Kubilay Agi

Lawrence Chen

Project 2B Report

**Summary of Findings:**

The first plot shows that there is no significant change in terms of positive or negative sentiment throughout the year. This implies that people tend to maintain their position/stance on trump despite any new information or changes over the course of the year. It is also apparent that the percentage of negative sentiment than that of positive sentiment. From the second set of plots, specifically the negative trump sentiment plot, we observe that the central united states have a slightly less negative sentiment toward trump than the coasts. If we exaggerated the differences in our sentiment percentages, then this would be more apparent. On the third plot, the lighter the colors mean that the state is more even split between positive and negative sentiments. However, if the color is dark brown/red or dark blue, that means that the state leans heavily towards one sentiment over the other. Further, the ones near the central united states seem to be more polar whereas the sentiment along the coasts seems to be more split. From the fourth plot, it is difficult to draw any conclusions because it is possible for submissions to have one comment with 100% positive sentiment (one upvote). According to the per submission score, as the submission scores go up, the percentage of positive and negative sentiment converge to a value and tends toward negative sentiment. What is clear is that lower submission scores have a wider range of sentiment, but as you increase the submission score, there is a narrower spread of where the sentiment is. This implies that submissions with high scores will tend to have similar sentiment percentages. For comments, the positives become very low and the negative sentiment becomes very high which shows us that comments that are negative toward trump are highly score and lower scored comments don’t show a tendency toward one sentiment or the other

**Questions:**

1.) There are 7 functional dependencies implied by this data:

Input\_id -> {labeldjt, labeldem, labelgop}

Input\_id -> {labeldjt, labeldem}

Input\_id -> {labeldjt, labelgop}

Input\_id -> {labeldem, labelgop}

Input\_id -> labeldjt

Input\_id -> labeldem

Input\_id -> labelgop

2.) The comments table is not normalized. We notice that there are attributes for subreddit\_id and subreddit (which we interpreted to be the name of the subreddit). This can be decomposed into a single separate table because subreddits can be identified by their id, there is no need to add the name as well.

Additionally, assuming that the author\_flair\_text does not change depending on the location of where the author is and that a cakeday means a birthday, we do not need to include author\_flair\_text or the author\_cakeday in this table. We can have a separate table that associates the author of a comment with their flair text and their birthday. This data is redundant.

Perhaps they chose to store the data in this way to be able to display the path on the comment using the subreddit name, instead of having to look it up for every comment. They might have included the flair text and cakeday attributes in the table as well to make styling faster (i.e. displaying each users' flair text quickly, instead of doing a lookup for every author that posted something in the comments).

3.)

The join query to be explained (an inner join):

select s.id as submission\_id, s.created\_utc, s.author\_flair\_text, c.body as body, c.id as comment\_id from comments c inner join submissions s on s.id = SUBSTR(c.link\_id, 4, LENGTH(c.link\_id) - 3)

Explain output:

== Physical Plan ==

\*(2) Project [id#27 AS submission\_id#415, created\_utc#17L, author\_flair\_text#12, body#132, id#142 AS comment\_id#417]

+- \*(2) BroadcastHashJoin [substring(link\_id#144, 4, (length(link\_id#144) - 3))], [id#27], Inner, BuildRight

:- \*(2) Project [body#132, id#142, link\_id#144]

: +- \*(2) Filter isnotnull(link\_id#144)

: +- \*(2) FileScan parquet [body#132,id#142,link\_id#144] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf\_vm-shared/comments-minimal.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(link\_id)], ReadSchema: struct<body:string,id:string,link\_id:string>

+- BroadcastExchange HashedRelationBroadcastMode(List(input[2, string, true]))

+- \*(1) Project [author\_flair\_text#12, created\_utc#17L, id#27]

