CS521 Final Project

Analysis of Reddit Comments

Nick Sun

3/10/2020

Introduction

Reddit is one of the most popular sites on the internet and hosts a variety of online communities, ranging from sports fans to horror story writers. One of the most important areas of user experience are the comment sections where threads on the most popular posts can become small communities in their own regard, rife with in-jokes and slang. Data on all reddit comments is made available as `.zst` and `.xz` files from `pushshift.io`. For my final project, I took a closer look at all Reddit comments made in September of 2019.

Unzipping the `.zst` file required downloading the Zstandard Compression Program on my GCP instance. Thankfully, this is available to Ubuntu and Debian as the `zstd` package. When zipped, the file is 15GB. After the file was unzipped, it was 167GB total. The data itself was in a JSON format and contained 45 columns, among which included:

- The author
- The subreddit
- The post ID
- The comment itself, which was free text containing punctuation, Unicode, and slang
- other relevant comment data like scores, awards, controversiality status, date of creating, etc.

Wrangling Overview

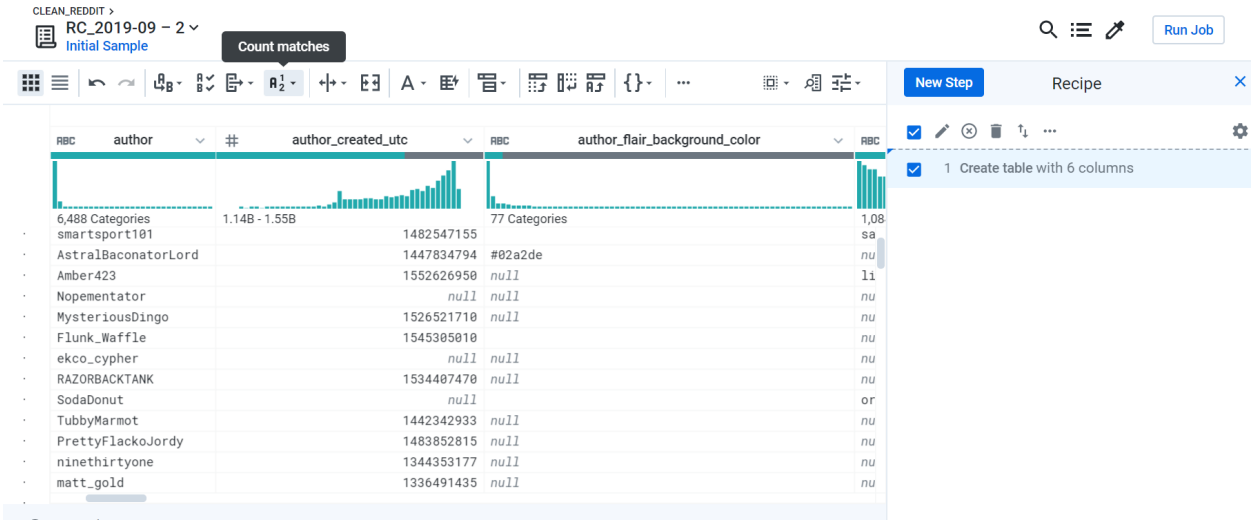
Overall, not much had to be done in terms of programmatic wrangling for this dataset. There were a lot of NULL values in the data, but they were contained inside of columns that I ended up stripping out in Dataprep. The biggest issue with wrangling this data was its size. Several GCP instances of progressively larger and larger disk space were created until the appropriate size was discovered to hold the unzipped file. In the end, an instance with a disk space of 200GB was sufficient to `wget` the `.zst` file and hold its unzipped contents.

I attempted to clean the free text body of the comments by removing punctuation and Unicode, such as emojis, but I in the end decided that removing these characters might be removing relevant data that identifies each subreddit community.

Loading the Data: Dataprep and BigQuery

Thankfully, this project was similar to previous assignments we had in this class, with the most significant difference being the size of the dataset. Once the data was unzipped, it was relatively straightforward to use `gsutil -m cp` to move that file into a storage bucket.

Once in the bucket, I used **Dataprep** to select just a few columns of interest. The recipe I used was pretty straightforward and is provided below:



I only kept the columns that were directly related to the queries I planned on executing. Those columns were:

- Author (string)
- Body (free text)
- Subreddit (string)
- Score (numeric)
- Awards (Gold, Silver, Platinum, etc.)
- Controversiality (0 or 1)

This simple Dataprep job took approximately 30 min to run on this dataset. Once it completed, I loaded the filtered data from the preassigned staging bucket into BigQuery.

