Using Machine Learning to Predict the Popularity of Reddit Comments

Sean Deaton Scott Hutchison Suzanne J. Matthews*

sean.m.deaton.mil@mail.mil
[scott.hutchison, suzanne.matthews]@usma.edu
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, NY 10996

ABSTRACT

Predicting the popularity of social media content enables companies and individuals to affect user behavior on-line. These effects may be manipulative and malicious in their nature, to include the spreading of false information. Reddit, an open-source social media platform, is an excellent vector for transmitting false information, and has come under fire in the past for witch-hunts, rumor-mongering, and doxxing. In this paper, we examine the efficacy of machine learning at predicting Reddit comment popularity. For our experiment, we used features commonly associated with Reddit popularity derived from literature. We tested our approach on a dataset of over two million Reddit comments. Supervised machine learning classifiers were fit with a limited feature set and accuracy consistently ranged from 42.0% to 52.7% with a Cohen's kappa score ranging from -0.160 to 0.056. Given the low kappa statistic, these results do not indicate the success of the combination of features and classifiers we chose to classify the popularity of comments.

1. INTRODUCTION

The popularity of on-line media profoundly impacts our activity on-line. Popularity implies that content is vetted by other users and worthy of our time. Many companies devote a vast amount of resources to predict the future popularity of on-line material. If they had an automated method for instantly predicting the popularity of a Reddit comment, this could be extremely valuable for marketing campaigns or targeted advertising. More concerning however, would be its use to a malicious actor. If an individual knew what features led to popularity in a subreddit, he or she could manipulate comments in the community to spread propaganda. As many individuals use Reddit as a news source [1], they could be presented with false information that is seemingly vetted by the community.

Reddit is a popular social media website where users share news, pictures, video, and other types of media. While the Pew Research Center estimates that only 4% of Americans use Reddit [1], the site's self-aggregated statistics indicate that there were 234 million unique visitors in December 2015, and approximately 8 billion unique page views [10]. Reddit attracts an audience that is roughly equally male and female (53% vs. 47%) and an equal parity of U.S. versus International users (54% vs. 46%) [10].

In addition to sharing media, users who create accounts on the website can subscribe to communities known as "subreddits", which enables them to keep track and interact with content of personal interest. Each subreddit has its own community page. Users use voting to move particular topics to the top of a subreddit. Posts with a high number of positive votes ("upvotes") rise to the top of of a community's front page, where they have a high chance of being viewed by other visitors to the subreddit.

Increasingly, visitors use Reddit as a news source. In a recent survey, approximately 70% of surveyed Reddit users indicated they use the site as their primary news source [1]. For the 2016 Presidential campaign for instance, 45% of Reddit users used the site to track the election. This is on-par with other social media platforms such as Facebook and Twitter, which were estimated at 52% and 43% respectively [1]. Politicians and celebrities have taken notice; several have done Q&As with Reddit users on the website, notably Presidents Barack Obama and Donald Trump, and former presidential candidate Bernie Sanders [2].

However, the site does have a history of inspiring vigilantism, with victims frequently having their personal information posted on the Internet (a practice known as doxxing). One notable example included Sunil Tripathi, a missing student who was made a leading suspect in the Boston Marathon bombings before the true culprits were identified. Reddit general manager later issued an apology in which he condemned the "witch-hunts and dangerous speculation." [9]

Reddit has made efforts to curb the spread of misinformation. In November 2016, Reddit banned the PizzaGate subreddit from the site, stating that they "don't want witch-hunts on our site" [8]. This resulted in a backlash from users on `r/theDonald`, who stated the move amounted to censorship. Reddit changed the way its front-page algorithm works after users of the `/TheDonald` were allegedly exploiting the algorithm to spam the front page with pro-Trump

*corresponding author

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

messaging. One moderator has been subjected to threats. The CEO has been hesitant to ban the subreddit due to the controversy it could cause; however, posts stickied from the subreddit have been banned from the front page [6]

In this paper, we seek to determine if the popularity of *comments* on particular Reddit posts can be predicted using popular machine learning techniques. In the context of Reddit, comments that are rated as popular tend to show near the top of a post's comment page. Theoretically, a reader can be influenced by the top comments on a post's page.

We hypothesized that the same features commonly marking a post as being popular can be applied to comments as well. To test our hypothesis, we surveyed the literature for features commonly associated with post popularity and machine learning techniques that have been used to predict Reddit popularity in the past. We tested our hypothesis through obtaining, fitting, and testing two million Reddit comments through three different supervised machine learning classification algorithms. Our experimental methodology was also derived from the literature.

The results were less favorable, showing accuracy worse-than-random, with extremely low kappa statistics. With our results, there is no evidence to suggest that the set of identified features allow a user to predict the popularity of a comment. Tailoring a comment to achieve popularity in any given subreddit may be more nuanced than focusing on any sole feature, making it difficult for a malicious actor to unfairly alter their comment.

The rest of the paper is organized as follows. Section 2 gives relevant background information, while Section 3 gives an overview of our experimental methodology. We discuss our results in Section 4. Lastly, we conclude in Section 5.

2. BACKGROUND

Reddit is entirely open source. As such, the algorithm for determining a post's popularity is open for inspection. In a post on Hacking and Gonzo, the author explores this algorithm and notes that submission time and the score of the post, denoted by the amount of upvotes minus downvotes. The score is logarithmic [12], meaning that the first ten upvotes are equivalent to the next 100 and so on.

