ELE/COS 381 Final Report

What makes a good Reddit Post?

Alan Chen, Eric Chen, and Luis Gonzalez-Yante

January 17, 2017

1 Introduction

In ELE/COS 381, we have studied various networks both physical and digital. In particular, Chapter 8 of Networked Life focused on the study of topology and functionalities of social networks like Facebook and Twitter. In this project, we are interested in the study of Reddit and what specifically makes a good Reddit post.

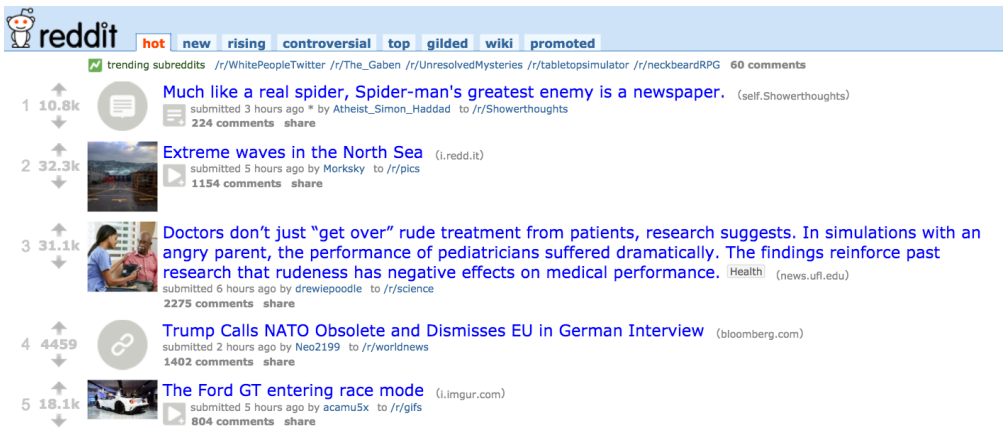


Figure 1: The frontpage of reddit.com.

Reddit is a bulletin board of user-generated content. As opposed to strictly ordering results by date, Reddit's user interface strongly focuses on showing users content that is popular with other users. The site does this through a voting mechanism, where each user can upvote and downvote specific posts to the site. For individual users participating on the site, there is a motivation to create posts that resonate with other users, and thus generate a large number of upvotes.

Another key aspect of Reddit is its emphasis on sub-communities, which are each their own fiefdom on Reddit. According to redditmetrics.com, on January 14, 2017, there were 1,005,275 unique subreddits.

Communities are organized around topic or interest. Some communities such as /r/pics and /r/news are very broad and general interest communities. Other communities appeal to much smaller groups, such as /r/vexillology, which is dedicated to discussion and commentary about flags.

It is important to note that a large component of the Reddit community does involve commenting. Comments, like posts, can be upvoted and downvoted, and comments are sorted by popularity. However, in this project, we decided to focus only on analyzing posts–which subreddits they are in and how many upvotes they receive–to simplify analysis. Further and more extensive work would likely include the study of comments, as well as analysis of the content of posts and comments.

2 Goals

We decided to look at three different methods of quantifying what makes a good Reddit post. Reddit contains many emergent phenomena not explicitly designed for in the site and not completely obvious to new users. The site can be difficult to approach from the outside, but from our experience with the site (and one shared by the many frequent users of the site), we believe Reddit can be a funny, insightful, and engaging online community. Thus, ultimately our goals from this analysis were to be able to provide a set of conclusions and actions that might be given to a new user of the site in a "How-to use Reddit" guide.

For questions addressed in sections x.1 and and x.3 that follow, we are interested in characteristics of different types of subreddits. We decided on three types: top 100, small ($< 10,000$ subscribers), and original content.

Top 100	**25 Small ($< 10,000$ subscribers)**	**10 Original Content**
AskReddit	lexington	OCPoetry
pics	improv	ReadmyStory

Figure 2: Three categories of subreddits analyzed in x.1 and x.3

Top 100 subreddits are the biggest 100 subreddits based on the number of subscribers. The largest subreddit on the site is /r/AskReddit with over 15 million subscribers, and the 100th biggest subreddit has just under 500,000 subscribers. In contrast, we sampled 25 small subreddits that were randomly selected among subreddits with less than 10,000 subscribers. Finally, we hand selected 10 subreddits that contained significant amounts of original content, where we defined original content as a post the user creates the material of the post and is not aggregating external content. It is important to note that the original content subreddits had subscriber counts more similar to the small category, as well.

A particular user will often follow general-interest subreddits, but may also follow one for her municipality or for a niche RPG that she plays. So these three categories of subreddits provide a broad perspective of both general-interest and niche communities that we believe accurately reflects the usage patterns of users on the site.

2.1 Quadrant Analysis: How much engagement do top posts get?

Given the focus of Reddit on distinct communities, and with the over 1 million unique subreddit communities on the site, we expect that there will be large variations in communities. There might be countless ways to quantify the differences, but we decided on examining subreddits along two dimensions: individual engagement and community engagement.

Individual engagement is defined as follows for any subreddit:

$$\text{individual engagement} = \frac{\sum_{u \in \text{top users}} \frac{u_{\text{posts in this subreddit}}}{u_{\text{total posts}}}}{|\text{top users}|}, \quad \text{individual engagement} \in [0, 1].$$

For example, we might look at the top posts of /r/politics and generate and individual engagement score of 0.7. We calculate this number by looking at the users who author the top posts in a subreddit. For each user, we examine their post history, seeing how many posts are in /r/politics and how many posts are in other subreddits. We then calculate the proportion of posts that are in /r/politics for each user, and then average across all of the users who author the top posts in /r/politics to generate the individual engagement for the subreddit. A value of 1 for a given subreddit means that users are very engaged in that subreddit–they only ever post there. Whereas a value of 0.05 means that the top users in that subreddit only post to that subreddit 5% of the time.

