

Database System
CS143 Spring 2018

Project 2B Report

Fangyao Liu 204945018

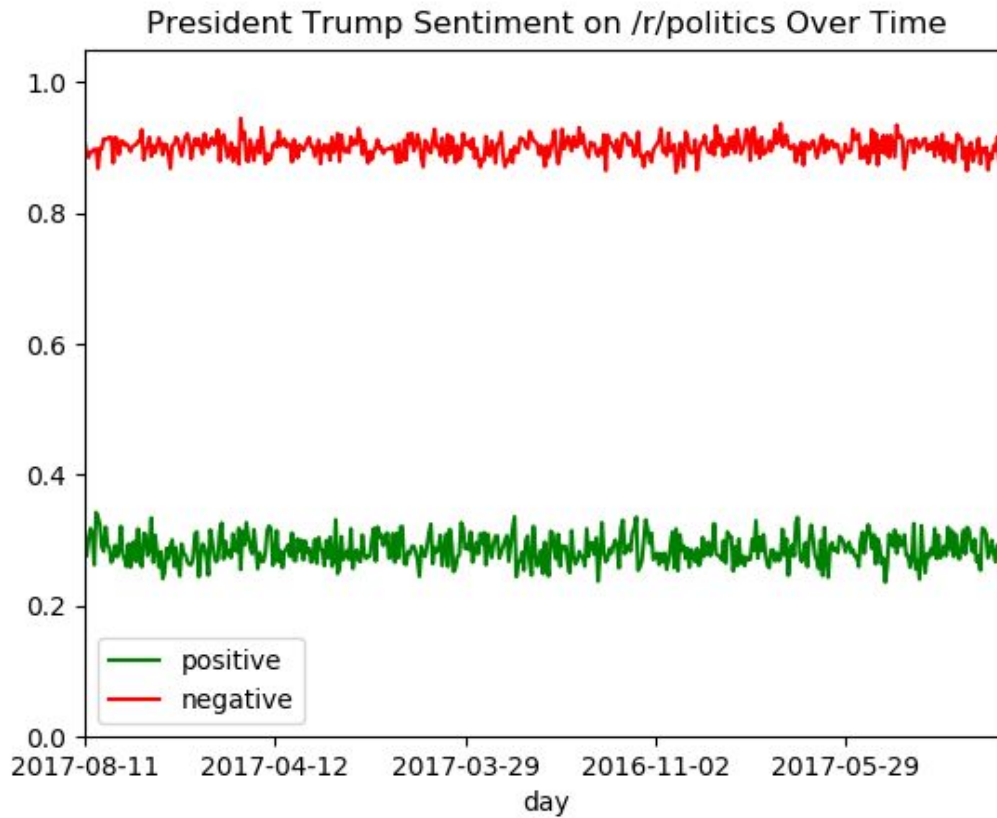
Xuan Hu 505031796

Yanzhe Xu 404946757

6/8/2018

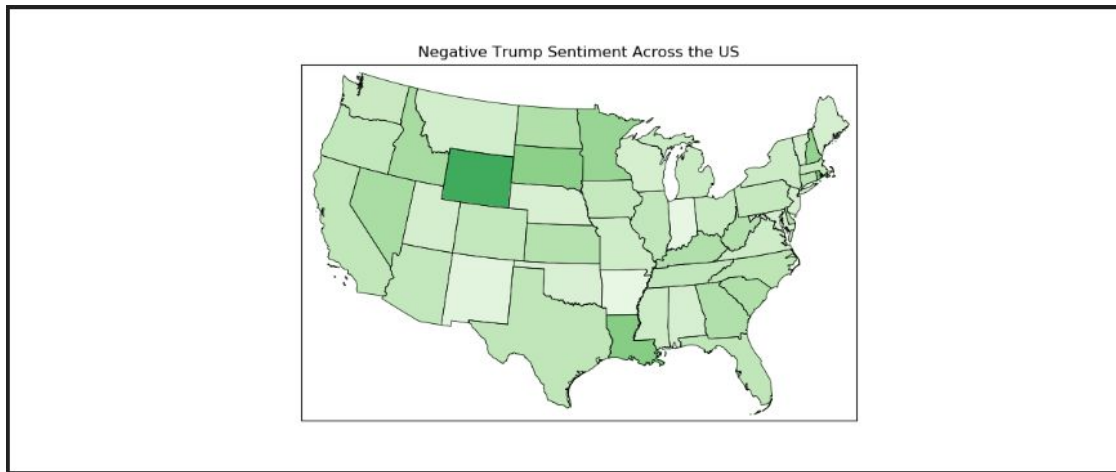
1. Final Result

1. Time series plot of positive and negative sentiment

