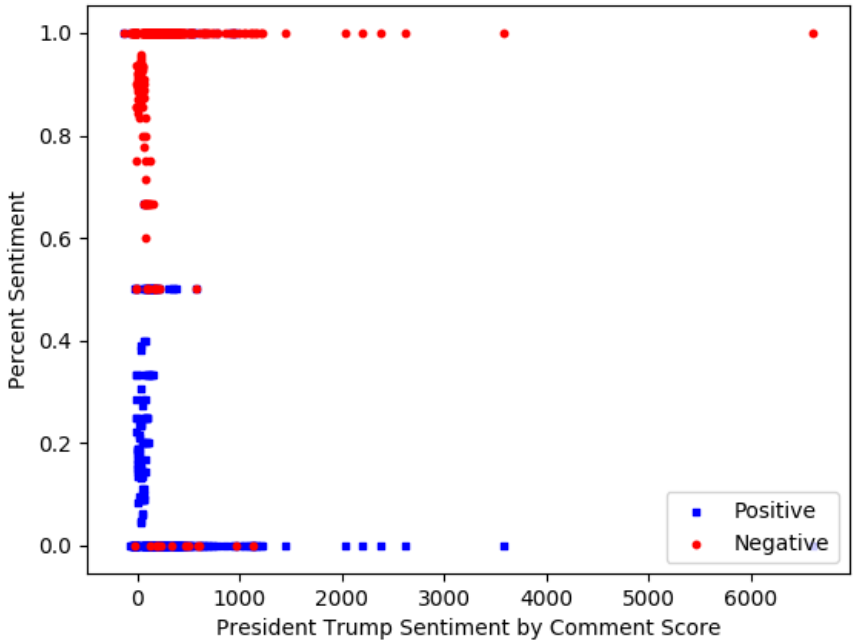
```
Trump's "approval poll" offers no negative options
One of Trump's first acts as president made it easier for the mentally ill to buy guns
Donald Trump boasts of 'plunging' health insurance stocks following executive order on Obamacare
Obama's top lawyer: Mueller has assembled best team in 'history'
Trump's lack of empathy about Puerto Rico is staggering
Trump to Be Barred From U.K. Parliament Over 'Racism and Sexism'
Trump blasts ex-aide Bannon, says he has lost his mind
Now Trump is an "independent" president? Give us a break
Thousands expected at march to impeach President Trump in downtown Los Angeles
Less than an hour after Florida school shooting, online communities devised a plan to spread conspiracies
Roger Stone Wants President Trump to Legalize Marijuana
```

The most negative stories are

```
Pelosi on Criticism from Dems: 'I Think I'm Worth the Trouble'
Discussion Megathread: The House votes on the AHCA
With tough races over, Bernie Sanders renews call for reforms at DNC
Discussion Megathread: The House votes on the AHCA
Warren: No confirmation hearings until ethics concerns addressed
Discussion Megathread: The House votes on the AHCA
Legal marijuana foes offer a compromise: Decriminalize it
House Oversight chairman wants US intelligence leaks surrounding Trump investigated
Palantir: the 'special ops' tech giant that wields as much real-world power as Google
Analysts: Trump Immigration Policies Could Spell Trouble for the Texas Economy
```

5. 2 scatterplots of submission score, comment score v.s. positive, negative comments.





6. NA.

7. NA.

8. Paragraph summarizing the findings.

    - r/politics mostly has negative comments on President Trump – over 80% of the comments are negative.
    - The sentiment certainly varies by state: as shown by the map, with some states in very deep shades and others in light shades.
    - Positive or negative sentiments does not seem to vary by time, or at least does not demonstrate a positive trend over time or negative trend over time. But Figure1 demonstrates large variation from day to day – as both curves are not smooth, and percentage of positive comments have relatively large variance compared to the percentage of negative comments over time.
    - In terms of comment score, the higher the comment score, the more likely the comment is negative shown by as we move to the right, the red dots are mostly on the y level of 1.0 and negative dots are on the y level of 0.0. For the comments with less scores, they attract there are are red/blue dots floating round 0.8/0.2, meaning the comments are less polar.
    - In terms of submission score, the ones with the most upvotes attract a lot of negative comments whereas the submissions with smaller scores, tend to attract more diverse sentiments – they are not almost exclusively negative.