Community engagement is defined as follows for any subreddit:

$$\text{community engagement} = \frac{\sum_{p \in \text{top posts}} \frac{p_{\text{upvotes}}}{\text{subreddit subscribers}}}{|\text{top posts}|}.$$

The intuition for the metric is as follows: if the top posts in subreddits A and B both receive 1,000 upvotes on average, but if A has 1 million subscribers while B has only 100,000 subscribers, then subreddit B has 10x the community engagement of subreddit A. Similar to individual engagement, community engagement is a metric that is calculated for any particular subreddit. For the top posts in the subreddit, we normalized the upvotes with the number of subscribers to that subreddit. Then, we averaged over all the top posts in the subreddit.

Our intention for creating individual and community engagement based on the strong suspicion that some subreddits, perhaps those of a more niche topic, attract higher individual engagement, while general interest subreddits might generally have lower individual engagement. Additionally, because community engagement corresponds to what proportion of the community a top post needs to appeal to, we thought it reasonable that it's easier to create a top post in a subreddit with low community engagement because you have to appeal to a smaller fraction of the subscriber base.

2.2 Cross-pollination between subreddits: Measuring weighted link importance

Reddit users are, like most people, multidimensional and likely to frequent multiple subreddits. As they move between communities, they bring with them the context of the different communities they participate in. This context could take the form of knowledge of memes or inside jokes.

Because subreddits are composed solely through contributions to the subreddit, we implemented a metric that takes the top posts at the current time of the subreddit, and analyzes all the posts of the authors of those top posts–who we call top users–to see how often they post are in the original subreddit a, and how often they post in the target subreddit b.

$$\text{participation}_{a\to b} = \frac{\sum_{u\in\text{top users}} 4\cdot \frac{u_{\text{posts in } a}}{u_{\text{total posts}}}\cdot\frac{u_{\text{posts in } b}}{u_{\text{total posts}}}}{|\text{top users}|}, \quad \text{participation}_{a\to b}\in[0,1].$$

By taking the concentration of posts in the first subreddit, taken as a representation of the degree the author is a member of the first subreddit, and multiplying by the concentration of posts in the second subreddit, taken as a representation of the degree the author is a member of the second subreddit, we receive a metric describing the crossover between subreddits for a given author. This participation function is maximized when a user participates 50% in subreddit a and 50% in subreddit b and contains an extra factor of four to possible values between 0 and 1. The metric for the subreddit is the average of the values for its top users.

Modeling subreddits as nodes in a network is an extension of Chapter 8: How do I influence people on Facebook and Twitter. An important notion from that chapter is that some links between users more important than others, such as links that were included in many shortest paths. Here, we extend that intuition to our participation metric which generates a network of weighted links, and we use the weights to directly infer which links are important in the network representation.

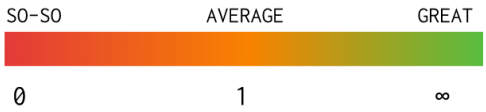
2.3 The Reddit Power Index: How "average" are top users?

Because of the lack of real names on Reddit, it is not immediately clear who are the "top users" and who are not. For example, on Twitter, the users with the most followers are often celebrities and public figures. But who are these top users of Reddit, and how dominant are they? Specifically, is it possible for the average user of Reddit to have a post become very popular in a subreddit? Or are the frontpages of subreddits controlled by an elite group of users?

To answer these questions, we were interested in quantifying how good the top users are. To do so, we created the a metric called the Reddit Power Index (RPI). Fundamentally, RPI is a metric that can be calculated on any post, and is defined as follows:

$$\text{RPI}_{\text{post}} = \frac{\text{post upvotes}}{\text{subreddit avg. upvotes}}.$$

Thus, the RPI for any post is the number of upvotes it has divided by the average number of upvotes for a post in that subreddit. An RPI below 1 means that the post is below average, and any RPI above 1 means that the post is above average.



Because we are interested in categorizing subreddits as a whole, we extend the definition of RPI first to users and then to subreddits as follows:

$$\text{RPI}_{\text{user}} = \frac{\sum_{p \in \text{posts}} \text{RPI}_p}{|\text{posts}|},$$

$$\text{RPI}_{\text{subreddit}} = \frac{\sum_{u \in \text{top users}} \text{RPI}_u}{|\text{top users}|}.$$

Naturally, the RPI for a user is the average RPI of its posts, and the RPI for a subreddit is the average RPI of its top users.

With RPI, we created a heuristic which indicates the difficulty of creating a top post in any given subreddit. A subreddit with high RPI is one dominated largely by users who consistently have successful posts. Whereas a subreddit with lower RPI is in some sense more "democratic" because the community surfaces posts from users who are closer to the average Reddit user in terms of past post upvote performance.

3 Implementation

We collected our data using Python scripts and the Python package PRAW, the Python Reddit API Wrapper. The wrapper authenticates using OAuth, creating a Reddit instance that contains the client_id, client_secret, password, and username, which the rest of the PRAW API acts upon. While Reddit has its own public API that exposes data through JSON, we preferred using PRAW as an intermediary because it handled authentication, rate-limiting, and exceptions, and allowed us to focus on writing the data collection scripts.

