UNIVERSITY OF CALIFORNIA, LOS ANGELES
CS 143: Database Systems
Project 2B

Reddit Politics Comment Data Analysis Based On Spark

Name: Jiahui Lu
UID: 204945099
Instructor: Prof. Ryan Rosario
Jun 5, 2018

1 INTRODUCTION

In this project, we will focus on the data management and basic analysis with machine learning knowledge base under Spark framework. We will use the Reddit data to find the sentiment across time, across topics, and across states regarding President Trump. Moreover, we will do the comment data analysis based on the source data given to see the positive and negative sentiment among Reddit users.

2 DATA INTERPRETATION

2.1 Source Data

The source data comes from Reddit /r/politics and covers submissions and comments from November 2016 (right before the election) to the latest data dump covering to February 2018. It also includes all comments and submissions made to other subreddits made by users that post to /r/politics.

The comments-minimal.json.bz2 contains comments in JSON format and then the submissions.json.bz2 contains submissions in JSON format. In the meanwhile, Professor took a sample of 2,000 Reddit Politics comments and posted them to Mechanical Turk in order to get the file labeled_data.csv, which contains a comment_id and sentiment labels on all three dimensions (Democrat, Republican, Trump specifically). Here -1 means negative, 1 means positive, 0 means neutral and -99 means not applicable. Here, we will only work with positive and negative labels.

Table 1 Source Data

Source Data	**Description**
comments-minimal.json.bz2	comments in JSON format
submissions.json.bz2	submissions in JSON format
labeled_data.csv	comment_id and sentiment labels on all three dimension

2.2 Functional Dependencies

We can take a look at our functional dependencies for the source table we have.

<u>QUESTION 1</u>: we now take a look at labeled_data.csv, which is relatively simple

then we can write the functional dependencies implied by the data.

$$\{\text{Input } ID \rightarrow \text{labeldem, labelgop, labeldjt}\}$$

QUESTION 2: we can also take a look at the schema for the *comments/submission* data-frame.

For the comment and submission table, the superkey can be ID or Body/Title itself, it satisfies the requirement that the key contain no duplicates. ID and Body, however, is related, id is decided by Body. And the other fields are related with Body, because the main thing in this table is to represent the comment and related information. The other thing that we should notice is that the subreddit, subreddit id and subreddit may be redundant, which may break the integrity inserting or updating. So we can break these three and the other apart. To sum up, the table is not the minimum amount of data necessary to describe a comment. That's not a normalized table. So the data frame is not free of redundancies and it might affect insert/update integrity.

So we would decompose based on ID or Body/Title as the only superkey to build up relationship with the other fields. The advantage of storing in this way is it can improve performance by minimizing the need for joins, precomputing aggregate values so that computing them at data modification time, rather than at select time and of course reducing the number of tables, meaning reduce the JSON file numbers in this case. So moreover, it boosts the viewing due to the fact that normalized data will use less space, but may need to apply joins so as to construct something we want, which will use more time. If deformalized, data are replicated in several places, which will cost more space, but the desired viewing of the data we have is readily available without further operations.

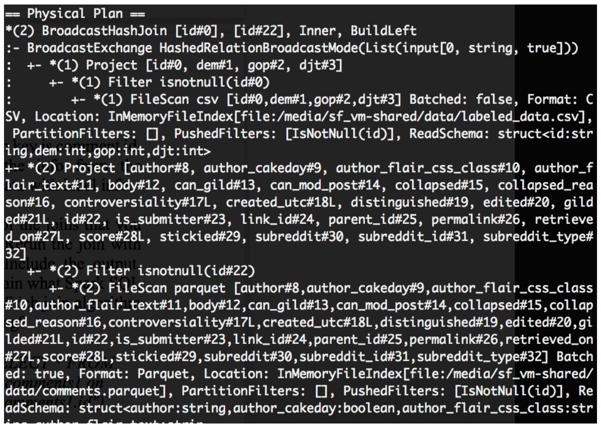
3 DATA MANEGEMENT

In order to train a classifier, we only need to work with the labeled data, but we need to do join to make labeled_data.csv associated with the comment data. The join key is comment id, after join we can get the info from the comment data as well as data with label info.