Question 1. Functional dependencies:

```
Input_id -> labelgop
Input_id -> labeldem
Input_id -> labeldjt
```

Question 2. The data does not look normalized. The possible redundancies are:

- subreddit name gets repeated for each comment in that subreddit
- can_gild, author_flair_text, author_flair_css_classString, author_cakeday gets repeated for each user whenever he/she posts a comment

Possible decomposition:

```
user_details(userid, can_gild, author_flair_text, author_flair_css_classString, author_cake)
comments(comment_id, user_id, subreddit_id, ... other attributes...)
subreddiit(subreddit_id, name)
```

`comment.user_id` is a foreign key referencing `user_details.user_id` and `comments.subreddit_id` is a foreign key referencing `subreddit.subreddit_id`.
The author designed it this way because even though it contains redundancies, the information that we consider redundant may be very relevant in analysis of a comment and if we need to look at these features each time we analyze a comment, obtaining it through joining (with less redundancy) is going to be expensive.

Question 3 Pick one of the joins and run it with `explain`.

We choose to run `comments.join(labeled_data, comments.id == labeled_data.Input_id).explain()`. The output is the following:

```
== Physical Plan ==
*(2) BroadcastHashJoin [id#14], [Input_id#200], Inner, BuildRight
:- *(2) Project [id#14, created_utc#10L, body#4, author_flair_text#3, link_id#16, score#20L AS commentscore#176L]
:  +- *(2) Filter isnotnull(id#14)
:     +- *(2) FileScan parquet [author_flair_text#3,body#4,created_utc#10L,id#14,link_id#16,score#20L] Batched: tru\
e, Format: Parquet, Location: InMemoryFileIndex[file:/home/cs143/data/comments-minimal.parquet], PartitionFilters: \
[], PushedFilters: [IsNotNull(id)], ReadSchema: struct<author_flair_text:string,body:string,created_utc:bigint,id:s\
tring,link_id:string,score:big...
+- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
   +- *(1) Project [Input_id#200, labeldem#201, labelgop#202, labeldjt#203]
      +- *(1) Filter isnotnull(Input_id#200)
         +- *(1) FileScan csv [Input_id#200,labeldem#201,labelgop#202,labeldjt#203] Batched: false, Format: CSV, Lo\
cation: InMemoryFileIndex[file:/home/cs143/data/labeled_data.csv], PartitionFilters: [], PushedFilters: [IsNotNull(\
Input_id)], ReadSchema: struct<Input_id:string,labeldem:int,labelgop:int,labeldjt:int>
```

I first noticed that the plan is presented as a tree, the plan executes starting from the leaves (the inner indents) to the root (the top-level indent).

Each node performs a `FileScan` to read in the data and `Filter` out the rows where the join key is null, project to the variables that is going to show up into the output. After doing so for each worker, they BoroadcastExchange information so that Spark can better plan and estimate further plan. Finally, it performs the join in the node that handles comments data.

It seems to be using broadcast hash join – Spark SQL choose to do broadcast hash join when one of the data is smaller than a threshold. In this case, the size of `labeled_data.csv`, which is around 30634 bytes, much smaller than 10MB, that hinted it to use a *broadcast* hash join. Spark "broadcasts" (by having a read-only copy in memory) to every worker node, compared to shipping a copy for each task and it is more efficient in an Spark application where worker nodes communicates through network (since the data gets shipped only once). Among all other types of joins that Spark support (for example, sort-merge join, shuffled hash join, etc), it is the most efficient, however, since we are running this project on a single machine, the performance boost may not be obvious.`BuildRight` means that the right table – `labeled_data.csv` is the one that is used to create the hash table.