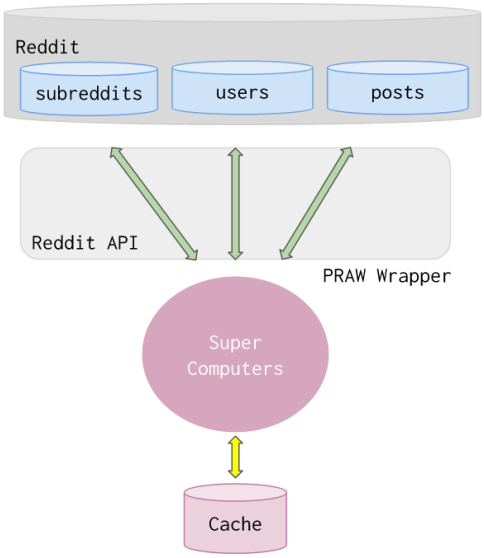


Figure 3: Diagram of our data collection process.

The Reddit Instance generated by PRAW can access any subreddit, post, or user that is accessible through reddit.com. Additionally, PRAW generates iterables which are very useful for data collection. For example, we can create a PRAW object that is a list of all of the current top posts in a subreddit, and this object can be iterated through like any other list in Python. We then can get key data for each submission, such as author, upvote count, and title. Similarly, each Reddit user can generate a PRAW object which is a list of its recent posts. This technique of (a) iterating through the top posts in a subreddit, (b) finding the upvotes and author of each top posts, and (c) finding the history of posts for that user forms the foundation of our data collection techniques.

While PRAW simplified using the Reddit API, we still encountered significant roadblocks in regards to rate-limiting. Each request to the API can return a maximum of 100 posts at a time, and PRAW delays 2 seconds between API requests. In all of our scripts, there was the trade off where collecting more data resulted in longer execution times. We were able to strike a reasonable balance by often choosing to look at samples of top posts of size 25 to 100.

However, when collecting data for the RPI, these rate-limits actually became a bottleneck factor (script execution time took hours). Our issue was that calculating the RPI for each user often requires collecting average upvote data for dozens of subreddits, each requiring a scraping of random posts that took several seconds. We largely overcame the issue by caching the average upvote value for subreddits and storing this data in a CSV file so it was persistent across different executions. Then, for example, if we had already scraped /r/AskReddit before, our code gets the average upvotes from the local cache instead of hitting the API, which makes a multi-second operation essentially free.

Also, we should note that many subreddits closely follow power-law distributions for the number of upvotes on a random post, which had a noticeable impact on our sampling results.

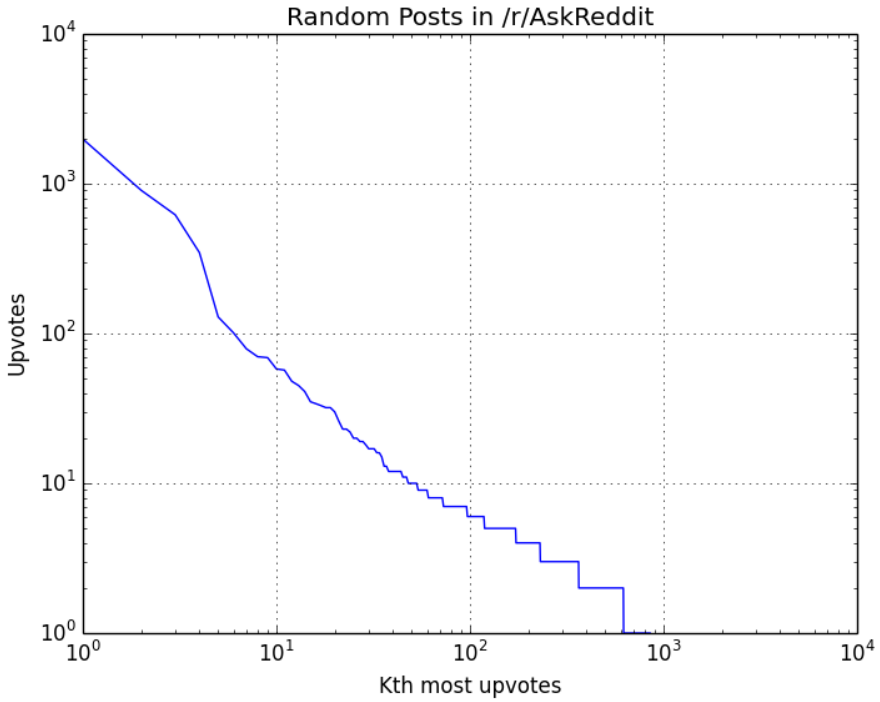


Figure 4: Log-log plot of upvotes for 1,000 random submissions to /r/AskReddit, a Top 100 subreddit.

Thus, because rate-limits constrained the number of posts we could sample, there was the potential for statistics to be moved wildly by sampling one outlier post out of 100. We believe that the reason for power-law distributions for upvotes on Reddit posts is because of how Reddit surfaces content to users. By default, users of the site are shown posts that are already popular, and then as a function of their increased visibility, these popular posts will continue to receive more upvotes. This is analogous to the model of preferential attachment studied in Chapter 10: Does the Internet Have an Achilles' heel of *Networked Life*. In preferential attachment, new nodes added to a network tend to connect to nodes which already have high in-degree, similar to how new upvotes tend to accumulate on posts that already have a lot of upvotes. Thus, when analyzing the RPI for subreddits in 4.3, because RPI for a subreddit depends on the average upvotes from various other subreddits, we believe the median is a more explanatory measure of typical RPI.

