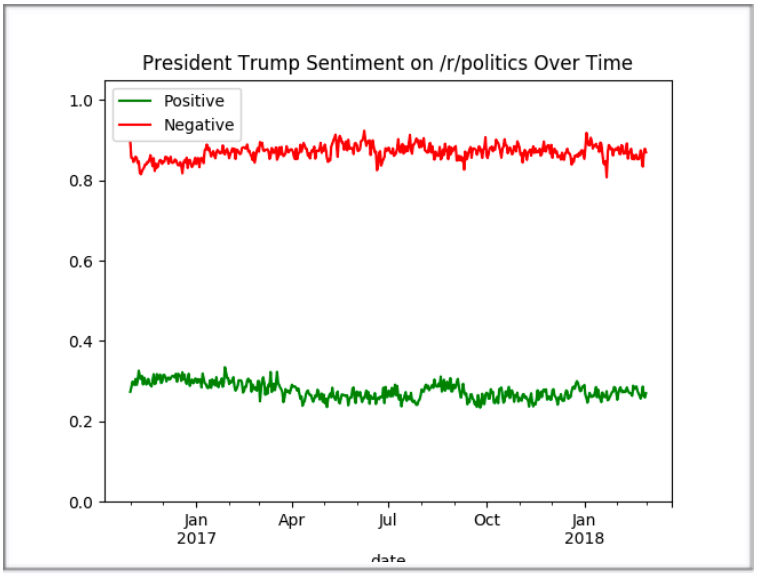
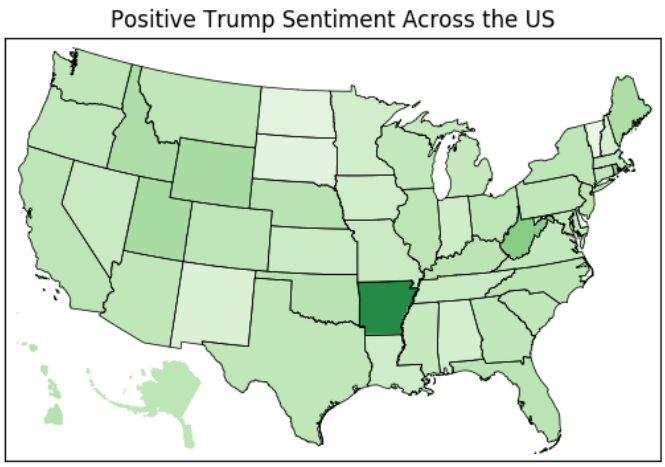
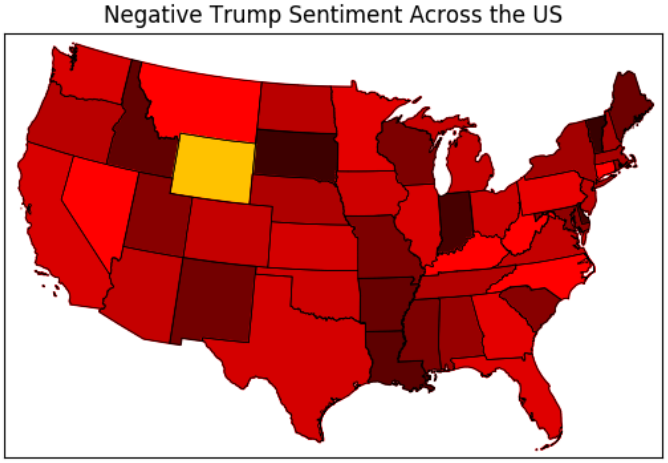
1. Time Series Plot