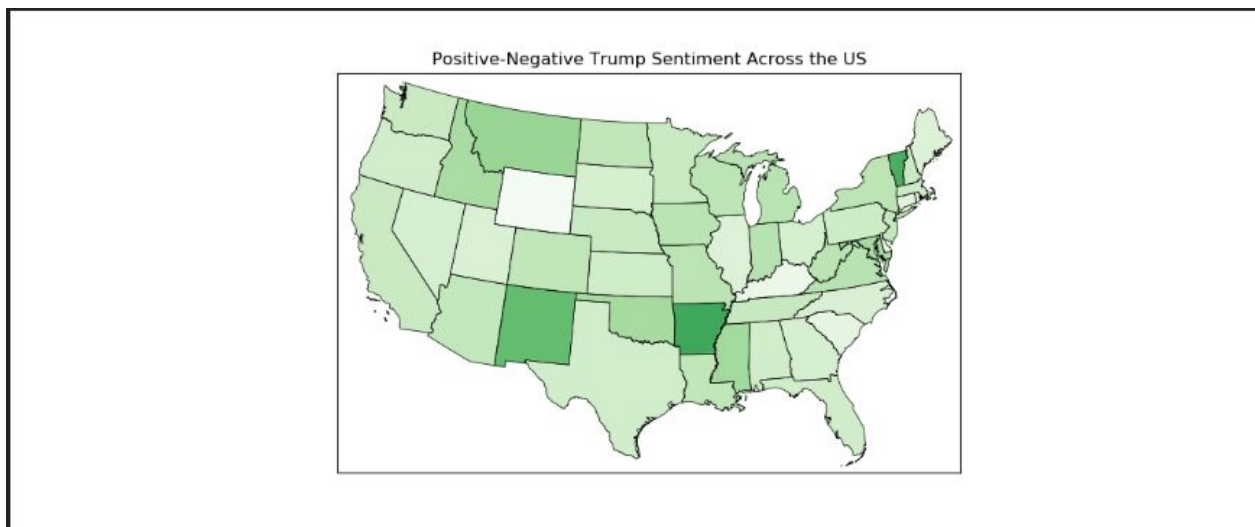


2. United states color Map





3. Third Map of United States(Positive-negative)



4. Top 10 positive stories and Top 10 negative stories

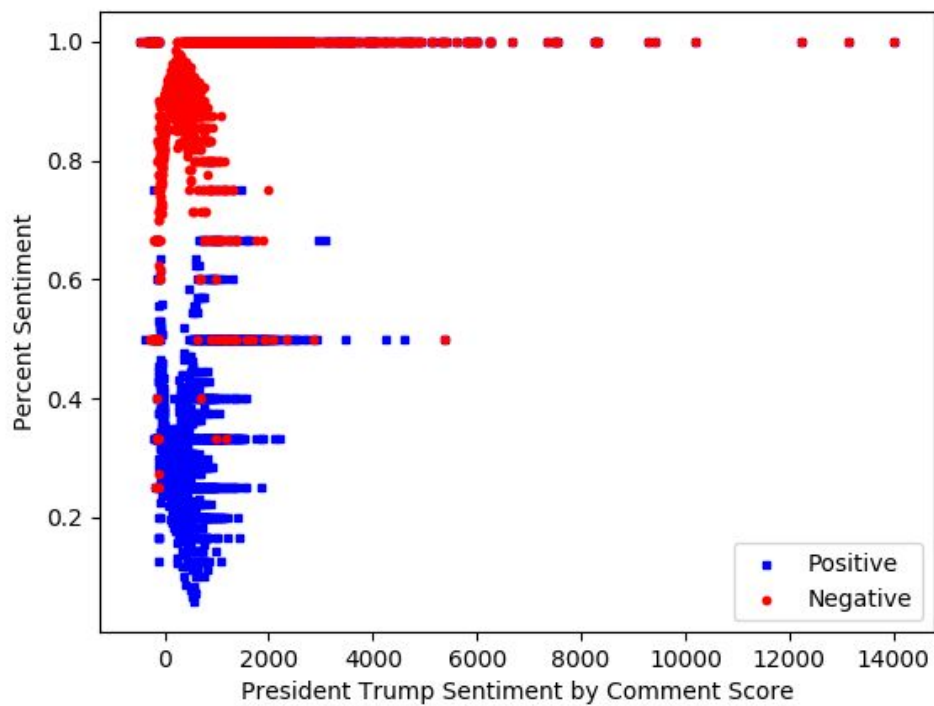
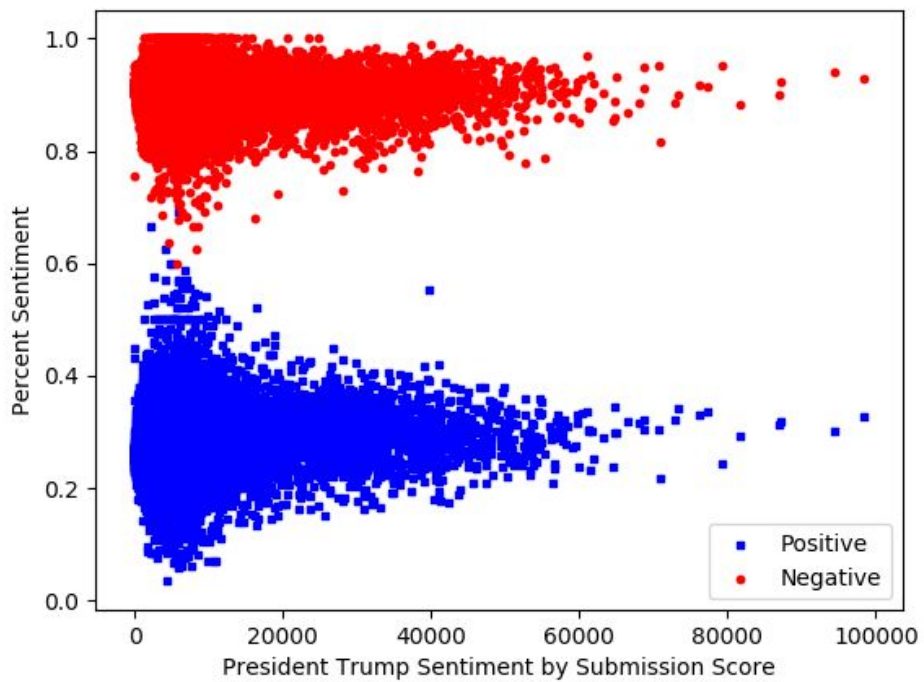
Top 10 positive stories

story_score	percentage
5861	0.692308
2268	0.666667
4255	0.625
4770	0.6
5023	0.6
6726	0.588235
2578	0.576923
4100	0.571429
6950	0.571429
5945	0.571429

Top 10 negative stories

story_score	percentage
4590	1
5409	1
11860	1
3845	1
6156	1
7146	1
3866	1
9686	1
4256	1
5936	1

