PART A: DATA GATHERING

While several Reddit related datasets exist, we chose to create our own dataset for a variety of reasons. Since we selected only the r/TIFU subreddit we wanted a significantly large dataset that would reflect only posts in this subreddit. Existing Reddit datasets contain a month’s worth of data for the entire website, resulting in files in the range of several gigabytes of which only a few megabytes would be relevant to the problem. Additionally, these datasets would rarely include user profiles, a source of data that we would pull additional features from. Instead of downloading hundreds of gigabytes of data and then scraping the associated the user profiles, we decided to create a scrapping tool in Python to pull threads from various Reddit APIs.

A community built and maintained Python library, PRAW (Python Reddit API Wrapper) was almost perfectly suited for this task. Using PRAW, we were able to get a listing of all posts in the subreddit, pull all of their data and get the associated author data. One major issue we ran into with this approach was the limitation of the Reddit API implementation. Since PRAW simply wraps the existing Python REST API, it experiences the same limitations that exist with any other way of directly accessing Reddit data, namely the inability to request more than 1000 “things” (Reddit’s terminology for any kind of listing of posts, comments, users, etc).[[1]](#footnote-1) Having access to 1000 posts only yields roughly the last 4 days of posts, which is insufficient data to train most models. The solution to this problem was the use of a community run, purpose built data science Reddit API, Pushshift.io.[[2]](#footnote-2) Pushshift differs from the existing Reddit API because it ingests Reddit data regularly, creating its own dataset from which items can be queried much more efficiently. Although Pushshift also limits us to 1000 results at once, we can specify a date range for our query. Specifying a date range of one day and a maximum result size of 1000, we can see all the submissions for any given day. By progressively going back in time day by day, we can create a collection of all submissions going back several years.

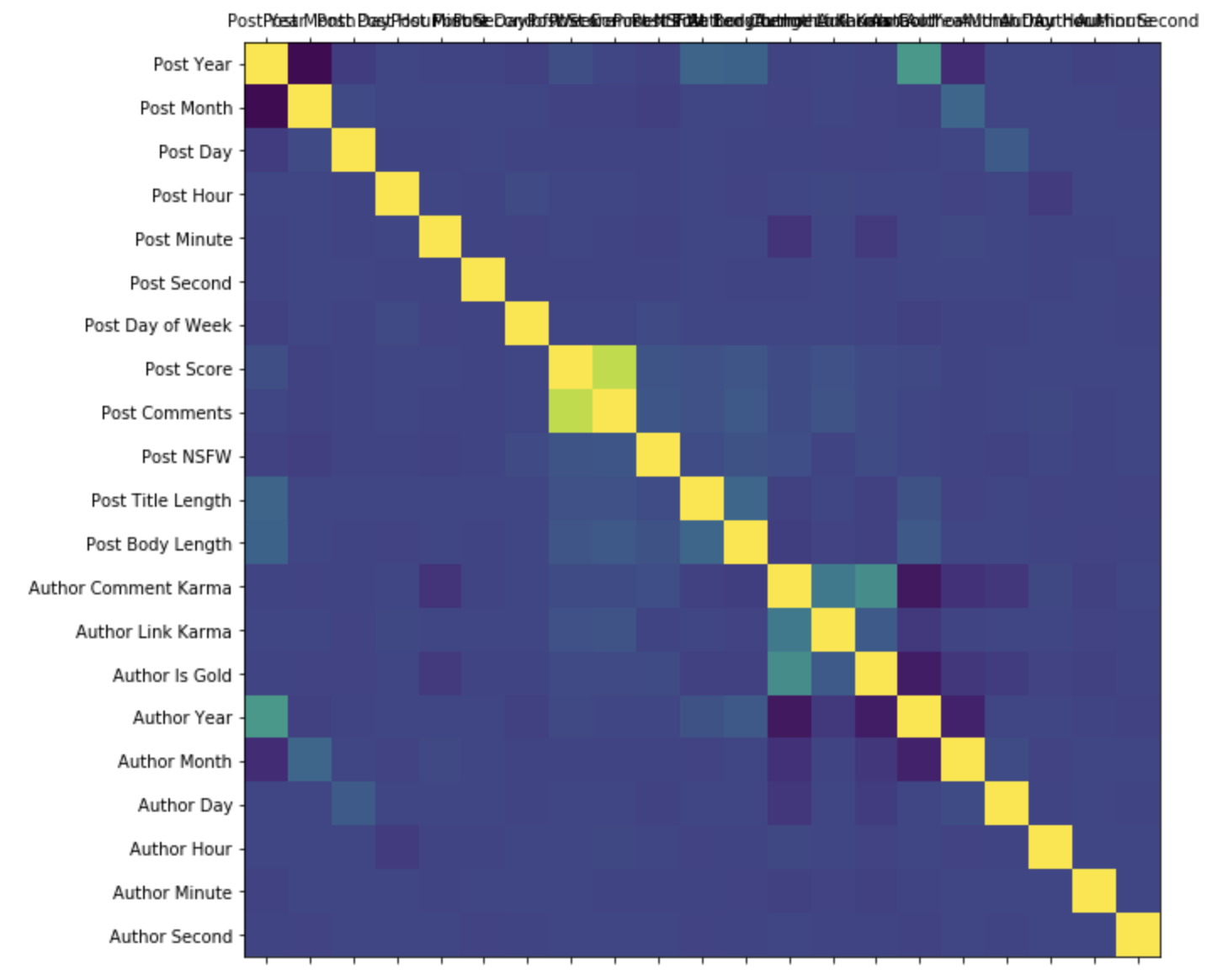
Pushshift’s disconnected dataset allows us to make more computationally challenging queries, unfortunately the data is never updated. If a post is cached by Pushshift with one point, it will not increment the point count in step with Reddit. This means that the post score, along with almost every other metric about the post, can (and usually is) out of date. Additionally, Pushshift does not have any way of accessing user data. To remedy both of these issues, a hybrid approach was undertaken. Pushshift would be used to grab the listing of posts for each day, and then two PRAW calls would be made to get the current value of the post or author using the IDs provided by Pushshift. This approach would allow us to get up to date data as well as a complete listing of posts going back to April 2015. One unresolvable issue was the Reddit API call limit of 60 requests per minute, restricting us to processing 30 posts per minute as each post requires two API calls. This API limit resulted in our collection script running 24 hours a day for approximately three days to collect around 60,000 submissions. We normalized the title and body text, changing all words to lower case and removing any punctuation or non-ascii characters.