Below is a screenshot of the dataset loaded into BigQuery. In September 2019, there were 137,540,219 comments total resulting in a usable dataset of 26GB after I selected only the columns that I needed.

Table ID	final-reddit-analysis:reddit_comments.reddit_comments
Table size	26.12 GB
Number of rows	137,540,219
Created	Mar 9, 2020, 7:43:39 PM
Table expiration	Never
Last modified	Mar 9, 2020, 8:15:36 PM
Data location	US

Sample

Since the data itself is structured in a JSON format, it is possible to read a sample of it into `pandas` as a dataframe. The first row of some data I sampled is pictured below:

```
In [69]: df.iloc[0]

Out[69]: author                                                    Dethcola
         author_flair_css_class
         author_flair_text                                        Clairemont
         body                                                       A quarry
         can_gild                                                       True
         controversiality                                                  0
         created_utc                                              1506816000
         distinguished                                                  None
         edited                                                            0
         gilded                                                            0
         id                                                           dnqik14
         is_submitter                                                   False
         link_id                                                    t3_73ieyz
         parent_id                                                  t3_73ieyz
         permalink                /r/sandiego/comments/73ieyz/best_place_for_gra...
         retrieved_on                                             1509189606
         score                                                             3
         stickied                                                       False
         subreddit                                                   sandiego
         subreddit_id                                               t5_2qq2q
         author_cakeday                                                 NaN
         word_count                                                       8
         Name: 0, dtype: object
```

This sample data is also provided alongside this submitted report as a JSON file.

Analysis Questions

The questions I chose to tackle were fairly broad. In the month of September, 2019:

1. What are the most active subreddits?
2. Where should a redditor spend the most time if they want to maximize karma?
3. Which subreddits are the most controversial?

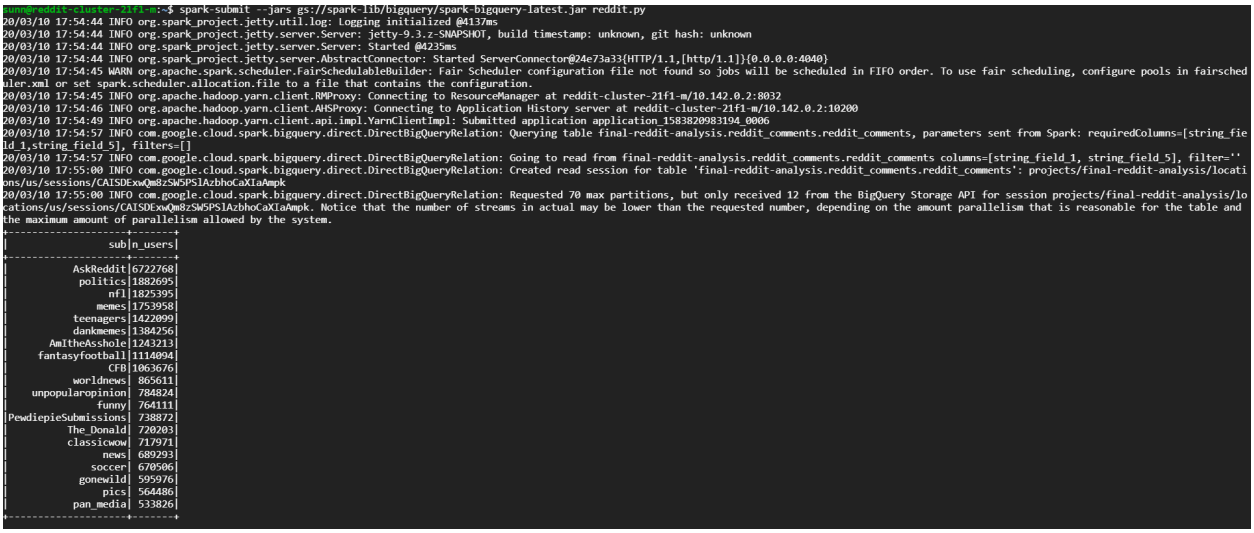
1 is a fairly straightforward questions. I decided the best way to do it would be to use **spark.sql** to run a query against the entire dataset, counting up the total comments per subreddit, then reporting the most popular.

