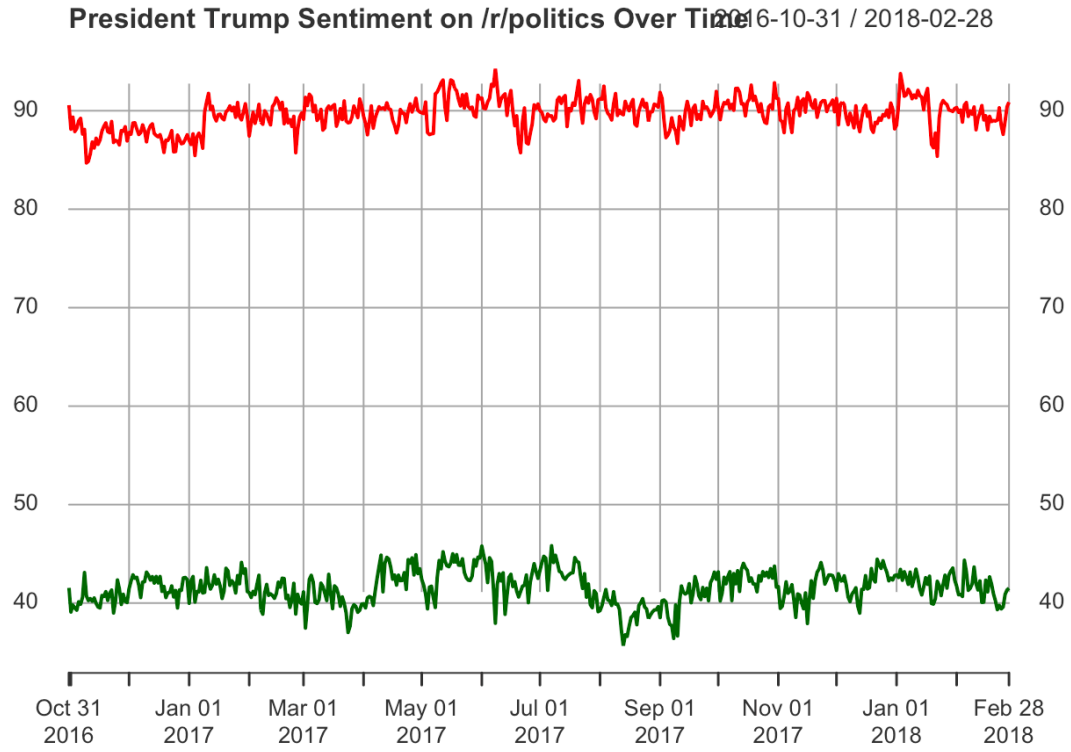
Project 2B Report

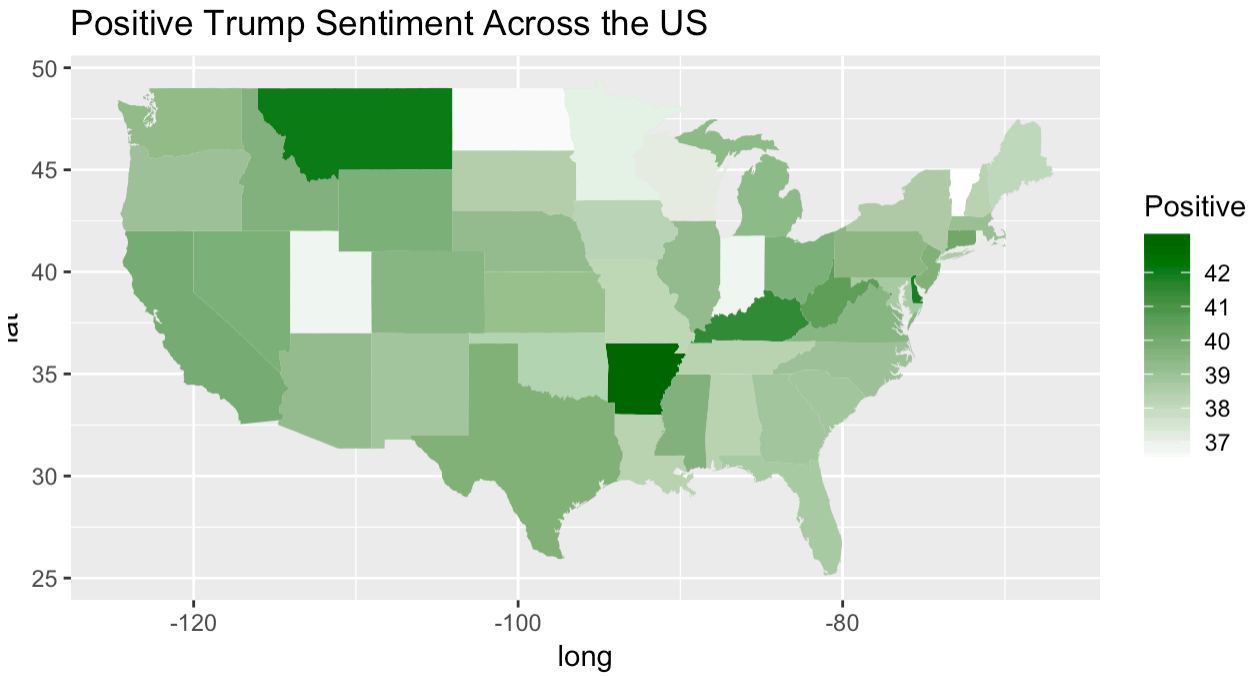
Ankith Uppunda
Sheldon Dong

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.