4 Discussion and Conclusions

4.1 Quadrant Analysis: How much engagement do top posts get?

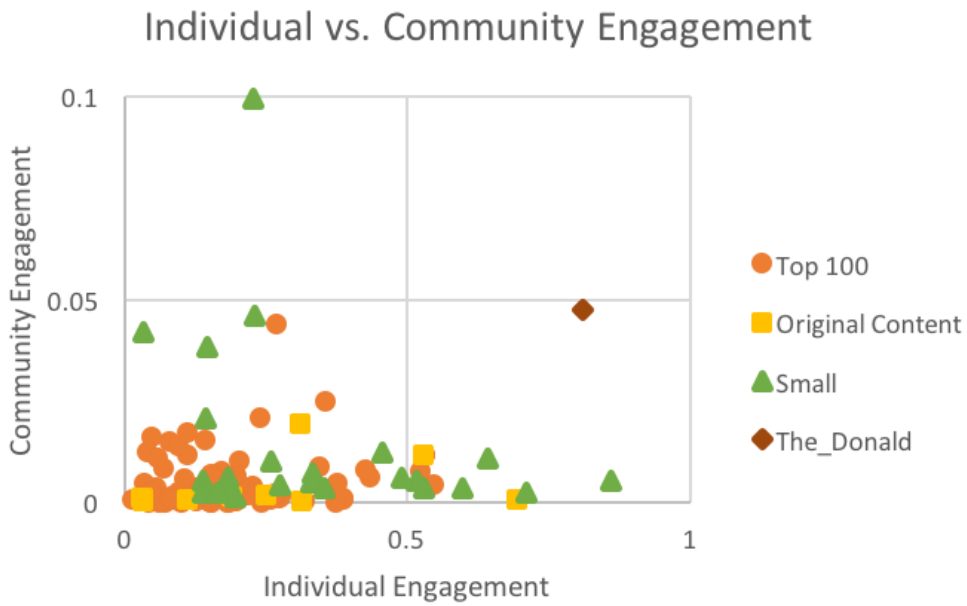


Figure 5: Plot of individual and community engagement for various subreddits.

The category of Top 100 subreddits is clustered in the lower-left quadrant, meaning the top posts are authored by users with low individual engagement and those posts receive low community engagement. Given that these subreddits are general-interest, this pattern is not unsurprising. For instance, while many people might be interested in the funny pictures from /r/Funny, very few people will only be interested in funny pictures.

For Original Content subreddits, it is difficult to discern a significant difference from the other categories in terms of individual engagement, but it is clear that community engagement is relatively low on the whole. Perhaps this a factor of subreddits driven by Original Content not needing to appeal to a very wide fraction of the subreddit.

For small subreddits, we do see relatively large individual engagement and community engagement compared to the Top 100 subreddits. We propose the following explanatory mechanism: smaller subreddits are about more niche topics, so while many people might subscribe to /r/Funny because they like funny pictures, the only users that will subscribe to /r/lexington are people who live in Lexington, KY. As a result, this self-selection means that the typical subscriber to /r/lexington is more invested in the subreddit than the typical subscriber to /r/funny. As the plot shows, these self-selecting groups tend to have top posts from users with higher individual engagement (frequently above 0.5), and they can have very high community engagement.

One interesting datapoint separate from the three categories that we added was /r/The_Donald. This subreddit is for supporters of Donald Trump and has created quite a bit of controversy on Reddit, with Reddit coming under fire for possible censoring of the subreddit, and members of the subreddit being accused of being toxic and harmful to the overall community. The controversial status of /r/The_Donald carries over to the plot, as the subreddit is a datapoint with no peers. It has extremely high individual engagement of 0.81, which means that the users who have the top posts in /r/The_Donald contribute 81% of their total posts to that community. The community engagement is the second highest of the subreddits we investigated, coming second to a much smaller community dedicated to Star Wars prequel memes (335,783 subscribers /r/The_Donald, 5,910 to /r/PrequelMemes).

For users new to Reddit looking to create posts that rise to the top, we suggest looking at subreddits in the lower-left quadrant, with low individual and community engagement. We suggest to look for low individual engagement subreddits because these subreddits likely surface top posts that don't require extensive history of context and knowledge about the particular inside jokes and idiosyncrasies of that subreddit. We also suggest subreddits with low community engagement because these are communities posts can become popular while appealing to a smaller proportion of the community. An example of a subreddit that matches these criteria is /r/PersonalFinance, with scores of 0.06 and 2.6e-4 for individual and community engagement respectively.

4.2 Cross-pollination between subreddits: Measuring weighted link importance

First we had to decide which subreddits to measure against each other. We first observed top subreddits, such as /r/funny and /r/pics, but what we found was that in all tested cases, the connection value was either negligibly low or 0. This is probably because the posters to those subreddits are so varied and their interests so varied that the size of samples we were taking did did not discover cross-pollination.

The second tests we made were across political subreddits, including /r/The_Donald and /r/hillaryclinton, trying to match them with subreddits with similar views, such as /r/dncleaks and /r/enoughtrumpspam. However, because the top posters in political subreddits seem to keep very heavily inside those subreddits, they received 0 scores even with subreddits with similar views.

We then decided to test a network known to be based on geography, local sports, to see if we could gain information about the sports tendency of areas and locations. We examined the "big four" of Philadelphia–basketball, football, baseball, and hockey teams–and their relatedness to themselves, as well as some of the relevant league subreddits (e.g. /r/NBA).