5. Scatter Plots



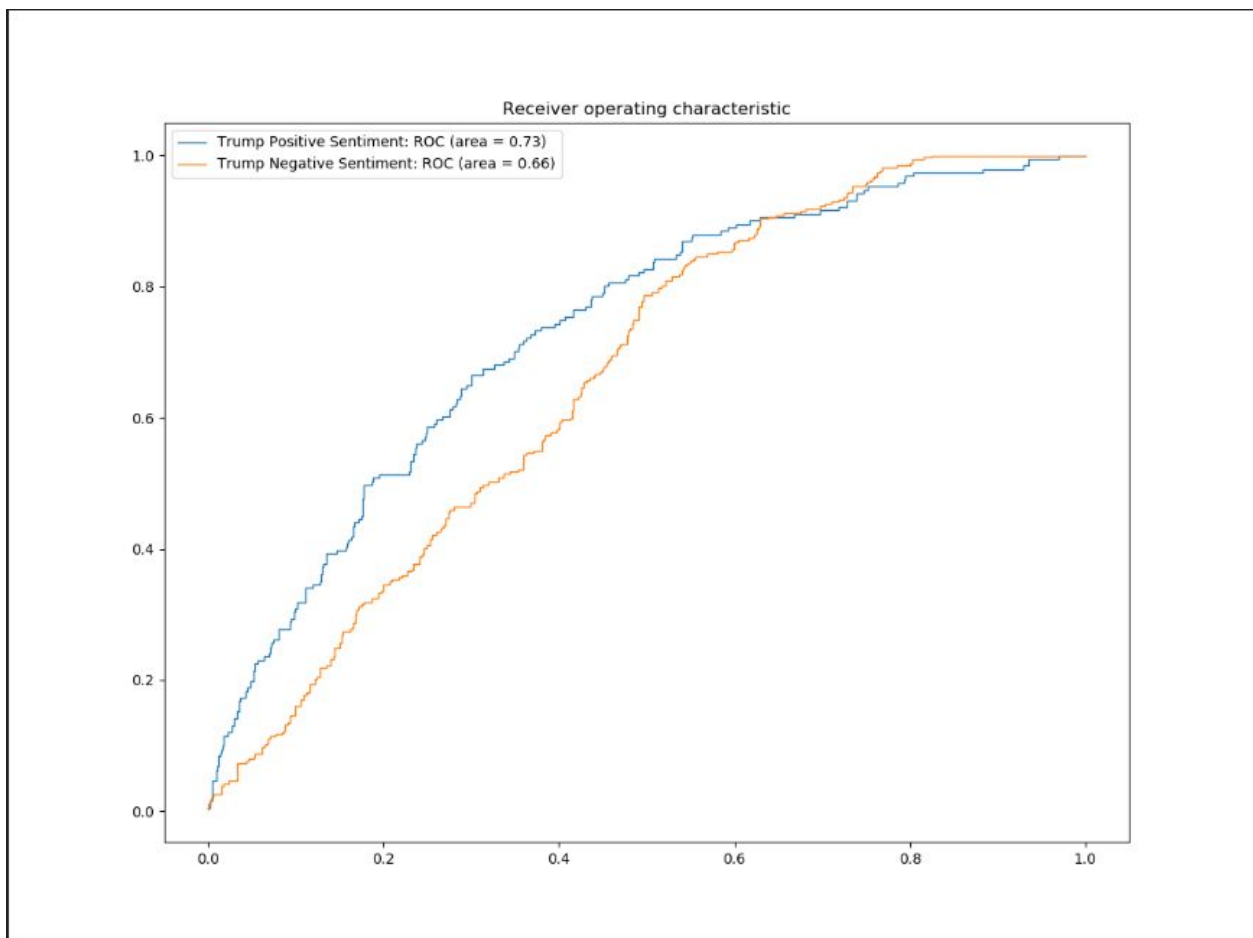
From the two plots above, we can easily notice that the difference between positive and negative percentage becomes larger as the submission score grows. On the other side, most points in second graph locate in 0-2000 range, we can't have clear boundary between positive and negative percentage. Thus, submission score can be a good feature while comment score may not perform well as the submission score.