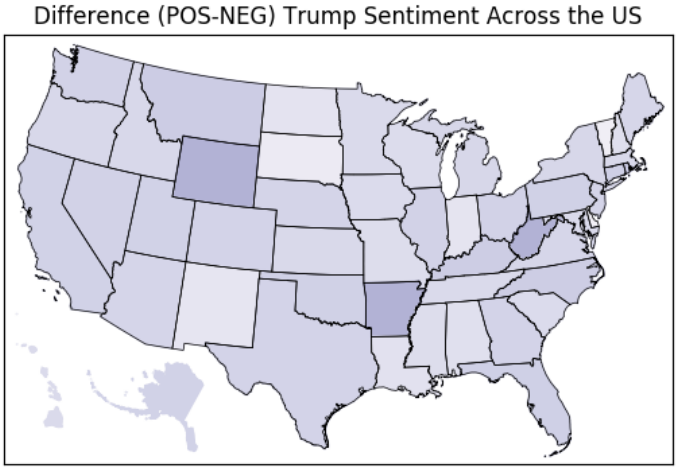


2. Positive and Negative Sentiment Maps





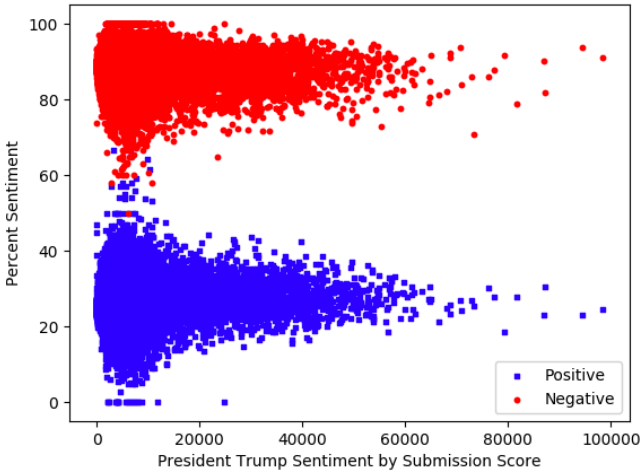
3. Difference (Positive minus Negative) Map

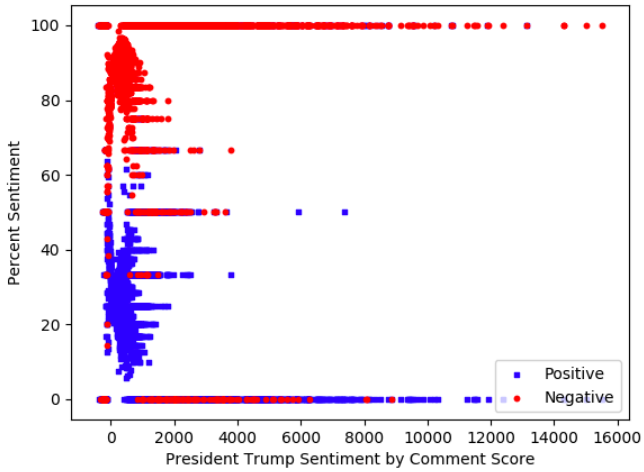


4. Most positive and most negative submissions (the lists contain the submission_id's)

most_positive	most_negative
5kyoao	69euq8
66esg4	6ardm6
5plni4	69fa14
6y8ul4	66b1nk
5w8kcd	69fdar
62bcl6	69g98r
655jy0	6afhol
639xea	6ad6cp
7paxu2	6623yq
5ncsql	6adp7i