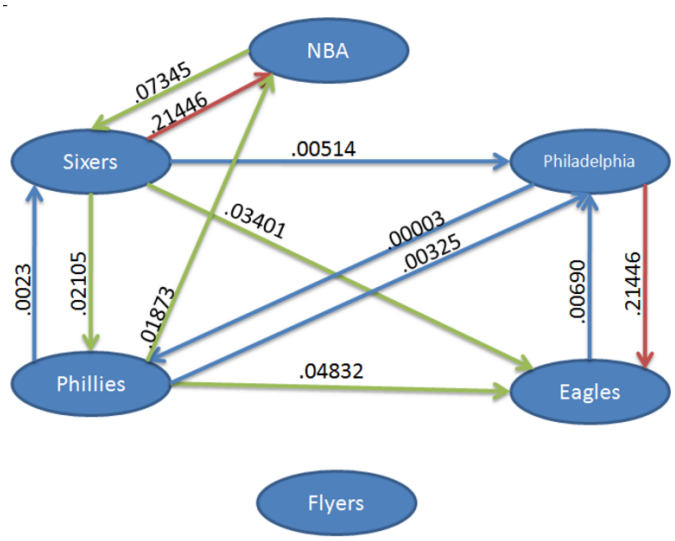


Figure 6: Cross-pollination graph for Philadelphia sports teams.

Here in the graphs, red lines indicate a participation from one subreddit in another of .1 or greater, green lines indicate a participation from one subreddit in another of .01 or greater, and blue indicate any other non-zero participation. What we found was interesting. Comparing to known information, we can confirm that /r/timberwolves and /r/sixers users are a subset of /r/NBA. We can also confirm that Philadelphia is a football town, as is known, and also make the interesting observations that /r/NBA is the biggest sports subreddit, even though local people are most likely to follow the football team in addition to any other sports of the area.

We also applied to same analysis to Minnesota sports teams.

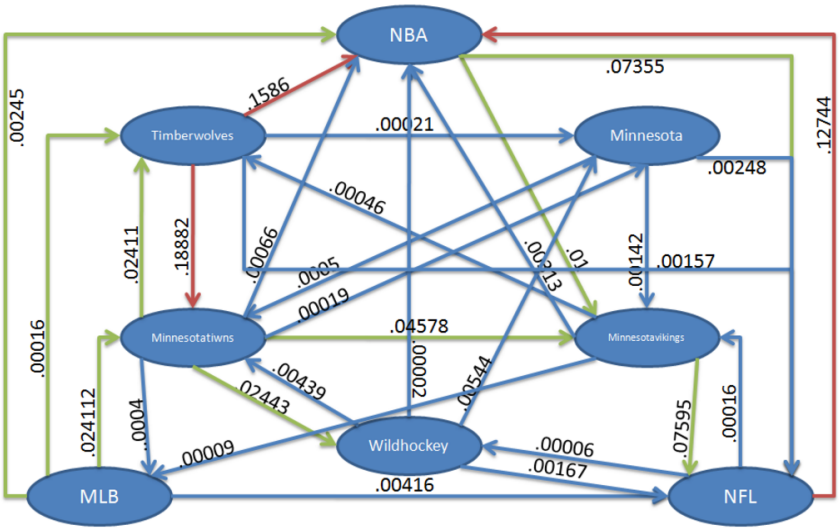


Figure 7: Cross-pollination graph for Minnesota sports teams.

The above graph suggests that if you want to become a star in /r/Timberwolves, then you should also participate in /r/NBA and /r/MinnesotaTwins. There is likely context in terms of discussion, memes, and knowledge among these communities that is shared among the top users.

Overall we can conclude that subreddits with users that stay only in the subreddit have 0 cross pollination, and that it is clear that larger communities have more participation in and less participation out.

Future improvements that could be made include scraping all posts of a given subreddit, though it would take much more time, and also nuancing the conclusions reached here by interlacing these results with the previous analysis of engagement of the top posts.

4.3 The Reddit Power Index: How "average" are top users?

RPI is a metric we created that can help us pinpoint the difficulty of creating a top post in any given subreddit. The RPI also allows us to clearly see what subreddits are dominated by users that consistently have successful posts. All this really means is that with the RPI we are able to find subreddits that will have more or less "alpha redditors" that one will have to compete with for upvotes.

Category	**Average RPI**	**Median RPI**
Top 100	149.8	13.8
25 Small ($< 10,000$)	3.0	2.0
10 Original Content	15.2	2.6

Figure 8: RPI scores for the three different categories of subreddits examined.

On the whole, RPI did provide a reasonable number we could use in analysis. We implemented a 10% trimmed mean in calculating the $\text{RPI}_{\text{subreddit}}$ to protect the metric against large outliers. However, there are still cases such as /r/gaming in the Top 100 which had an outlier RPI of 4767 that skewed the average RPI of Top 100 subreddits upwards. This is why we believe that median RPIs are the better way to quanitfy the typical subreddit RPI in our categories.

Shifting our focus to the Figure 8, we can see that the median RPI are more reliable. Subreddits that fall into the Top 100 category have significantly higher RPI than small or original content subreddits. This tells us that the popular posts in the Top 100 subreddits are typically created by users that typically get more than 10x the average number of upvotes! Conversely, our data show that the RPI for top users in subreddits less than 10,000 subscribers has a median value of 2, which is much closer to being an average Reddit user. The Original Content subreddits also have low RPI, though we suspect that this might be largely a function of size of subscribers, as the Original Content subreddits we examined happened to have smaller subscriber bases.