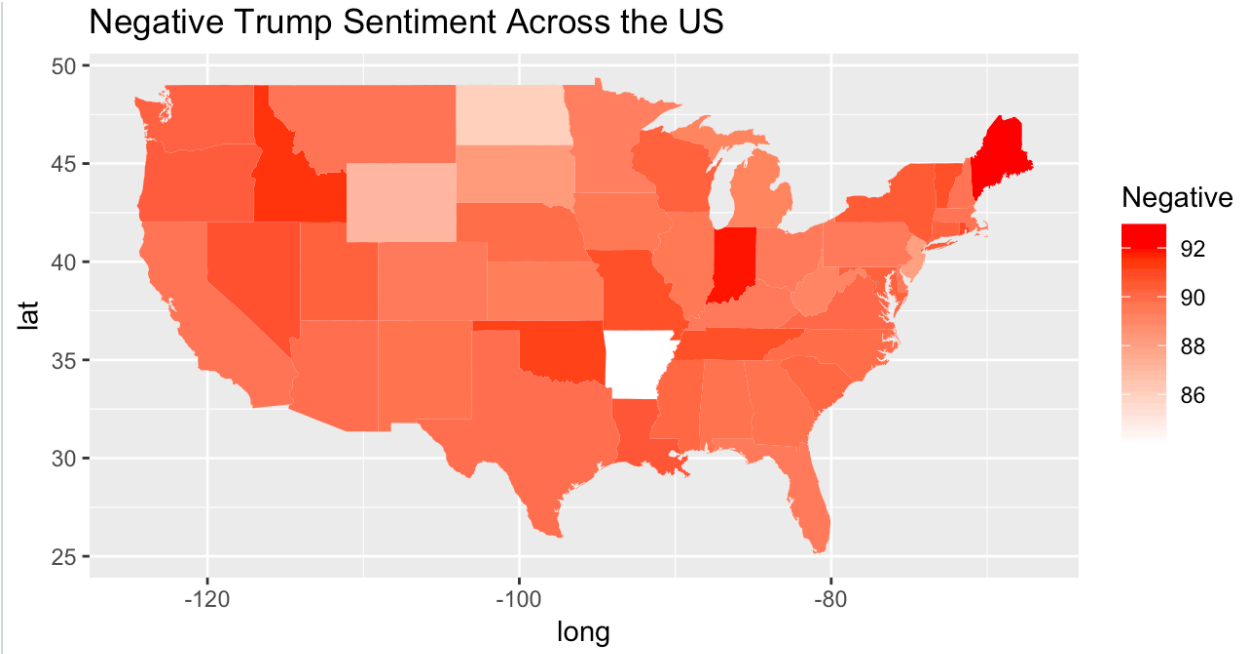


2. Create 2 maps of the United States: one for positive sentiment and one for negative sentiment. Color the states by the percentage.