5. Scatter Plots

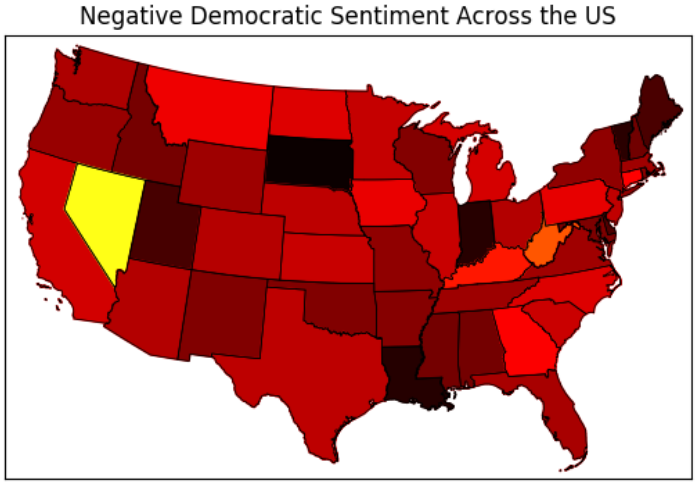


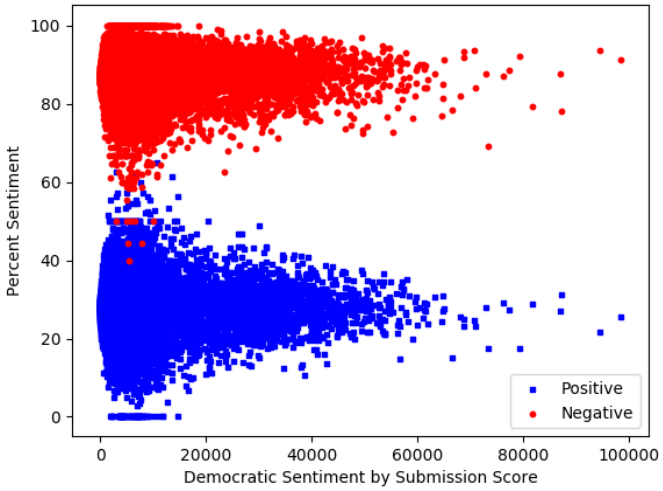


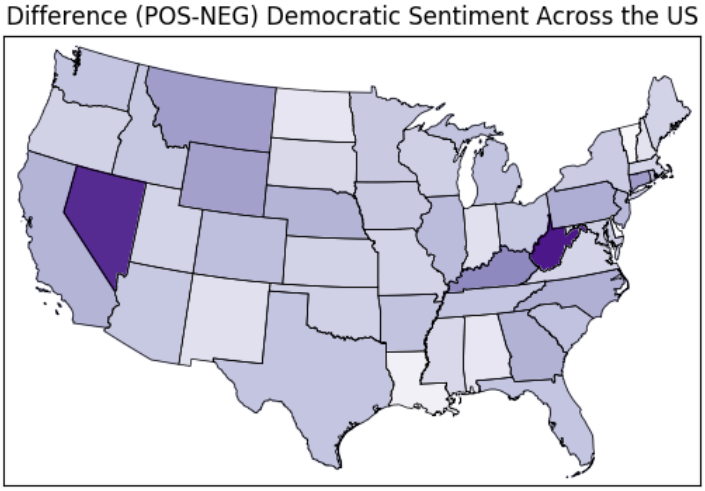
6. Extra credit plots

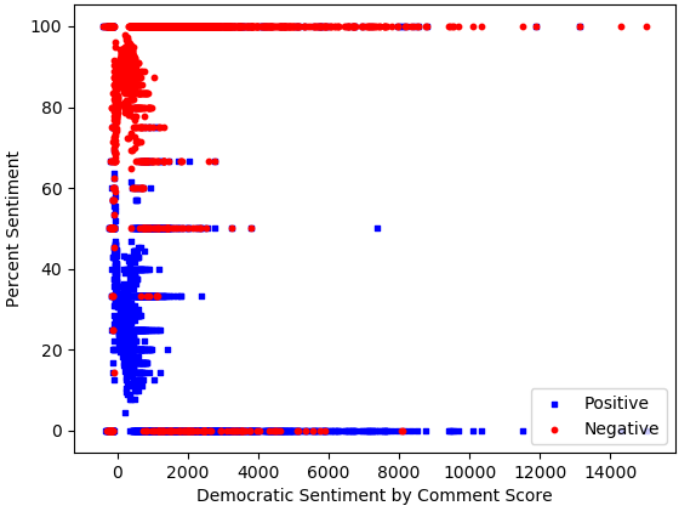
We created the same plots for the other 2 labels: democratic and GOP, in order to compare the sentiment to that expressed towards Trump.

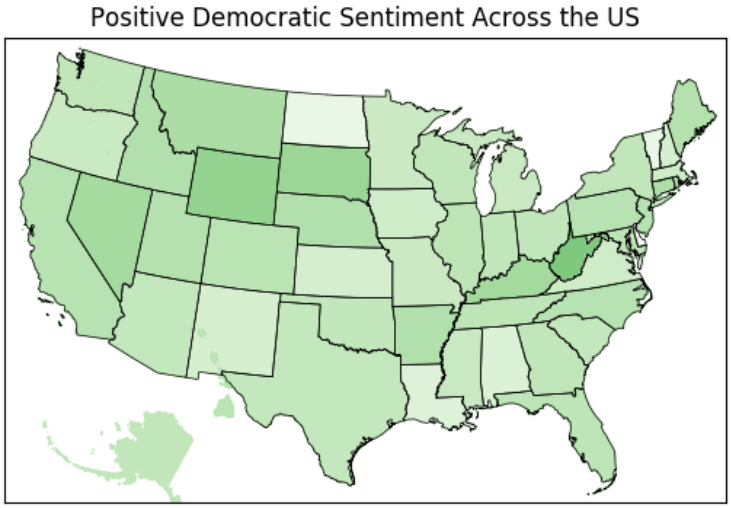
Democratic:

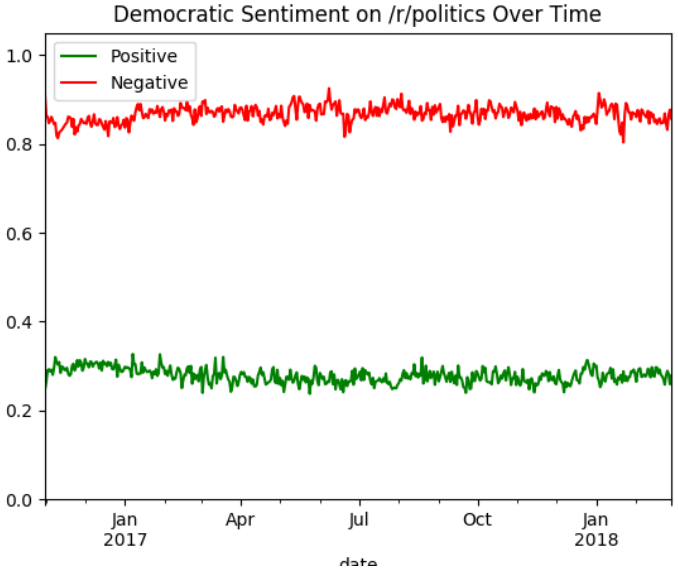

Negative Democratic Sentiment Across the US


Percent Sentiment
Democratic Sentiment by Submission Score
Positive
Negative

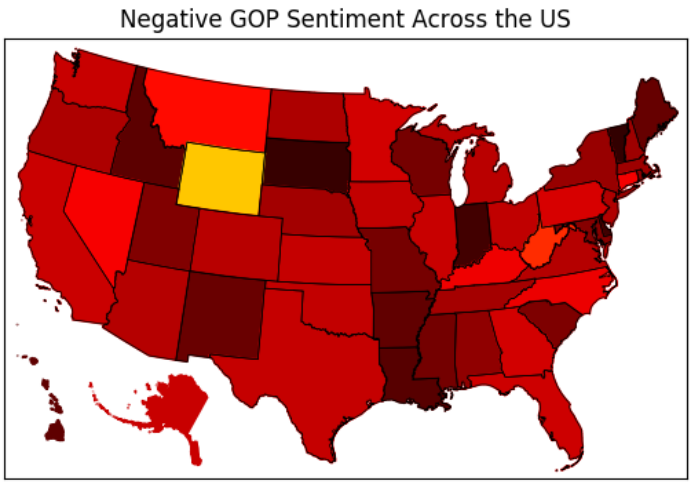

Difference (POS-NEG) Democratic Sentiment Across the US

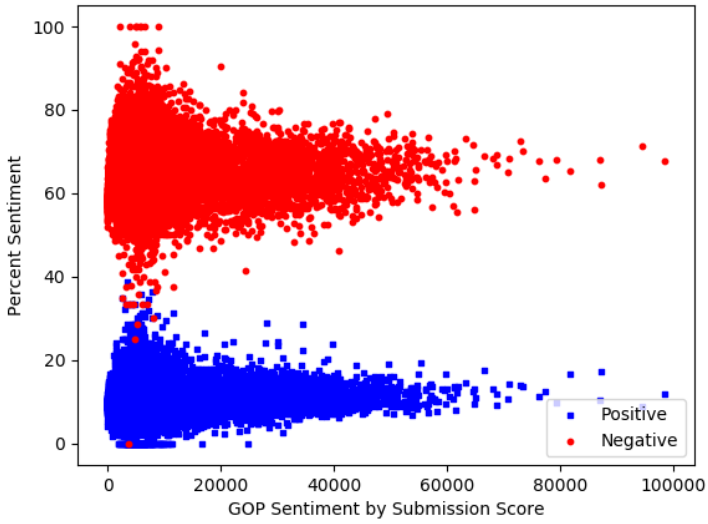

Percent Sentiment