New users of Reddit should target subreddits with RPI scores closer to 1. These subreddits commonly surface content from more average users, not just from Reddit superstars. Our data tell us that smaller subreddits, such as /r/improv with an RPI of 1.4 and 7,332 subscribers, typically have lower RPI scores. Even some larger subreddits, such as /r/Frugal with an RPI of 1.7 and 613,046 subscribers and /r/GetMotivated with an RPI of 1.6 and 9,779,376 subscribers, have low RPI scores as well.

Posting to a subreddit with low RPI does not guarantee success. What it does mean is that you are competing against more average users of Reddit to get the top posts, hopefully giving yourself a chance to standout.

5 Final Words

"Man naturally desires, not only to be loved, but to be lovely." Adam Smith wrote that line in 1759 in *The Theory of Moral Sentiments*, and the same could be said of our behavior on Reddit today. We, the users of the this social bulletin board called Reddit, crave recognition and popularity. We want to be "loved" and "lovely"–we want our posts to become popular.

To that end, our analysis produced a few steps of action. There are clear differences between large and small subreddits in terms of individual and community engagement, as well as the RPI of top users. These differences are important to keep in mind for new users coming to the site, as it might be beneficial to begin in subreddits tailored to your interests while also keeping an eye out for low RPI, low engagement subreddits. Additionally, understanding the importance of what we termed cross-pollination will help you tailor which sets of communities to participate in. By deliberately choosing a set of subreddits, you can benefit from the same shared context that top users in those subreddits already have. Hopefully, with these takeaways, the factors of what makes a good Reddit post have been made clearer, and we can all use them to be more "loved" and more "lovely" on the site.