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?

*sqlDF = spark.sql("SELECT * FROM labeled_data1 JOIN comments1 on labeled_data1.id = comments1.id")*

We can dive into the very first join we implement. Using explain function attached, we can have the output:



```
== Physical Plan ==
*(2) BroadcastHashJoin [id#0], [id#22], Inner, BuildLeft
:- BroadcastExchange HashedRelationBroadcastMode(List(input[0, string, true]))
:  +- *(1) Project [id#0, dem#1, gop#2, djt#3]
:     +- *(1) Filter isnotnull(id#0)
:        +- *(1) FileScan csv [id#0,dem#1,gop#2,djt#3] Batched: false, Format: C
SV, Location: InMemoryFileIndex[file:/media/sf_vm-shared/data/labeled_data.csv],
 PartitionFilters: [], PushedFilters: [IsNotNull(id)], ReadSchema: struct<id:str
ing,dem:int,gop:int,djt:int>
+- *(2) Project [author#8, author_cakeday#9, author_flair_css_class#10, author_f
lair_text#11, body#12, can_gild#13, can_mod_post#14, collapsed#15, collapsed_rea
son#16, controversiality#17L, created_utc#18L, distinguished#19, edited#20, gild
ed#21L, id#22, is_submitter#23, link_id#24, parent_id#25, permalink#26, retrieve
d_on#27L, score#28L, stickied#29, subreddit#30, subreddit_id#31, subreddit_type#
32]
   +- *(2) Filter isnotnull(id#22)
      +- *(2) FileScan parquet [author#8,author_cakeday#9,author_flair_css_class
#10,author_flair_text#11,body#12,can_gild#13,can_mod_post#14,collapsed#15,collap
sed_reason#16,controversiality#17L,created_utc#18L,distinguished#19,edited#20,gi
lded#21L,id#22,is_submitter#23,link_id#24,parent_id#25,permalink#26,retrieved_on
#27L,score#28L,stickied#29,subreddit#30,subreddit_id#31,subreddit_type#32] Batch
ed: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf_vm-shared/
data/comments.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(id)], Re
adSchema: struct<author:string,author_cakeday:boolean,author_flair_css_class:str
```

Generally, Spark will use auto Broadcast Join Threshold and automatically broadcast data, which means BroadcastHashJoinExec will get chosen when there are joining keys and one of the following holds. here it will go for inner, and is BuildLeft, which is the same as what we set up. Then spark use a BroadcastHashJoin, which act as a filter to select the data it needs as well as the columns as we specified before. It will also project and filter as well not null item through the file formate which is both csv and parquet and the location is InMemoryFileIndex.

In general, Spark prefer a sort merge join over a shuffled hash join. One thing that ShuffledHashJoin is helpful when we could fit in memory, in the meanwhile, if the build side is much smaller than the other side, it will also useful. Because the building hash table on smaller one is faster.

So spark will go sort-merge join for large joins and it is more robust as Shuffled Hash Join requires the hashed table to fit in memory, however, Sort Merge Join can spill to disk to help speed up and make best use of the distributed system property.

After deal with the join, we can see that Spark allows us to define custom SQL functions called user defined functions (UDFs). In order to apply the clean text function, we need to register UDFs. After we clean the text of comment and parse them into a smooth format that we can eventually use to train a classifier. This step will add a column containing the cleaned comment data with grams from unigram, bigrams, and trigram as a array of string.

Another thing we need to carry out is to use a binary CountVectorizer so as to turn the raw features into a sparse feature vector, which is a data structure that Spark ML framework understands. Here we set minDF to be 5, which will filter low frequency ones whose occurrence is less than 5 times. Then we will get the unseen vector selected feature data, each feature may have a value associated with it such as a count of a TF-IDF value.

After we create two new columns representing the positive and negative labels following the rules of for the positive label that is 1 when the original label is 1, and 0 everywhere else and same rule for negative data. We can get the data in order to be trained.