I created a `pyspark` script containing the appropriate query and submitted in the master node of my Dataproc cluster using the `spark-submit` command. The `pyspark` script that I used was modeled heavily after the `spark.sql` example provided in the documentation.

```
#!/usr/bin/python
from pyspark.sql import SparkSession

spark = SparkSession \
            .builder \
            .master('yarn') \
            .appName('final-reddit-analysis') \
            .getOrCreate()

# Use the Cloud Storage bucket for temporary BigQuery export data used
# by the connector.
bucket = "reddit-temp"
spark.conf.set('temporaryGcsBucket', bucket)

# Load data from BigQuery.
comments = spark.read.format('bigquery') \
        .option('table', 'final-reddit-analysis:reddit_comments.reddit_comments') \
        .load()
comments.createOrReplaceTempView('comments')
```

```
# Perform word count.
comment_query = spark.sql(
    'SELECT string_field_5 as sub, COUNT(string_field_1) as n_users FROM comments GROUP BY sub ORDER BY
)
comment_query.show()
comment_query.printSchema()

# Saving the data to BigQuery
comment_query.write.format('bigquery') \
```

The output of this script was printed to stdout and is provided below. The pyspark script I ran is included in this submission.

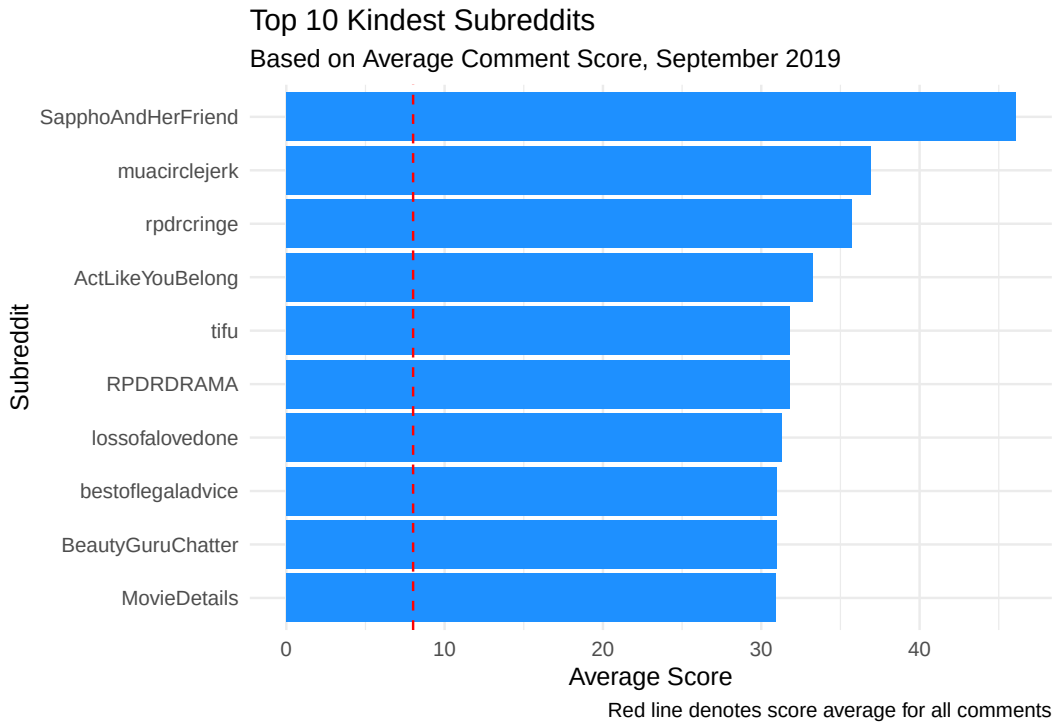


Not surprisingly, the most active subreddits are also the most popular subreddits! Almost all of the top subreddits appear regularly in the front page of reddit `r/all`, and those that don't are famous (or infamous) for having very active communities, for example, `r/T_D` which is a very controversial subreddit dedicated to Donald Trump and `r/PewdiepieSubmissions` which is a community based around the humor of popular youtube Pewdiepie.

