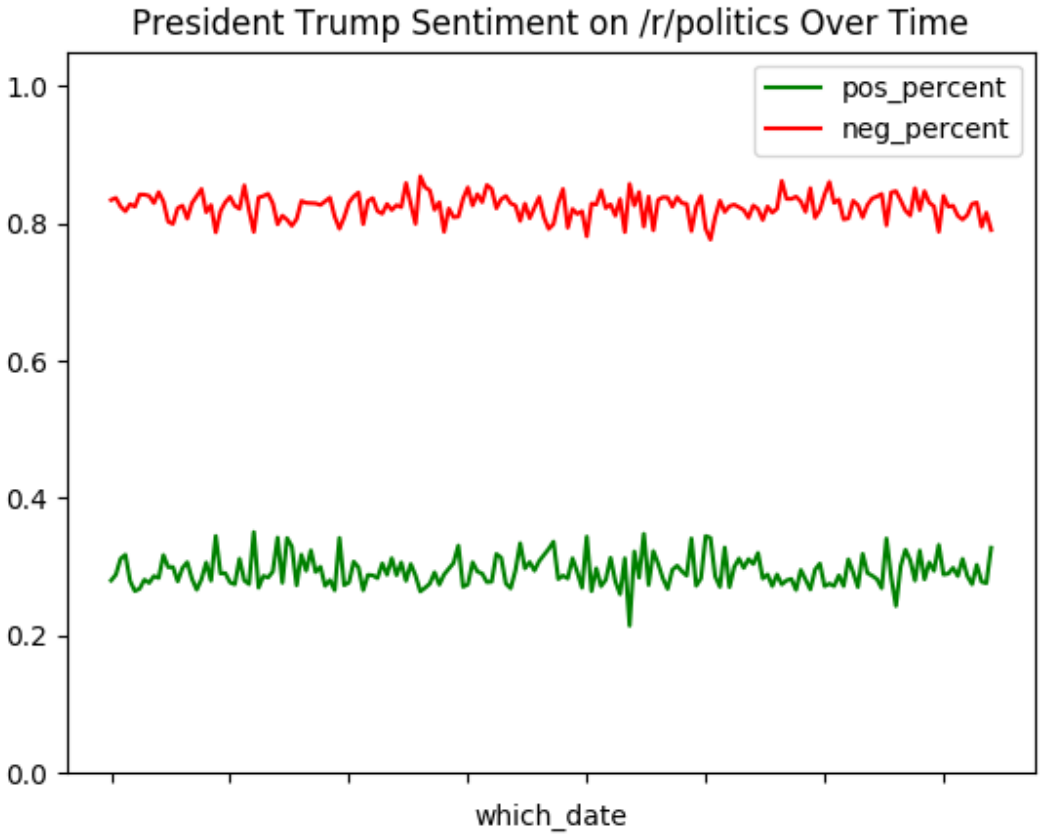
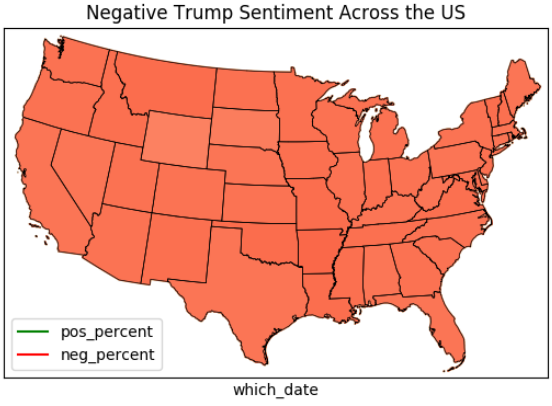
CS143 Project 2B Report
Zixia Weng(305029822), Don Lee(505032705), Shunji Zhan(405030387)

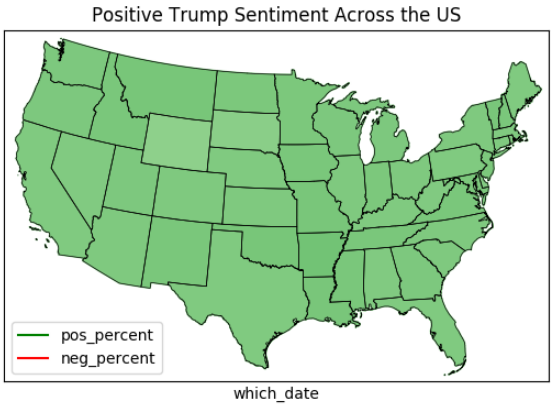
Final Deliverable

1.