Table 2 Schema Table

Schema	Usage
Comment ID	Join key
Other Features in comment	A bunch of other things
Grams	Clean text output
CountVectorizer features	Unseen vector as features for training
Positive label	0/1 data
Negative label	0/1 data

4 MODEL FITTING WITH LOGISTIC REGRESSION

We will build a positive sentiment model and a negative sentiment model based on the positive label, negative label and unseen features vector.

Applying logistic retrogression, we train a logistic regression model on the features (unigrams, bigrams and trigrams). Instead of dividing the data into a training and testing set, we will take it into 5 fold and get the best model.

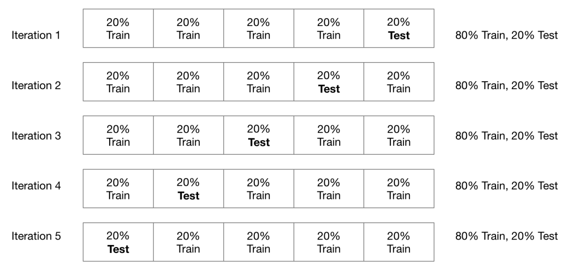


Figure 1. 5-fold Validation

We can get the ROC curve for as a measurement for the accuracy. We get a area under the curve is 0.708 for positive model and 0.721 for negative model. In the meanwhile, the ROC curves are plotted accordingly, which show the positive and negative classifier.

We must threshold these probabilities to convert them into 0/1 indicating if the comment is positive sentiment and if the comment is negative sentiment (probabilities are not intuitive to work with). If the positive probability is greater than 0.2, label it the positive sentiment a 1. If the negative probability is greater than 0.25, label the negative sentiment as 1. Basically, this data is imbalanced, that is, the number of positive labeled comments is not the same as the number of negative labeled comments. Instead, we use this proportion as the cutoff, to represent a result that is obtained solely by chance.

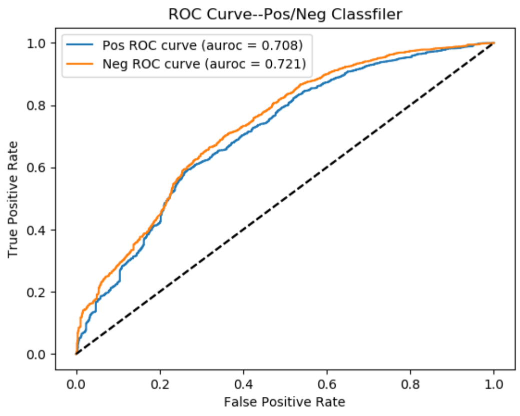


Figure 2. ROC Curve

5 MODEL EVALUATION

In this part, we will make test based on the data from file comments-minimal.json.bz2 and submissions.json.bz2. Then we will run the data analysis with

For the data management, we first need to merge data from comment and submission first. We will get the timestamp when the comment was created, the title of the submission (post) that the comment was made on, and the state that the commenter is from. Several other detailed things will also be carried out like the first three characters represent the type of link and looks like "t3_". This must be stripped off when joining between link_id (comments) and id (submissions). In addition, the state will come from the author_flair_text field. The time will be transformed by function from_unixtime to day. We also get rid of "/s" and text starts with "&gt" to make the result more convicing.

After applying the same transformer in the previous part, we will use the transform method to apply the CountVectorizer to the unseen data and then apply the classifier to unseen data to get predicted probabilities. Among the three new columns in posResult and negResult we only care about probability, which has already been threshold, which is the final result. Below table gives the details about the schemas for this data management regarding of evaluation.

Table 3 Schema for Evaluation Table

Schema	Usage
Comment ID	ID
Ts	Time when created
Title	It is the story
Body	Raw Comment
c-Score	Comment Score
s-Score	Submission Score
state	State info
grams	Unaltered feature
CountVectorizer features	Unseen vector as features for training
Positive label	0/1 data
Negative label	0/1 data

Overall, we sampled around 20% of the total data, which is about 179324 data. The psotive is 31.74% and the Negative is 84.49%

Firstly, we can create a time series plot (by day) of positive and negative sentiment. Which has two lines, one for positive and one for negative.