2 is all about maximizing the karma (aka "worthless internet points") a user can get by commenting in particular subreddits. I computed average score for each of the subreddits using **BigQuery** and then sorted the subreddits based on this average. The query to perform this is:

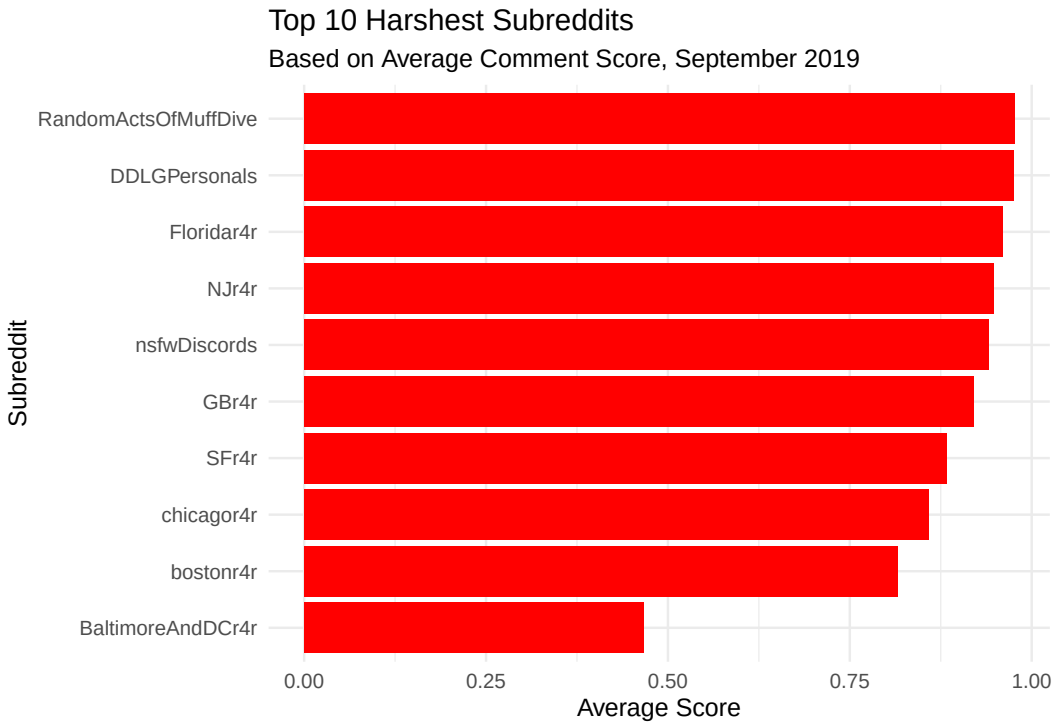
```
SELECT sub, avg_score, comment_count
FROM (
  SELECT string_field_5 as sub,
    AVG(int64_field_4) as avg_score,
    COUNT(string_field_1) as comment_count
  FROM reddit_comments.reddit_comments
  group by sub
)
WHERE comment_count > 1000
ORDER BY avg_score DESC, comment_count DESC
LIMIT 10
```

I output the results as a .csv file and created a barplot using `ggplot`.



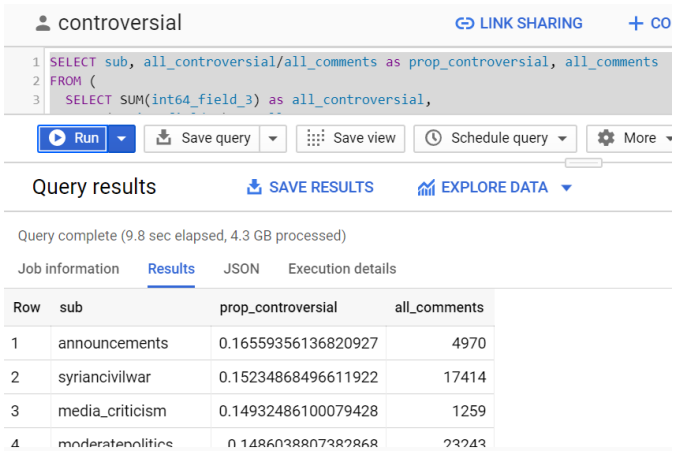
Interestingly, the kindest subreddits are communities centered around female or traditionally feminine topics. For example `r/SapphoAndHerFriend` is a lesbian empowerment subreddit, `r/muacirclejerk` is for makeup, and any subreddit with `rpdr` involves RuPaul's Drag Race (although as a fan, I can attest that the show doesn't strictly cater to women).

Since this BigQuery job was inexpensive to run, I also decided to find the subreddits with the lowest average comment scores, modifying the `ORDER BY` clause.