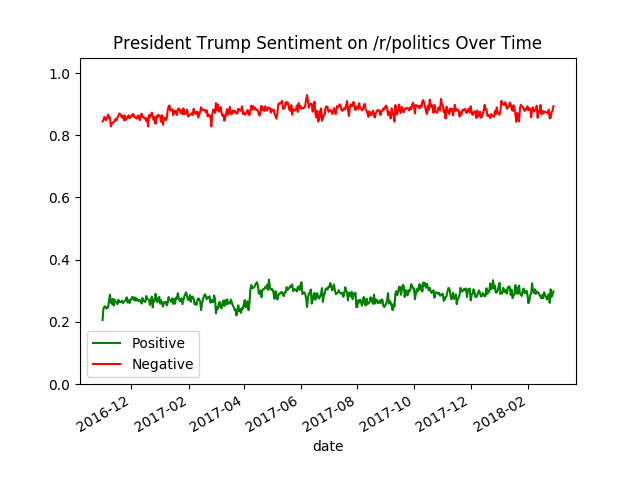
+- \*(1) Filter isnotnull(id#27)

+- \*(1) FileScan parquet [author\_flair\_text#12,created\_utc#17L,id#27] Batched: true, Format: Parquet, Location: InMemoryFileIndex[file:/media/sf\_vm-shared/submissions.parquet], PartitionFilters: [], PushedFilters: [IsNotNull(id)], ReadSchema: struct<author\_flair\_text:string,created\_utc:bigint,id:string>

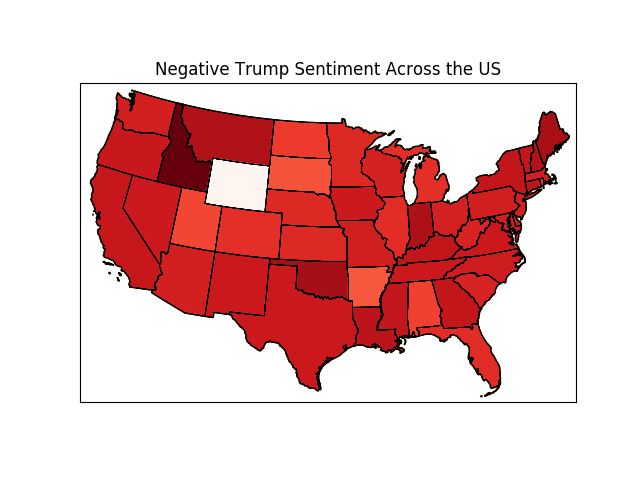
We can see from the output that Spark is using a Hash join for this inner join. It builds the hash table on the join keys that were used for the inner join. This is used when one of the dataframes is small enough to fit in the memory of a single machine. It then broadcasts the dataframe to all other cores that each have a chunk of the larger dataframe. The join is then computed at each of the cores.

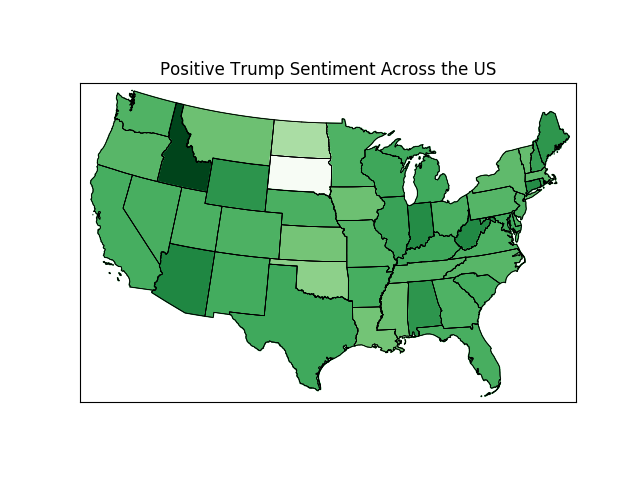
Plots:

1.)

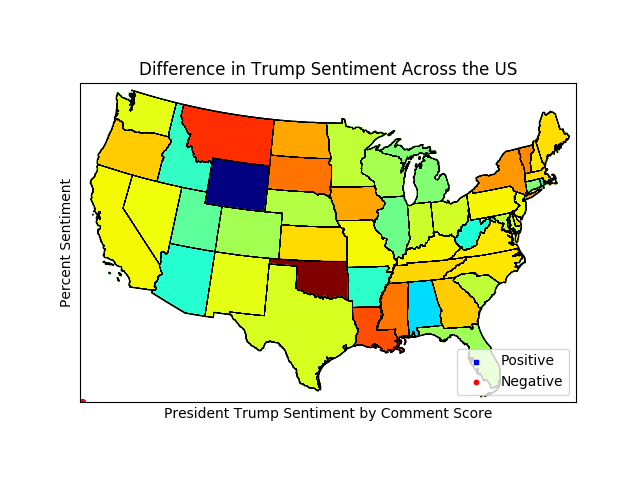


2.)