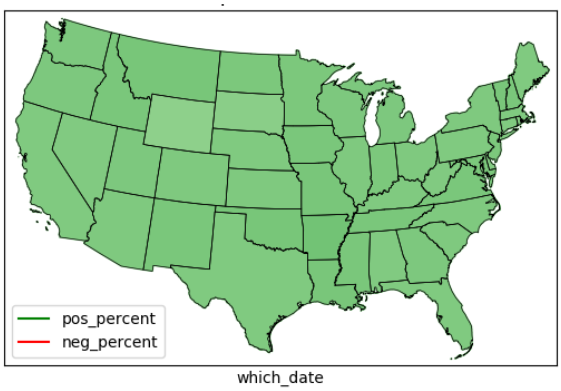


2.





3.



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):

10 positive stories:

Why did the Democrats fail so badly in the November 2016 elections?

Here's a List of Democrats Who Also Met With The Russian Ambassador

Putin praises to French presidential favorite Francois Fillon

European Union considering its own nuclear weapons program over fears it cannot rely on Trump's America

Trump as National Security Threat Revisited: A Scorecard

Newspapers are turning to hate group Conservative Republicans of Texas over the state's proposed bathroom ban

The Corporate, Third-Way Democrats Should Know That Screwing Public Schools Won't Get Dems. Elected

Want to Stop Brexit?? This is it!

Hey r/THE_DONALD, Trump will lose.

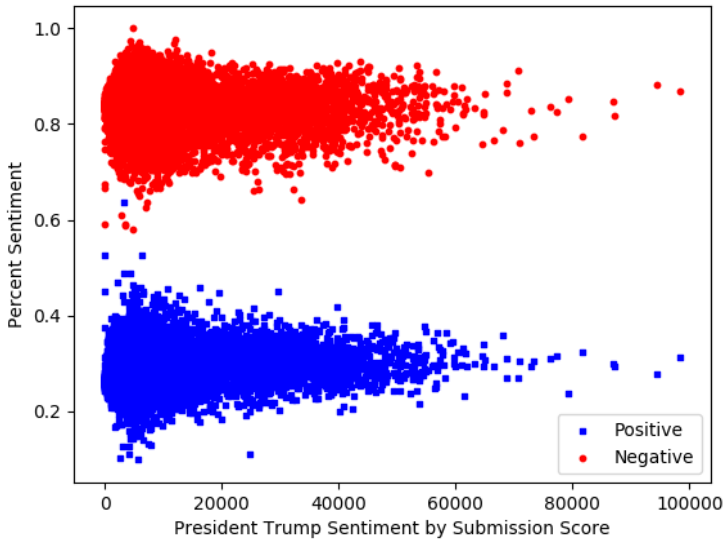
Kazakhstan to re-examine 2004 banker's death, may target Nazarbayev critic