6. Any other plots that make sense will receive extra credit.

Not Applicable.

7.Extra Credit: Roc Curve

In this section, we want to evaluate our classifiers by using ROC curve. We use False Positive Rate as X axis and True Positive Rate as Y axis. In the meanwhile, we calculate the auc for these two curve to help us analyze their performance. There are two curves in the graph, one is for positive classifier and the other one is for negative classifier.



8. Attitude towards President Trump

From the results shown above, it is obvious that most /r/politics users think negative about President Trump. And this phenomenon varies by state, although not too much. The percentage positive of all of the states in U.S are from 0.25 to 0.3 and the percentage negative is from 0.8 to 0.9 for all states. In addition, comments about President Trump also won't vary over time. From time series plot, the curve of percentage negative and positive are nearly two parallel lines to X axis which means that they are stable and won't vary over time. For comment score, when it is small, the percentage of negative is often high, only little points with low percentage. With the increase of comment score, less and less low percentage appears and almost all cases with a high percentage of negative(near 1). As for positive, it is totally opposite to negative situation, with very low percentage when comment score is small, and disappear with the increase of comment score. For submission score, it is same as the situation of comment score when it is low. With the increase of submission score, the percentage of negative gradually close to 0.9 and the percentage of positive gradually close to 0.3.

2. Answer for Question

QUESTION 1:

A: The functional dependencies in the data are Input_id -> labeldem and Input_id -> labeldjt.

QUESTION 2:

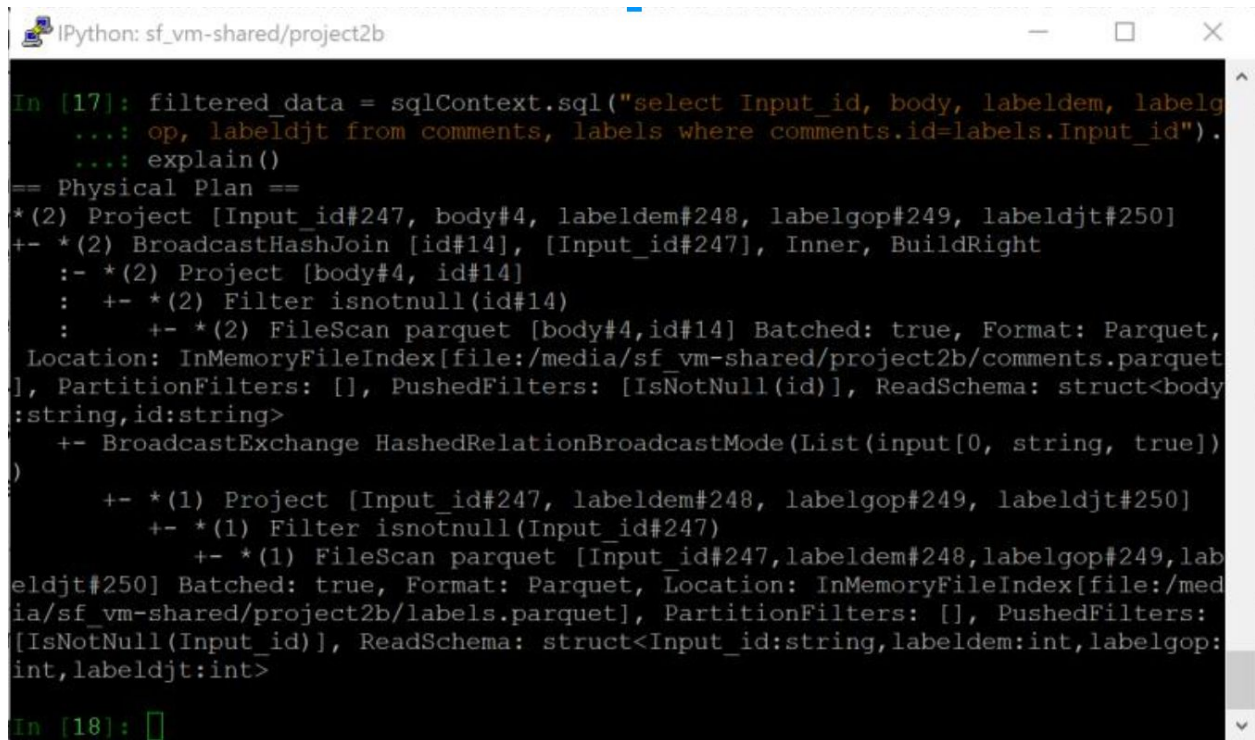
Take a look at the schema for the comments dataframe. Forget BCNF and 3NF. Does the data frame *look* normalized? In other words, is the data frame free of redundancies that might affect insert/update integrity? If not, how would we decompose it? Why do you believe the collector of the data stored it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