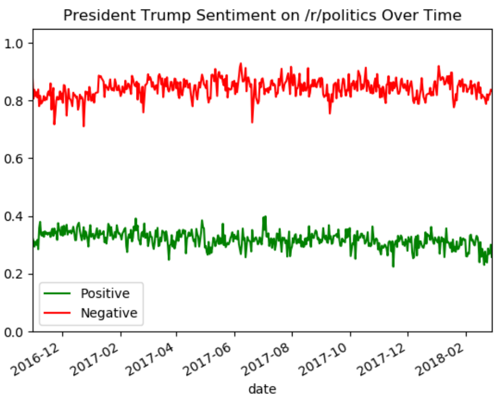


Figure 3 Sentiment over Time

Secondly, we can do 2 maps of the United States: one for positive sentiment and one for negative sentiment with colors in states by the percentage.

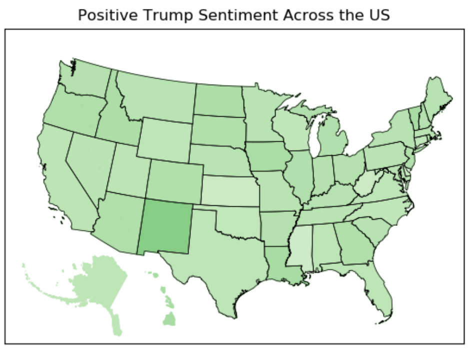


Figure 4. Positive Sentiment over States

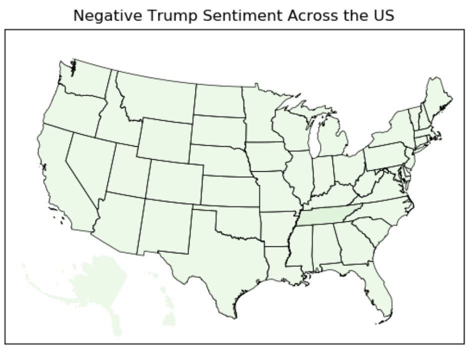


Figure 5. Negative Sentiment over States

Thirdly, we can do the third basemap of the United States for the difference between Positive and Negative for each state.

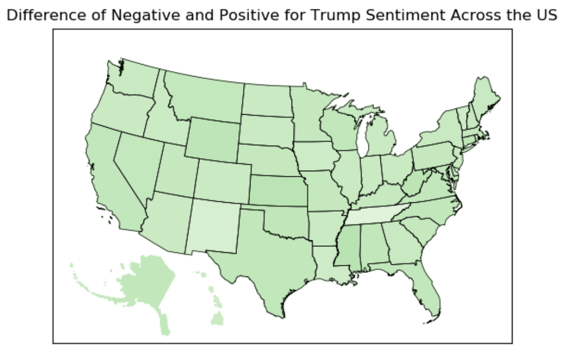
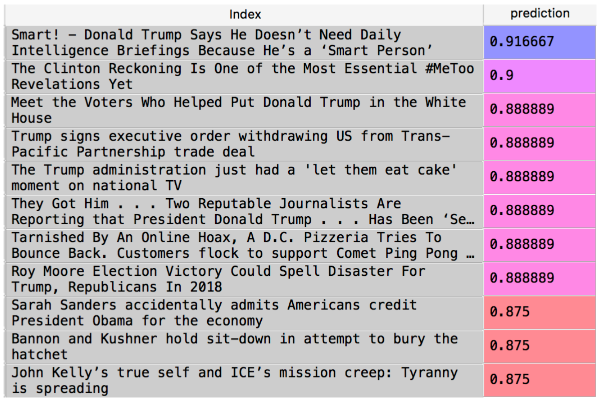

Difference of Negative and Positive for Trump Sentiment Across the US

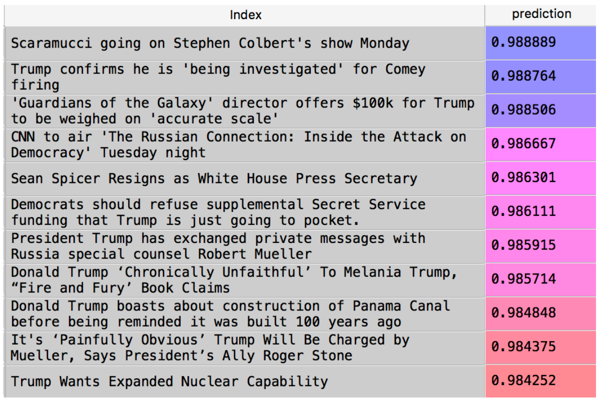
Figure 6. Sentiment Difference over States