Democratic Sentiment by Comment Score
Positive
Negative

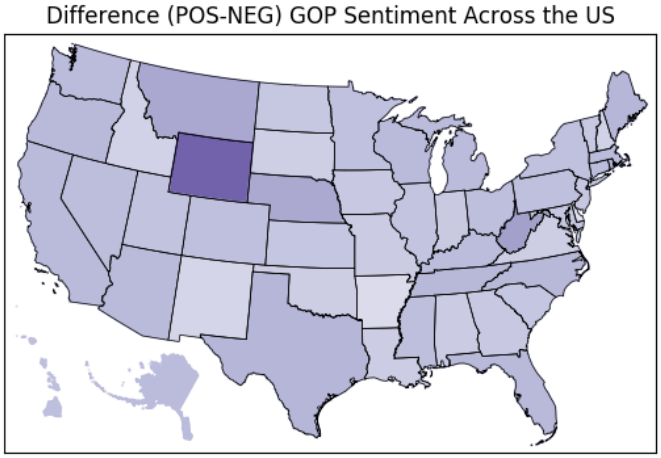

Positive Democratic Sentiment Across the US

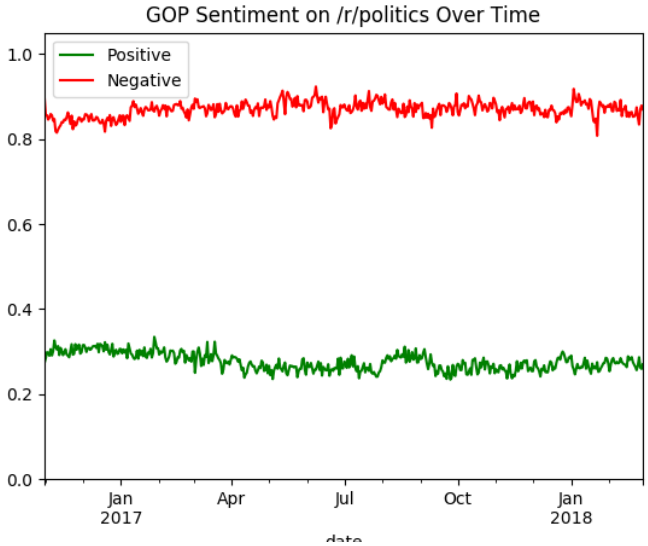

Democratic Sentiment on /r/politics Over Time
Positive
Negative
Jan 2017
Apr
Jul
Oct
Jan 2018

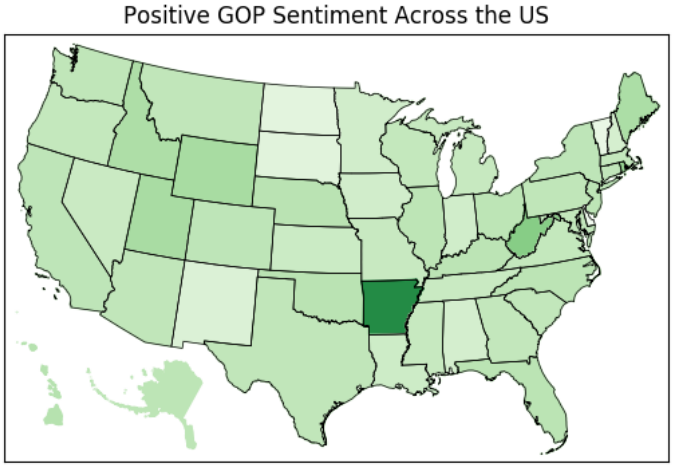
GOP:

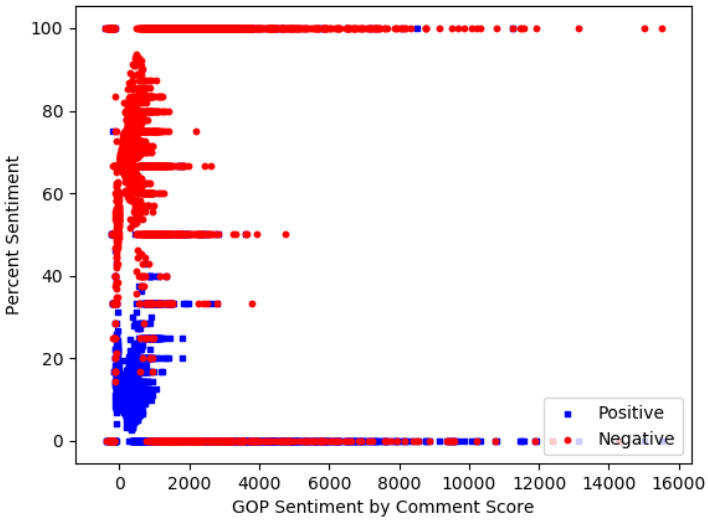

Negative GOP Sentiment Across the US


Percent Sentiment
GOP Sentiment by Submission Score
Positive
Negative


Difference (POS-NEG) GOP Sentiment Across the US


GOP Sentiment on /r/politics Over Time
Positive
Negative
Jan
2017
Apr
Jul
Oct
Jan
2018


Positive GOP Sentiment Across the US


Percent Sentiment
GOP Sentiment by Comment Score
Positive
Negative

7. ROC curve - (did not complete)

—

8. Paragraph summarizing findings

We notice that /r/politics seems to have an exceedingly negative opinion of President Trump from before he was elected to months after he was inaugurated. There is not one state whose Reddit users express more positive than negative sentiment about Trump. It is Reddit after all. By our model, Arkansas has the most positive things to say about Trump, which isn't surprising. Historically, Arkansas is as red a state as you can get, so our model seems to have gotten something right. On the other end of the spectrum, users from Delaware and Vermont, blue states, have the least positive sentiment in their Reddit comments. When we observe the sentiment towards the democratic party, the shades of quite a few of the states seem to invert. States that had more positive things to say about Trump don't have the same positive attitude towards the democratic party and vice versa. Overall, Reddit seems to be a forum for people to voice negative thoughts. This probably isn't because people are naturally pessimistic or angry, but rather because people like to challenge other people's opinions and stories and the natural sentiment that our model gleans from that is negative.

In terms of the graphs, we notice that, with time, there isn't a large degree of change in the positive/negative sentiment towards Trump. However, after Jan 2017 (Trump's inauguration), we see a slight dip in the positive score towards Trump, and a slight rise in the negative curve. This makes sense, since people realized that their optimism towards was ill-founded. In terms of submission score, we notice a large cluster of data points on the lower end of the spectrum. This is probably because these posts are highly controversial, and hence the difference in their upvotes and downvotes is not very high. We notice similar findings in terms of comment score.

3 Questions:

QUESTION 1:

Check out some rows in the labeled dataframe: labeled.show()

Functional Dependencies:

- Input.id ---> labeldem, labelgop, labeldjt

Which implies, by Armstrong axioms:

- Input.id ---> labeldem
- Input.id ---> labelgop
- Input.id ---> labeldjt

- Et al. by decomposition/association

QUESTION 2:

Checkout the schema for comments dataframe: `comments.printSchema()`

root
l-- author: string (nullable = true)
l-- author_cakeday: boolean (nullable = true)
l-- author_flair_css_class: string (nullable = true)
l-- author_flair_text: string (nullable = true)
l-- body: string (nullable = true)
l-- can_gild: boolean (nullable = true)
l-- can_mod_post: boolean (nullable = true)
…

Checkout a row in the dataframe: `comments.show(n=1)`

Is it normalized?

There are 3 distinct entities involved in the table: sub-reddit, user, comment

- The comment id is candidate key of entire relationship.
- User data like flair, cake etc. has partial dependency on author
- Subreddit data like sub name has partial dependency on the s_id

So, considering basic 2NF criteria:

- Table is not normalized.
- As a starter, can be decomposed into distinct comment, user and sub-reddit tables.
- Keyed by id, author and s-id attributes respectively.

Consider comment data:

- Candidate key is id + any trivial superset
- Functional Dependencies:
- id —> all attributes (by def of key)

- body —> link_id i.e. a transitive dependency that can be decomposed in 3NF

Why is given table not normalized?

- Typically, every rendering of a comment on a reddit page uses all the attributes in the non normalized table
- If normalized, we'll need to compute an expensive JOIN for each useful render of a comment
- So even though there is redundancy that can be removed, the data makes sense collectively - particularly on the front-end.