Appendix

The pages that follow contain the code used to collect the data from Reddit used in our analysis, roughly in the order in which the results appear in the report. The data used to generate the plots and charts can be accessed here.

```
# 1. Quadrant Analysis - Community and Individual Engagement

import praw
import datetime, csv, time
import numpy as np
from scipy import stats

# turns each line of .txt file into new string in a list
# code from http://stackoverflow.com/questions/3277503/how-to-read
#     by-line-into-a-list-with-python
def textFileToList(filename):
    return [line.rstrip('\n') for line in open(filename)]

# which subreddit to look at?
subredditList = textFileToList('smallSubreddits2.txt')

# build-up filename
today = datetime.datetime.now()
todaysDate = "%s-%s-%s" % (today.year, today.month, today.day)
csvFilename = todaysDate + "-theDonald-results.csv"

# create CSV and headers
with open(csvFilename, 'wb') as f:
    writer = csv.writer(f)
    writer.writerow(['subreddit', 'propThisSubreddit', 'upvotes/su

# authenticating reddit instance
reddit = praw.Reddit(client_id='DuP1WxEg9W_T5Q',
                     client_secret="dzhbIEv0jwqO2T9lR4RCkupO_0g",
                     password='chengonzalez',
                     user_agent='USERAGENT', username='ele381')


# how many authors to get / how many of the top posts to look at
numberOfAuthors = 99
numberOfPosts = 99

# for each author, how far to go back in their history
# can take values {all, year, month, week, day, hour}
submissionTimePeriod = "month"
```

```
proportionList = []
communityEngagementList = []

# find community and individual engagement for subreddits in list
for j, thisSubreddit in enumerate(subredditList):
    # get the author of the top posts in this subreddit
    authors = []
    ups = []
    while True:
        try:
            for submission in reddit.subreddit(thisSubreddit).top(
            submissionTimePeriod, limit=numberOfAuthors):
                authors.append(str(submission.author))
                ups.append(submission.ups)
            break
        except Exception as e:
            print(type(e))
            print 'pausing'
            time.sleep(5)

    # get subreddit of recent posts submitted for each author
    freqThisSubreddit = []
    freqOtherSubreddits = []

    # find individual engagement for authors in this subreddit
    for i, redditUser in enumerate(authors):
        print thisSubreddit + ':', redditUser, (i+1), 'of', len(au
        countThisSubreddit = 0
        countOtherSubreddits = 0

        # catch exception if the user is deleted
        try:
            for submission in reddit.redditor(
            redditUser).submissions.new(limit=numberOfPosts):
                #print submission.title
                if str(submission.subreddit) == thisSubreddit:
                    countThisSubreddit += 1
                else:
                    countOtherSubreddits += 1
```

```
            freqThisSubreddit.append(countThisSubreddit)
            freqOtherSubreddits.append(countOtherSubreddits)
    except Exception as e:
        print(type(e))

'''
data to record
'''
tempProportion = float(sum(freqThisSubreddit)) / (sum(freqThis
sum(freqOtherSubreddits))
tempCommunityEngagement = (float(np.mean(ups))/
reddit.subreddit(thisSubreddit).subscribers)

# debugging printing
print (j, 'of', len(subredditList)-1, thisSubreddit, ': Ind En
tempProportion, 'Com Eng = ',tempCommunityEngagement)

# write to csv
with open(csvFilename, 'a') as f:
    writer = csv.writer(f)
    writer.writerow([thisSubreddit, tempProportion, tempCommun
```

```
# Power-law distribution analysis

import praw
import time
import numpy as np
import matplotlib.pyplot as plt

# get upvotes for a random selection of posts
def getUpvotes(nameOfSubreddit, upvotes):
    timestampStart = 1464739200 # 6/1/2016
    timestampEnd   = 1484291223 # today
    randomSampleSubmissions = reddit.subreddit(nameOfSubreddit).su
    timestampStart, timestampEnd)

    # random sample of reddit posts during time period
    for i, submission in enumerate(randomSampleSubmissions):
        upvotes.append(submission.ups)
        if i+1 == numberOfSubmissions:
            break

# authenticating Reddit instance using OAuth
reddit = praw.Reddit(client_id='DuP1WxEg9W_T5Q',
                     client_secret="dzhbIEv0jwqO2T9lR4RCkupO_0g",
                     password='chengonzalez',
                     user_agent='USERAGENT', username='ele381')

# parameters for scraping
upvotes = []
subredditList = ['AskReddit']
numberOfSubmissions = 1000

# get the upvotes for posts
for i, subredditName in enumerate(subredditList):
    print i, 'of', len(subredditList)
    while True:
        try:
            getUpvotes(subredditName, upvotes)
            break
        except Exception as e:
            print(type(e))
```

```
            print 'pausing'
            time.sleep(5)

# sort the list in descending order
sortedUpvotes = np.sort(upvotes)
sortedUpvotes[:] = sortedUpvotes[::-1]
print sortedUpvotes[:]

# Plot on log-log axes
plt.figure()
ax = plt.gca()
ax.loglog(range(1, len(sortedUpvotes)+1), sortedUpvotes)
plt.grid(True)
plt.title('Random Posts in /r/AskReddit')
plt.xlabel('Kth most upvotes')
plt.ylabel('Upvotes')
plt.show()
```

```
# 2. Cross-pollination analysis

import praw, sys

# which subreddits to look at?
thisSubreddit = str(sys.argv[1])
thisSubreddit2 = str(sys.argv[2])

# how many authors to get / how many of the top posts to look at
numberOfAuthors = int(sys.argv[3])

# for each author, how far to go back in their history
# can take values {all, year, month, week, day, hour}
submissionTimePeriod = "all"

'''
Setting things up for parsing data
'''
# authenticating reddit instance
reddit = praw.Reddit(client_id='DuP1WxEg9W_T5Q',
                     client_secret="dzhbIEv0jwqO2T9lR4RCkupO_0g",
                     password='chengonzalez',
                     user_agent='USERAGENT', username='ele381')
# store data
authors = []
freqThisSubreddit = []
freqThisSubreddit2 = []
total = []

# get the author of the hot 100 posts in this subreddit
for submission in reddit.subreddit(thisSubreddit).hot(limit=number
    authors.append(str(submission.author))

# get subreddit of recent posts submitted for each author
for i in range(len(authors)):
    print ('Currently looking at ' + authors[i])
    #print (thisSubreddit.name)
    #print (thisSubreddit2.name)
    countThisSubreddit = 0
    countThisSubreddit2 = 0
```

```
    counttotal = 0

    for submission in reddit.redditor(authors[i]).submissions.top(
    submissionTimePeriod):
        #print ('Currently looking at ' + submission.subreddit.nam

        if str(submission.subreddit) == thisSubreddit:
            countThisSubreddit += 1
        elif str(submission.subreddit) == thisSubreddit2:
            countThisSubreddit2 += 1
        counttotal += 1
    freqThisSubreddit.append(countThisSubreddit)
    freqThisSubreddit2.append(countThisSubreddit2)
    total.append(counttotal)

score = 0

# calculate relationship score
for i in range(len(authors)):
    score += ((freqThisSubreddit[i]/total[i]) * (freqThisSubreddit
    * (4/len(authors)))
print (score)
```

```
# 3. Reddit Power Index

import praw
import time, datetime, csv, os.path, math
import numpy as np
from scipy import stats