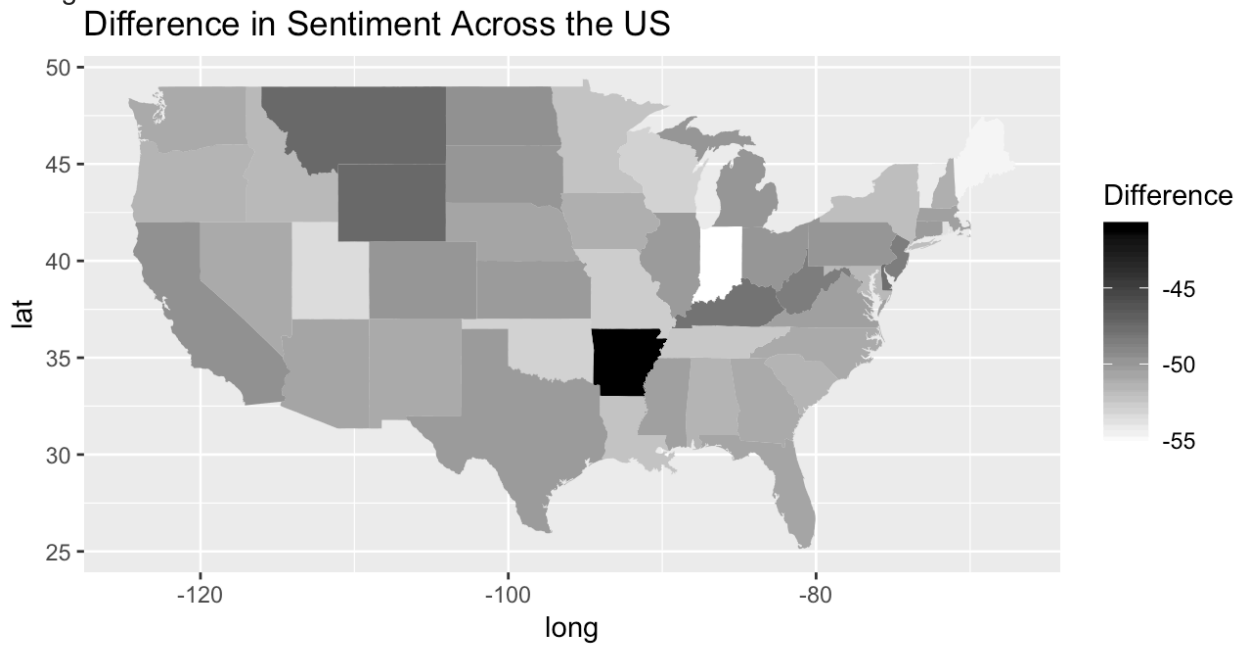
For Positive:



For Negative:



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



4. 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.

Top 10 Positive stories:

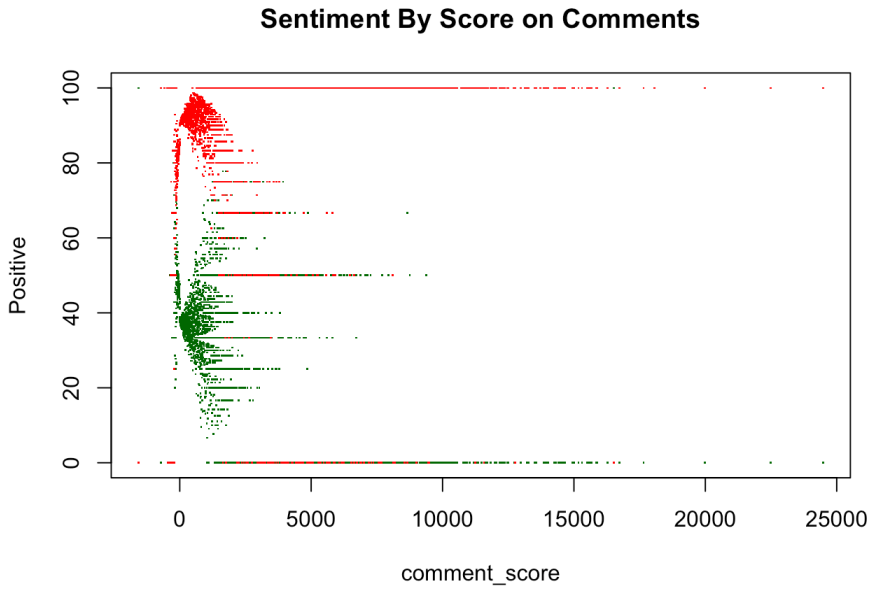
title	poscount
The Daily Show: Ivanka's Not a Feminist, She's a Trump	100.0
Old Mexico Lives On	100.0
Donald Trump set to make his 'biggest speech yet' in joint address to Congress	100.0
Comey Reportedly Shaped the Clinton Probe Around Russian Intelligence That He Knew Was Fake	100.0
My single vote is a bigger share of the total vote than you think	100.0
Iran Races to Clinch Oil Deals Before Donald Trump Takes Office	100.0
US employers add strong 211,000 jobs; unemployment 4.4 pct.	100.0
Fake News and the Founding Fathers	100.0
Somalis mistreated during US deportation effort, lawsuit alleges	100.0
The key to the Trump-Russia scandal? Follow the data	100.0