Fourthly, base on the result, we will give out the list of top 10 positive titles and the top 10 negative titles. They have the top percentages respectively. In order to get rid of the poor sampling when it comes to situation that only one positive or negative ones are counted against the final result, we drop the rows with prediction with 100% due to sampling problem. There are actually several ones. The tables below give the results as well as the percentage.



Index	prediction
Smart! – Donald Trump Says He Doesn’t Need Daily Intelligence Briefings Because He’s a ‘Smart Person’	0.916667
The Clinton Reckoning Is One of the Most Essential #MeToo Revelations Yet	0.9
Meet the Voters Who Helped Put Donald Trump in the White House	0.888889
Trump signs executive order withdrawing US from Trans–Pacific Partnership trade deal	0.888889
The Trump administration just had a 'let them eat cake' moment on national TV	0.888889
They Got Him . . . Two Reputable Journalists Are Reporting that President Donald Trump . . . Has Been ‘Se…	0.888889
Tarnished By An Online Hoax, A D.C. Pizzeria Tries To Bounce Back. Customers flock to support Comet Ping Pong …	0.888889
Roy Moore Election Victory Could Spell Disaster For Trump, Republicans In 2018	0.888889
Sarah Sanders accidentally admits Americans credit President Obama for the economy	0.875
Bannon and Kushner hold sit–down in attempt to bury the hatchet	0.875
John Kelly’s true self and ICE’s mission creep: Tyranny is spreading	0.875

Figure 7. TOP 10 Positive Title



Index	prediction
Scaramucci going on Stephen Colbert's show Monday	0.988889
Trump confirms he is 'being investigated' for Comey firing	0.988764
'Guardians of the Galaxy' director offers $100k for Trump to be weighed on 'accurate scale'	0.988506
CNN to air 'The Russian Connection: Inside the Attack on Democracy' Tuesday night	0.986667
Sean Spicer Resigns as White House Press Secretary	0.986301
Democrats should refuse supplemental Secret Service funding that Trump is just going to pocket.	0.986111
President Trump has exchanged private messages with Russia special counsel Robert Mueller	0.985915
Donald Trump 'Chronically Unfaithful' To Melania Trump, "Fire and Fury' Book Claims	0.985714
Donald Trump boasts about construction of Panama Canal before being reminded it was built 100 years ago	0.984848
It's 'Painfully Obvious' Trump Will Be Charged by Mueller, Says President's Ally Roger Stone	0.984375
Trump Wants Expanded Nuclear Capability	0.984252

Figure 8. TOP 10 Negative Title

Then, we can also go for a look of submission score, or comment score, in order to see if they can go as a feature.

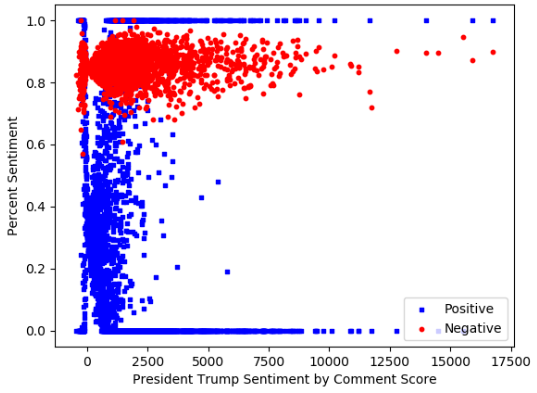


Figure 9. Sentiment over Comment Score

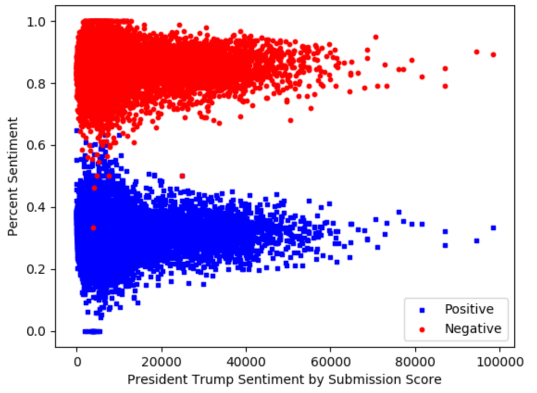


Figure 10. Sentiment over Submission Score

Moreover, we can define the approval rate:

$$\text{Approval Rate} = \alpha \frac{\%pos}{\%neg}$$

Here α is the ratio for average value of negative percentage over positive percentage. It will give the other aspect of the general aspect of the president's job, using ratio will better make us better see the sentiment state. The default value will be around 1 in average. We can compare our result with the result provided from Matthew Yglesias from *VOX*.

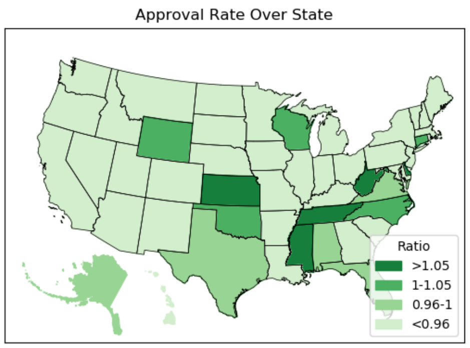


Figure 11. Simulated approval rate

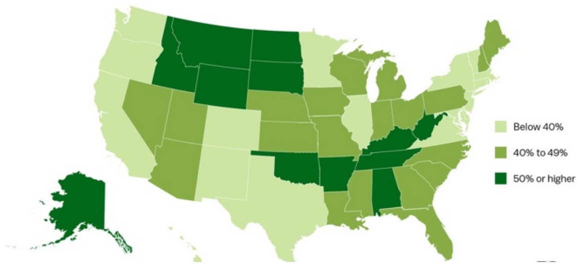


Figure 12. Actual approval rate

6 SUMMARY

In a nutshell, we cam draw a conclusion regarding what the /r/politics think about President Trump. A quick conclusion is that everyone seems be mad at our president. The percentage of overall negative and positive rate is that negative will take up to about 80% while positive will be nearly only 30%. So in a big scope, the people were expressing bad feelings about the president. For time, accordingly, these does not change heavily, but we can sense a tendency that positive is going down while the negative are going up. Generally, over states it reflects the the state that has low rate is consistent with the actual data (the shallow ones), and also some states have higher support rate can be view from Figure 8 and Figure 9, which can be viewed in the east-southern and northern part of each map. California in practically has one of the lowest support rate for Trump, which is rather reasonable. As for submission score and comment score, we can see that in general it shows a "bell" shape, so the highest percentage is not the one with the most emotion but the one being rather moderate, which is fair. Because, reasonable words will tend to get better score rather than spelling or flatting. However, for positive one with comment score, there are not too much pattern to follow.

REFERENCE

[1] Project 2B Specification https://ccle.ucla.edu/mod/page/view.php?id=2013590

[2] Spark SQL and Data Frames https://spark.apache.org/docs/latest/sql-programming-guide.html

[3] Open Source Spark SQL Guide https://spark.apache.org/docs/latest/ml-features.html

[4] Extracting Features in Spark MLLib https://docs.databricks.com/spark/latest/spark-sql/index.html

[5] Spark SQL Guide https://spark.apache.org/sql/

[6] Basemap plot for 50 states https://stackoverflow.com/questions/39742305/how-to-use-basemap-python-to-plot-us-with-50-states

[7] SILBERSCHATZ, A. DATABASE SYSTEM CONCEPTS. S.l.: MCGRAW-HILL EDUCATION.

[8] Trump's approval rating is below 50% in 38 states https://www.vox.com/policy-and-politics/2018/1/30/16949748/trump-approval-state

[9] Spark Join-related https://stackoverflow.com/questions/50019457/why-does-spark-planner-prefer-sort-merge-join-over-shuffled-hash-join