# turns each line of .txt file into new string in a list
# code from http://stackoverflow.com/questions/3277503/how-to-read
#     by-line-into-a-list-with-python
def textFileToList(filename):
    return [line.rstrip('\n') for line in open(filename)]

# this function is only called when average upvotes isn't cached y
def getSubredditNormalUpvotes(nameOfSubreddit):
    print nameOfSubreddit, 'not cached'
    global cachedNormalUpvotes
    submissionUpvotes = []

    # choose old time instead of new because new posts are still a
    # accumulating more votes
    timestampStart = 1475280000 # 10/1/2016
    timestampEnd   = 1484291223 # 1/13/2017

    # get random set of submissions within defined timeframe
    randomSampleSubmissions = reddit.subreddit(nameOfSubreddit).su
    timestampStart, timestampEnd)

    # random sample of reddit submissions during time period
    start_time = time.time()
    for i, submission in enumerate(randomSampleSubmissions):
        submissionUpvotes.append(submission.ups)
        if i+1 == averageUpvoteSubmissions:
            break
    print("-- %s seconds --" % (time.time() - start_time))

    # calculate average number of upvotes for a post
    avgUpvotes = np.mean(submissionUpvotes)

    # redo sampling for non-active subreddits
```

```
    # non-active subreddits might have 0 posts in the time frame,
    # new instead
    if math.isnan(avgUpvotes):
        submissionUpvotes = []
        randomSampleSubmissions = reddit.subreddit(nameOfSubreddit
        # random sample of reddit posts during time period
        for i, submission in enumerate(randomSampleSubmissions):
            submissionUpvotes.append(submission.ups)
            if i+1 == averageUpvoteSubmissions:
                break
        avgUpvotes = np.mean(submissionUpvotes)

    # debugging printing
    print nameOfSubreddit, 'average upvotes:', avgUpvotes

    # cache value
    cachedNormalUpvotes.append([nameOfSubreddit, avgUpvotes])

    # write out to CSV cached values
    with open('cachedUpvoteData.csv', 'a') as f:
        writer = csv.writer(f)
        writer.writerow([nameOfSubreddit, avgUpvotes])

    return avgUpvotes

# function returns the RPI for this submission
#   submission is a PRAW submission object
def getSubmissionRPI(submission):
    # look if the subreddit that submission is in already has avg
    normalUpvotes = -1
    for cachedData in cachedNormalUpvotes:
        if submission.subreddit in cachedData:
            normalUpvotes = cachedData[1]

    # not cached, need to find and cache
    if normalUpvotes == -1:
        while True:
            try:
                normalUpvotes = getSubredditNormalUpvotes(str(subm
                break
```

```
            except Exception as e:
                print(type(e))
                print 'pausing'
                time.sleep(5)

    # debugging printing
    submissionRPI = float(submission.ups) / normalUpvotes
    print 'Submission RPI (', submission.title, '):', submissionRP

    return submissionRPI

# function returns the RPI for this specific user
#   redditUser is the PRAW redditor object
#   numberOfPosts is how many submissions to look at
def getUserRPI(userName, numberOfSubmissions):
    listUserRPI = []

    # get RPI for this user's posts
    # reverse the new list so that reverse order of the 50 newest
    # way the posts looked at have accumulated most of all upvotes
    for i, submission in enumerate(reversed(list(reddit.redditor(
    userName).submissions.new(limit=50)))):
        if i+1 == numberOfSubmissions:
            break
        listUserRPI.append(getSubmissionRPI(submission))

    # debugging printing
    userRPI = np.mean(listUserRPI)
    print 'User RPI:', userRPI
    return userRPI

# get the authors of the top posts in a specific subreddit
# top is determined by submissionTimePeriod (e.g. week, month)
def getAuthorsOfTopPostsInSubreddit(subredditName, numberOfAuthors
submissionTimePeriod):
    while True:
        try:
            # find the authors of top posts in subreddit
            listTopAuthors = []
            for submission in reddit.subreddit(subredditName).top(
```

```
            submissionTimePeriod, limit=numberOfAuthors):
                listTopAuthors.append(str(submission.author))
            break
        except Exception as e:
            print(type(e))
            print 'pausing'
            time.sleep(5)
    return listTopAuthors

# get the RPI of the users who create top posts in a subreddit
def getSubredditTopSubmissionsRPI(subredditName, numberOfAuthors,
numberOfSubmissions, submissionTimePeriod):
    listTopAuthors = getAuthorsOfTopPostsInSubreddit(subredditName
    numberOfAuthors, submissionTimePeriod)

    # find RPI for each author
    listTopAuthorsRPI = []
    for author in listTopAuthors:
        count = 0
        while True:
            try:
                count = count + 1
                listTopAuthorsRPI.append(getUserRPI(author, number
                break
            except Exception as e:
                if count == 5:
                    print 'deleted user'
                    break
                print(type(e))
                print 'pausing'
                time.sleep(5)


    # Trim 10% at both ends
    topAuthorsRPI = stats.trim_mean(listTopAuthorsRPI, 0.1)

    # debugging printing
    print 'Subreddit RPI for ', subredditName, topAuthorsRPI

    return topAuthorsRPI
```

```
# authenticating PRAW Reddit instance
reddit = praw.Reddit(client_id='DuP1WxEg9W_T5Q',
                     client_secret="dzhbIEv0jwqO2T9lR4RCkupO_0g",
                     password='chengonzalez',
                     user_agent='USERAGENT', username='ele381')

# must define this variable globally, used by functions to cache a
cachedNormalUpvotes = []

# read in persistent cached values for average upvotes from previo
if os.path.isfile('cachedUpvoteData.csv'):
    with open('cachedUpvoteData.csv') as f:
        data=[tuple(line) for line in csv.reader(f)]
    for line in data:
        cachedNormalUpvotes.append((line[0], float(line[1])))

# print out the cached values for average upvotes
print cachedNormalUpvotes

# parameters for scraping
subredditList = textFileToList('smallSubreddits.txt')
# number of top users to look at
numberOfAuthors = 10
# number of submissions of each user to look at for subreddit RPI
numberOfSubmissions = 25
# number of submissions to look at for calculating average upvotes
averageUpvoteSubmissions = 200
submissionTimePeriod = "month"

# create CSV used to write out subreddit RPI scores
today = datetime.datetime.now()
todaysDate = "%s-%s-%s" % (today.year, today.month, today.day)
csvFilename = todaysDate + 'smallSubresults.csv'
with open(csvFilename, 'wb') as f:
    writer = csv.writer(f)
    writer.writerow(['subreddit', 'Top Post RPI'])

# get RPI of subreddits
for i, subredditName in enumerate(subredditList):
```

```
    print i, 'of', len(subredditList)
    thisSubredditRPI = getSubredditTopSubmissionsRPI(subredditName
    numberOfAuthors, numberOfSubmissions, submissionTimePeriod)
    print i, 'of', len(subredditList), ':', subredditName, thisSub

    with open(csvFilename, 'a') as f:
        writer = csv.writer(f)
        writer.writerow([subredditName, thisSubredditRPI])

# get RPI of user
# username = 'ericjwdchen'
# a = getUserRPI(username, 100)
# print username, 'RPI =', a
```