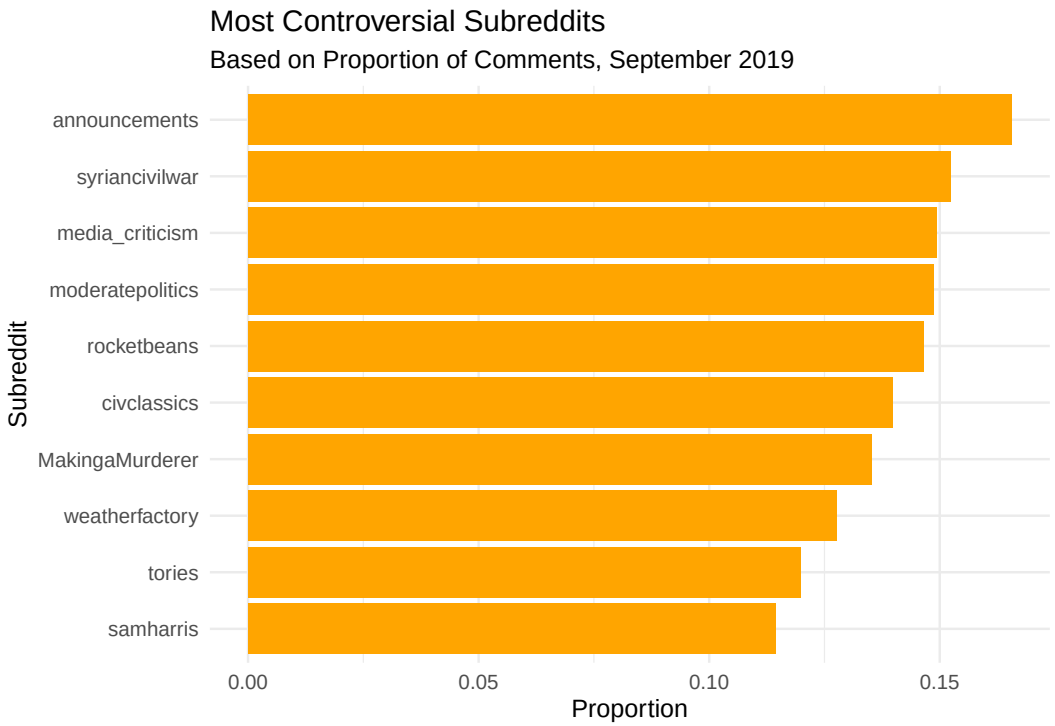
Less surprisingly, the subreddits with the lowest average scores are for the most part relatively mundane and elicit no strong feelings one way or the other. Any subreddit that has `r4r` is about ridesharing and carpools in different cities. The exceptions to this are some unpopular **NSFW** subreddits, which understandably might not engender strong positive reactions.

3 focuses on computing the proportion of comments in a subreddit that are considered *controversial*. For a comment to be controversial, it has to have a large number of votes with a relatively equal balance of upvotes and downvotes. The query used to calculate this as well as a screenshot of the output in BigQuery is provided below:




```
SELECT sub, all_controversial/all_comments as prop_controversial, all_comments
FROM (
  SELECT SUM(int64_field_3) as all_controversial,
  COUNT(string_field_1) as all_comments,
```

```
  string_field_5 as sub
  FROM reddit_comments.reddit_comments
  GROUP BY string_field_5
)
WHERE all_comments > 1000
ORDER BY prop_controversial DESC
```



While some subreddits are not surprising given the subject matter e.g. the Syrian Civil War, criticism of mainstream media, politics, and **Making a Murderer** which is a docuseries about a man framed for murder, there are some surprising additions to this list. For example, the most controversial subreddit is `r/announcements` which is a meta-subreddit about new features and changes to the website itself. Video games also apparently represent controversial topics, for example, `r/civclassics` is about a Minecraft server, `r/weatherfactory` deals with a gaming company, and `r/rocketbeans` deals with German gaming youtubers.

The BigQuery outputted tables for all of these queries is provided in the zip file.

Conclusion

This was a really fun project to work on and encompassed a lot of what I find fun about big social data. I will be playing around with this dataset and likely adding to it. Pushshift also has data dumps of reddit posts that would be interesting to join with this comments dataset, and future analyses could be done like regression modelling average comment score to post score.