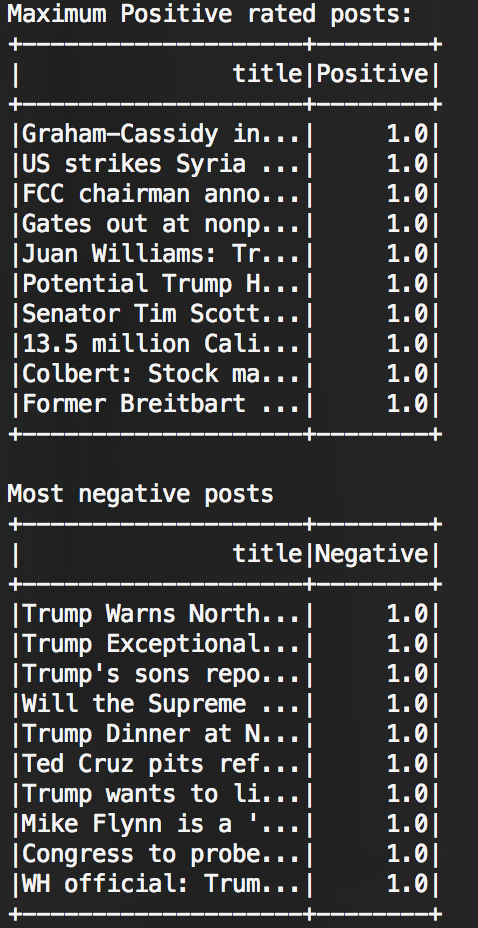




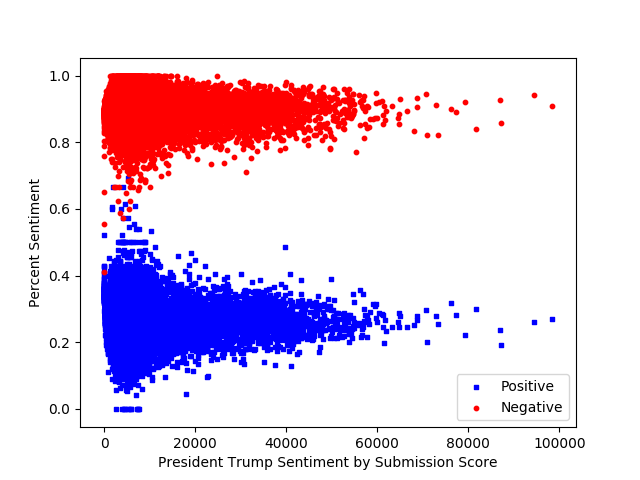
3.)

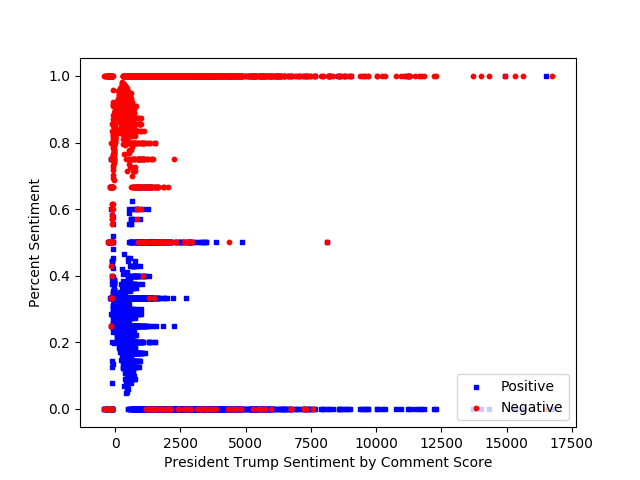


4.)



5.)





7.) 