A: This data frame is not normalized. There are redundancies that affect insert integrity. For example, one user changes his author_cakeday and makes a new post. This will violates with his previous integrity since the birthday is not the same as his previous post. The way to avoid this is to decompose the relation by splitting author information and comment information. Then create another table containing their foreign keys to refer each other.

We think the reason that the collector of the data stored it in this way is to process user requests faster. As we know, it is costly to do join on tables so that we want to avoid that. By using redundancies in the table, we can get rid of some join operations and then process request quickly. For those website, the user experience is most important so that it is reasonable to sacrifice some storage space for time.

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?



```
In [17]: filtered_data = sqlContext.sql("select Input_id, body, labeldem, labelgop, labeldjt from comments, labels where comments.id=labels.Input_id").explain()
== Physical Plan ==
*(2) Project [Input_id#247, body#4, labeldem#248, labelgop#249, labeldjt#250]
+- *(2) BroadcastHashJoin [id#14], [Input_id#247], 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/project2b/comments.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(id)], ReadSchema: struct<body:string,id:string>
   +- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
   +- *(1) Project [Input_id#247, labeldem#248, labelgop#249, labeldjt#250]
      +- *(1) Filter isnotnull(Input_id#247)
         +- *(1) FileScan parquet [Input_id#247,labeldem#248,labelgop#249,labeldjt#250] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf_vm-shared/project2b/labels.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(Input_id)], ReadSchema: struct<Input_id:string,labeldem:int,labelgop:int,labeldjt:int>

In [18]:
```

A: Spark uses BroadcastHashJoin to join table comments and table labels. From the screenshots above we can see that first Spark extract body, id from table comment. They use the attributes from comments table to build right table. Then Spark extract Input id, labeldem, labelgop, labeldjt from table labels and join them together. It is also mentioned that Spark use inner join to finish this task.

3.Extra Credit task

3.1.Task 10 question 5:

In these two classification, we have four results in total. Use number xx to represent the emotion of a comment. The first x represent whether a comment is positive or not, the second x represent whether a comment is negative or not. So we can get 00,01,10,11 four results in total. When xx is 11, it means the owner of comment think Trump is controversial. On the one hand, he did something meaningful. On the other hand he did something which people don't agree with. When xx is 00, it means Trump didn't show a strong sense of presence. He didn't do something impressive. When xx is 10, the owner of comment think Trump it is a good president. When xx is 01, Trump may not liked by the owner of comment.

By combining the emotion of positive and negative, we can get more information compared with watching these two separately. Things are always changing. People are also changing too. Maybe a person hates Trump before, but with the change of time, he may think, oh, he is a pretty nice person. So in this task, we try to find whether people change their attitude towards president Trump with the change of time. Sadly, there is no obvious variation of people's attitude. Despite time of our data across from 2016 to 2018, we don't see any change. 00 is around 5%, 01 is around 65%, 10 is around 5% and 11 is around 25%. From the data we calculate, we can get two conclusions. First, people's attitude towards president Trump doesn't change from 2016 to 2018. Most people don't like him. Some people think he is controversial and few people like him.