Top 10 Negative stories:

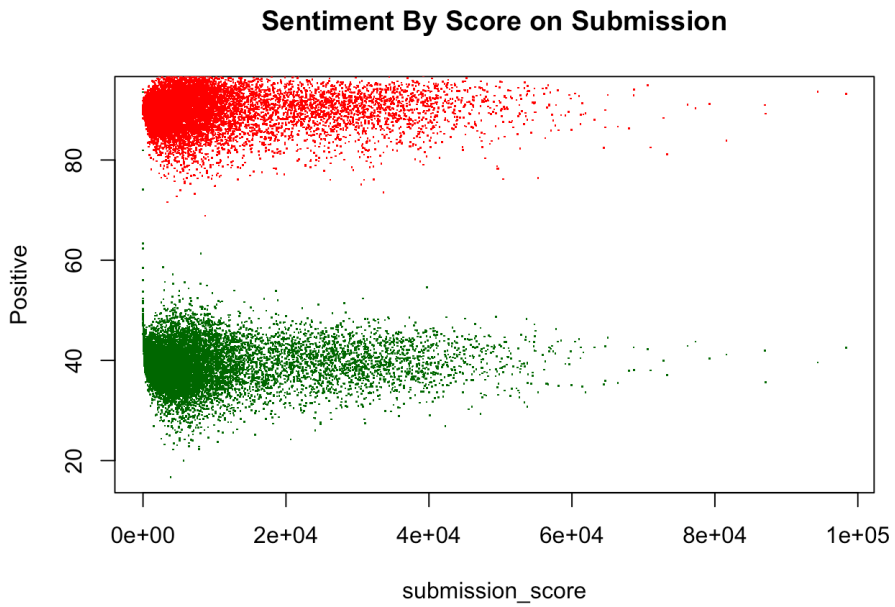
title	poscount	negcount
Report: Russia could send Edward Snowden back to the US as a 'gift' to Trump	25.0	100.0
Charleston Black Lives Matter leader shot, killed in New Orleans, niece says	33.33333333333333	100.0
U.S. Supreme Court Ruling Leads to Offensive Trademark Requests	66.66666666666666	100.0
Donald Trump Just Sexually Harassed Senator Kirsten Gillibrand, Critics Say	41.17647058823529	100.0
How Much Mussolini Is There in Donald Trump?	46.15384615384615	100.0
Bill Browder, Putin's loudest critic, just got his US visa back	33.33333333333333	100.0
Secret Service 'at the end of their rope' after being 'treated like servants by Trump': report	50.0	100.0
Somalis mistreated during US deportation effort, lawsuit alleges	100.0	100.0
Flake: Trump Administration Is Reminiscent Of McCarthy Era	66.66666666666666	100.0
Trump says he will visit Parkland, Florida, site of high school mass shooting	29.411764705882355	100.0

5. 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.

   Comments



   Submissions



6. Write a paragraph summarizing your findings. What does /r/politics think about President Trump? Does this vary by state? Over time? By story/submission?

   Based on our findings, we can see that there was an overwhelming more negative response and sentiment about Trump on r/politics. Looking at positive and negative Trump sentiment graphs, we see that the positive sentiment had a high of about a 42%, while for the negative sentiment, it reached almost 92% negative responses. Looking at the maps, we see that the southern states like Arkansas and Midwestern ones had a more favorable opinion about Trump, which is expected. On the negative sentiment graph, we see that Arkansas was basically white, the only one out of 50. This meant that very few people had a negative opinion about Trump. On the other hand, more liberal states (in general), like Maine and Ohio had very low positive percentage rates and high negative responses. Other states however, though overwhelmingly more negative than positive, were hard to distinguish from each other in terms of differences in negative sentiment, and slightly easier to distinguish on positive sentiment. This means that most states had about the same level of high negative sentiment, but varied positive sentiment on the lower end. Looking at the sentiment score by comments and submission, we still see a higher negative sentiment than positive. For both of these graphs, more data points are gathered around the left side of the graph, on the lower end of scores. We can see that scores with higher scores were usually expressed more negative thoughts about Trump. Based of the data, we can say that overall, considering everything, /r/politics had a more negative sentiment about Trump than positive.

QUESTION 1: Take a look at `labeled_data.csv`. Write the functional dependencies implied by the data.

Input_id → {labeldem, labelgop, 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?