10 Negative stories:

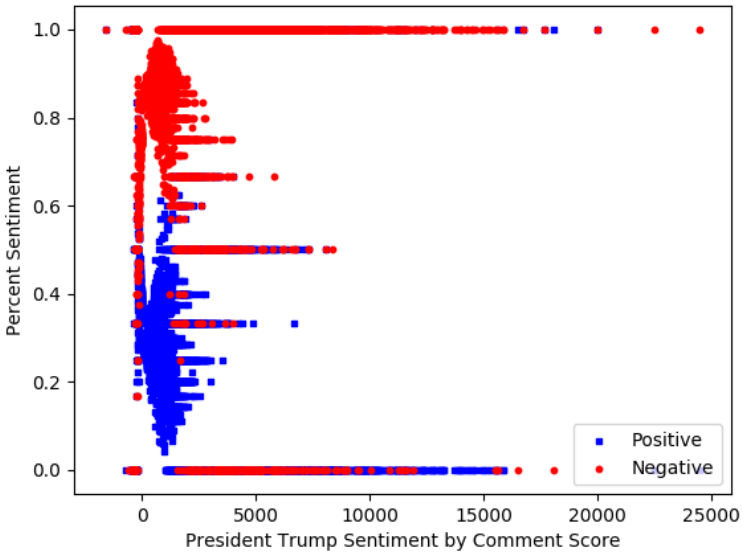
Trump's Deutsche Bank Records Subpoenaed by Mueller
Father of US Navy SEAL Ryan Owens killed in Yemen raid refuses to meet Donald Trump
The myth of the low-level aide
Trump administration to allow 872 refugees into U.S. this week
The simple explanation for why ESPN did not fire Jemele Hill but did fire Curt Schilling
Analysis | Trump called the news media an 'enemy of the American People.' Here's a history of the term.
WH official: Trump to tell Congress he approves GOP memo release
The Ultimate Hypocrisy? Trump Plan to Renegotiate NAFTA Resembles TPP Deal He Withdrew From
Tomi Lahren Was Made for the Trump Era
Did Obama just come out for legalizing marijuana?

5

a



b



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

Function dependencies: 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 is not normalized. We should decompose the data frame into one table for user data, one for subreddit data, and one for the comment relation and the comment it self. The reason that the collector stored the data this way could be that he/she wants to optimize the read performance in the cost of physical storage. With redundant data all in one table, read queries that normally would need join operations can be sped up since they no longer need to join tables.

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?

The selected join is :

```
SELECT      c.id,
            c.retrieved_on,
            body,
            c.score as comment_score,
            s.score as story_score, title,
            c.author_flair_text as state
FROM        comments_view c,
            submissions_view s
WHERE       SUBSTR(link_id, 4) = s.id AND NOT body LIKE '%&gt%'
```

The result of the EXPLAIN of the query is:

```
== Physical Plan ==
*(2) Project [id#14, retrieved_on#19L, body#4, score#20L AS comment_score#202L, score#92L AS story_score#203L, title#106, author_flair_text#3 AS state#204]
+- *(2) BroadcastHashJoin [substring(link_id#16, 4, 2147483647)], [id#69], Inner, BuildRight
   :- *(2) Project [author_flair_text#3, body#4, id#14, link_id#16, retrieved_on#19L, score#20L]
   :  +- *(2) Filter ((isnotnull(body#4) && NOT Contains(body#4, &gt)) && isnotnull(link_id#16))
   :     +- *(2) FileScan parquet [author_flair_text#3,body#4,id#14,link_id#16,retrieved_on#19L,score#20L] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf_vm-shared/comments.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(body), Not(StringContains(body,&gt)), IsNotNull(link_id)], ReadSchema: struct<author_flair_text:string,body:string,id:string,link_id:string,retrieved_on:bigint,score:bi...
   +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
      +- *(1) Project [id#69, score#92L, title#106]
         +- *(1) Filter isnotnull(id#69)
            +- *(1) FileScan parquet [id#69,score#92L,title#106] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf_vm-shared/submissions.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(id)], ReadSchema: struct<id:string,score:bigint,title:string>
```

The first thing to be noticed here is that Spark filters tuples before joining. After scanning the tables, the first thing to do is filtering. In our case, Spark removes all tuples from comments that have comment starting with '&gt'. After this, Spark does projection of selected columns on both joining tables, such that the actual join would require less memory. Then comes the actual join operation. The join used here by Spark is BoradcasHashJoin. This join puts one of child relation into memory, then uses a hash join that we've talked about in class. The last step is the final projection that projects the selected columns and discards the rest.