QUESTION 3:

We choose the join from task 8:

```
joined_2_explain = min_df.join(submissions, ["link_id"]).explain()
```

We obtained the following physical plan:

```
== Physical Plan ==
Project [link_id#36222, id#74, utc_created#36219L, body#64, state#36221, comment_s-
core#36223L, archived#110, author#111, author_cakeday#112, author_flair_css_class#113,
author_flair_text#114, brand_safe#115, can_gild#116, can_mod_post#117,
contest_mode#118, created_utc#119L, crosspost_parent#120, crosspost_parent_list#121,
distinguished#122, domain#123, downs#124L, edited#125, gilded#126L, hidden#127, ... 41
more fields]
+- SortMergeJoin [link_id#36222], [link_id#36230], Inner
   :- *(4) Sort [link_id#36222 ASC NULLS FIRST], false, 0
   :  +- Exchange hashpartitioning(link_id#36222, 200)
   :     +- *(3) Project [id#74, created_utc#70L AS utc_created#36219L, body#64, au-
thor_flair_text#63 AS state#36221, pythonUDF0#37029 AS link_id#36222, score#80L AS
comment_score#36223L]
   :        +- BatchEvalPython [<lambda>(link_id#76)], [author_flair_text#63, body#64,
created_utc#70L, id#74, link_id#76, score#80L, pythonUDF0#37029]
   :           +- *(2) Project [author_flair_text#63, body#64, created_utc#70L, id#74,
link_id#76, score#80L]
   :              +- *(2) Filter isnotnull(pythonUDF0#37028)
   :                 +- BatchEvalPython [<lambda>(link_id#76)], [author_flair_text#63,
body#64, created_utc#70L, id#74, link_id#76, score#80L, pythonUDF0#37028]
   :                    +- *(1) FileScan parquet [author_flair_text#63,body#64,creat-
ed_utc#70L,id#74,link_id#76,score#80L] Batched: true, Format: Parquet, Location: In-
MemoryFileIndex[file:/home/cs143/data/comments-minimal.parquet], PartitionFilters: [],
PushedFilters: [], ReadSchema: struct<author_flair_text:string,body:string,creat-
ed_utc:bigint,id:string,link_id:string,score:big...
   +- Sort [link_id#36230 ASC NULLS FIRST], false, 0
      +- Exchange hashpartitioning(link_id#36230, 200)
         +- Project [archived#110, author#111, author_cakeday#112, author_flair_css_-
class#113, author_flair_text#114, brand_safe#115, can_gild#116, can_mod_post#117, con-
test_mode#118, created_utc#119L, crosspost_parent#120, crosspost_parent_list#121, dis-
tinguished#122, domain#123, downs#124L, edited#125, gilded#126L, hidden#127, hide_s-
core#128, id#129 AS link_id#36230, is_crosspostable#130, is_reddit_media_domain#131,
is_self#132, is_video#133, ... 36 more fields]
            +- Filter isnotnull(id#129)
               +- FileScan parquet [archived#110,author#111,author_cakeday#112,au-
thor_flair_css_class#113,author_flair_text#114,brand_safe#115,can_gild#116,can_mod_-
post#117,contest_mode#118,created_utc#119L,crosspost_parent#120,crosspost_parent_list#
```

```
121,distinguished#122,domain#123,downs#124L,edited#125,gilded#126L,hidden#127,hide_s-
core#128,id#129,is_crosspostable#130,is_reddit_media_domain#131,is_self#132,is_video#1
33,... 36 more fields] Batched: false, Format: Parquet, Location:
InMemoryFileIndex[file:/home/cs143/data/submissions.parquet], PartitionFilters: [],
PushedFilters: [IsNotNull(id)], ReadSchema: struct<archived:boolean,author:string,au-
thor_cakeday:boolean,author_flair_css_class:string,author...
```

(It's a little weirdly formatted because Spark's output is really wide)
Spark seems to be using a SortMergeJoin, which involves sorting each table by the join key, "link_id", in ascending order with the null values first. Then it partitions the key-value pairs with a hash function on the keys to divide the data up. Then it projects the columns for each table from the corresponding parquet files. Finally it executes the merge join.