Looking at the schema for the comments dataframe, I believe that the data frame does not look normalized. It is not free of redundancies that might affect the insert/update integrity. There are redundant columns like for the user data and columns that are unnecessary for this data frame. To decompose this data frame, we can break apart the comments dataframe by removing the subreddit_id and subreddit_type attributes, as these exist in other dataframes already. I believe that the collector of this data stored it this way because of the ease of identifying the different attributes of a comment in one centralized place. This can imply that he was probably just doing analysis on this data and wasn't planning to do any insertions or updates to this data that would affect the integrity of the data.

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?

sqlContext.sql("SELECT COALESCE(Comments.author_flair_text,Submissions.author_flair_text, null) as author_flair_text, Comments.created_utc, Submissions.title, Comments.id, Comments.body FROM Comments JOIN Submissions ON Comments.link_id_new = Submissions.id").explain()

```
== Physical Plan ==
*(4) Project [coalesce(author_flair_text#3, author_flair_text#54) AS
author_flair_text#344, created_utc#10L, title#106, id#14, body#4]
+- *(4) BroadcastHashJoin [link_id_new#317], [id#69], Inner,
BuildRight
   :- *(4) Project [author_flair_text#3, body#4, created_utc#10L,
id#14, pythonUDF0#351 AS link_id_new#317]
   :  +- BatchEvalPython [remove_three(link_id#16)],
[author_flair_text#3, body#4, created_utc#10L, id#14, link_id#16,
pythonUDF0#351]
   :     +- *(2) Project [author_flair_text#3, body#4,
created_utc#10L, id#14, link_id#16]
   :        +- *(2) Project [author#0, author_cakeday#1,
author_flair_css_class#2, author_flair_text#3, body#4, can_gild#5,
can_mod_post#6, collapsed#7, collapsed_reason#8, controversiality#9L,
created_utc#10L, distinguished#11, edited#12, gilded#13L, id#14,
is_submitter#15, link_id#16, parent_id#17, permalink#18,
retrieved_on#19L, score#20L, stickied#21, subreddit#22,
subreddit_id#23, subreddit_type#24]
   :           +- *(2) Filter isnotnull(pythonUDF0#350)
   :              +- BatchEvalPython [remove_three(link_id#16)],
[author#0, author_cakeday#1, author_flair_css_class#2,
author_flair_text#3, body#4, can_gild#5, can_mod_post#6, collapsed#7,
collapsed_reason#8, controversiality#9L, created_utc#10L,
distinguished#11, edited#12, gilded#13L, id#14, is_submitter#15,
link_id#16, parent_id#17, permalink#18, retrieved_on#19L, score#20L,
stickied#21, subreddit#22, subreddit_id#23, ... 2 more fields]
   :                 +- *(1) FileScan parquet
[author#0,author_cakeday#1,author_flair_css_class#2,author_flair_text#
```

```
3,body#4,can_gild#5,can_mod_post#6,collapsed#7,collapsed_reason#8,cont
roversiality#9L,created_utc#10L,distinguished#11,edited#12,gilded#13L,
id#14,is_submitter#15,link_id#16,parent_id#17,permalink#18,retrieved_o
n#19L,score#20L,stickied#21,subreddit#22,subreddit_id#23,subreddit_typ
e#24] Batched: true, Format: Parquet, Location:
InMemoryFileIndex[file:/media/sf_vm-shared/CS143/project2/comments],
PartitionFilters: [], PushedFilters: [], ReadSchema:
struct<author:string,author_cakeday:boolean,author_flair_css_class:str
ing,author_flair_text:strin...
   +- BroadcastExchange HashedRelationBroadcastMode(List(input[1,
string, true]))
      +- *(3) Project [author_flair_text#54, id#69, title#106]
         +- *(3) Filter isnotnull(id#69)
            +- *(3) FileScan parquet
[author_flair_text#54,id#69,title#106] Batched: true, Format: Parquet,
Location: InMemoryFileIndex[file:/media/sf_vm-
shared/CS143/project2/submissions], PartitionFilters: [],
PushedFilters: [IsNotNull(id)], ReadSchema:
struct<author_flair_text:string,id:string,title:string>
```

Looking at the Explain output of this join, we see that SparkSQL is using “BroadcastHashJoin.” When one of the tables are able to fit into RAM, this Broadcast Hash Join would be used. The broadcast means that the table is passed along to other nodes in the network. This smaller dataset that is able to fit in RAM is copied on all of the other nodes, which helps keep the parallelism of the larger table. Looking at the top lines, as well as subsequent lines that start with “Project”, we see that the SQL explain lists what attributes are projected. For example, at the very first line, we see “Project [coalesce(author_flair_text#3, author_flair_text#54”. This measn that the SQL function is projecting the attributes author_flair_text, created_utc, title, id, and body.