The final result of our data scraping included data about the post such as the title text, body text, score and date; as well as information about the author such as their registration date, and sum of their previous post’s scores. We believed that if an author was successful on Reddit in the past, they would be more likely to know how to craft an interesting post. Additionally, we pulled information on the post such as the “over\_18” tag as we notice there were significantly more 18+ posts on r/TIFU than on the rest of Reddit leading us to believe that people like reading inappropriate content more.

PART B: FEATURE SELECTION

Initially, we expected to use all of the data collected. The assumption was that there would be few strong correlations between any one feature and the score, but the sum of correlations between all features and the score would be sufficient enough to make some sort of prediction on the success or failure of the submission.

Plotting a correlation matrix of the data harvested did not immediately yield any strong correlations between post and author metrics and the score of the post (except of course the number of comments, which we did not count as a feature).



At the very least, it appeared that the length of the title and body as well as the 18+ tag had some weak correlation to the score. With respect to the data about the author, there seemed to be an even weaker correlation to the score.

Additional processing was required to make use of the text in the post. Initially, we tried the naïve approach of using SKLearn’s CountVectorizer to transform each word in the title and body into a feature, with the value of the feature being the number of occurrences of that word in the post. This caused our feature count to balloon to over 15,000. It was clear that this would be ineffective in helping train our model, so we attempted to limit the number of features by looking at bi-grams, two word pairs, that appear in at least 0.1% of titles, or 1% of the body text. These minimum limits were high enough to ensure that we only picked up the most popular bi-grams, but low enough that we would expect at least a few posts to contain the bi-grams.

The ideal text processing for submissions would be identifying the subject of the post, and using the subject to try and identify what subjects are popular. Unfortunately, such text analysis is beyond the capabilities of the authors and outside of the scope of this paper. As a substitute, a rough estimate of the subject was calculated by looking at the occurrences of certain sets of words that would imply a subject. For example, seeing the words “boss” and “office” suggest that the story takes place in an office. The English language poses a challenge to this, since “boss” and “bosses” would be two different words. To solve this, we took the stem of each word turning “bosses” and “boss” both into the root word of “boss”.

This resulted in three datasets: one with no text features, one with bi-gram text features, and one with categories derived from the text.

1. https://www.reddit.com/r/redditdev/comments/30a7ap/does\_reddit\_api\_limit\_total\_listings\_returned\_to/cpqj883/ [↑](#footnote-ref-1)
2. https://pushshift.io/ [↑](#footnote-ref-2)