2.1 Popularity

One way Reddit classifies popularity is by describing a post as *hot*. When classified as hot, a post becomes prominently featured on the subreddit. Hotness is determined by the score of the post on a logarithmic scale such that the first ten upvotes produce the same weight as the next one hundred. This number is then added to the number of seconds since epoch divided by 45,000. This shows that time has a large impact on how prominently a post is features [12].

This is not applicable to comments, as the algorithm would favor comments posted closer to the post's original submission time [12]. Rather, the best overall comment should appear first in the responses, regardless of time of submission. This is done by utilizing a confidence sort [12]. The confidence sort balances the number of upvotes and downvotes, the algorithm for which is shown in the source code figure 2. A comment with 10 upvotes and 1 downvote would have a higher confidence score than a post with 40 upvotes and 20 downvotes, and thus be ranked higher and appear closer to the original post.

```
f _confidence(ups, downs):
  n = ups + downs
  if n == 0:
      return 0
  z = 1.281551565545
  p = float(ups) / n
  left = p + 1/(2*n)*z*z
  right = z*sqrt(p*(1-p)/n + z*z/(4*n*n))
  under = 1+1/n*z*z
  return (left - right) / under

f confidence(ups, downs):
  if ups + downs == 0:
      return 0
  else:
      return _confidence(ups, downs)
```

Source Code 1: Source code for Reddit's confidence sort, translated [12].

The algorithm looks only at number of votes and the disparity between upvotes and downvotes. While the algorithm shows how comments are ranked as more popular that others, the relationship between upvotes and comments is unclear. Thus, our primary experimental goal was to try and identify the set of features that make users upvote a particular comment.

2.2 Related Work

Several researchers have attempted to study what makes Reddit posts popular, with varying levels of success. For example, two independent studies [5, 3] have pointed to the time of day as being an important indicator for a post making it to the front page, particularly 0900 PST. Lakkaraju *et. al.* studied [7] the importance of submission titles in predicting the popularity of posts containing images. The authors concluded that sentiment is a strong predictor and is specifically niche to particular communities. Furthermore, polarizing posts, determined by some sentiment analysis, fair better than neutral ones [7]. Certain subreddits inherently get more up votes than others, to include `r/funny`, which also ties to the fact that images are also more highly upvoted. The authors also found that the title of the post should be similar to the wording used within the community, but at the same time novel to introduce dissimilarity.

A student paper at Stanford [15] attempted to classify posts to subreddit based on post title alone. In preprocessing, they removed each common word and counted number of instances. Every tenth post was used to the actual model while the other nine were used for training. In classifying, they applied three main algorithms to classify post titles: linear classification, Multinomial Naïve Bayes, and support vector machines (SVM). They concluded that a SVM classifier performed best on the test set, with an accuracy of 96.46% on 2 subreddits and 84.91% on 10 subreddits.

In a blog post published in 2016, DataStories analyzed the trends found in the top 100 posts that occur on the Reddit front page [3]. For their analysis, they scraped the top 100 posts for every 2 minutes for 22 days. They deleted any posts that were on the front page for less than 2 min-

utes. This yielded a total of 2,344 unique posts. Their findings supported the work of the earlier researchers, including the importance of posting time, the relative success of polarizing posts compared to neutral ones, and the relative success of positive headlines to stay on the front page for a long time [3]. While image posts do tend to get more upvotes than textual votes, the latter tends to stay on the front page longer, and receives more comments than image submissions. They also found that certain subreddits dominate the front page, such as `r/funny`, `r/pics`, `r/gifs`. Top posts that put numbers in their headlines also increase the likelihood that posts are make the front page. Lastly, the authors note that the average lifetime for a post on the front page is 4 hours and 15 minutes; over 85% of the posts get to the front page in less than 3 hours [3].

In 2016, Tracy Rohlin studied Reddit post popularity as the subject of his masters thesis [11]. He gathered data from six subreddits. Only the first 1,000 posts in a subreddit were considered. Each post was first preprocessed and represented as a feature vector using the bag of words (BOW) model, term frequency – inverse document frequency (TFIDF), or Linear Dirichlet Allocation (LDA). Each post in a subreddit was labeled "popular" or "unpopular" by comparing its voting score to a threshold. The resulting feature vector was fed into either a Naïve Bayes classifier (NBC) or a support vector machine (SVM). The accuracy of each classification was above 50% for each subreddit. The author notes that lowering the text length cutoff would have increased the dataset size and possibly increased model accuracy.

Most research in this field has been performed on Reddit posts. Posts are the primary mechanisms for content delivery on Reddit. Many users are dubbed *lurkers*, or users who only consume content and rarely ever make comments. These users may also only look at the original post and never make it to the underlying comments. However, comments are a great method for delivering in-depth and directed responses to other users. This makes it an ideal mechanism to perpetuate false information or perform targeted attacks. If a malicious actor knew what would be popular within the subreddit, they could more easily gain support from the community to achieve their objectives. The main contribution of our work is providing an initial analysis of how feasible it would be for a malicious actor to manipulate features of a comment to make them more popular.

3. METHODOLOGY

We study the performance of the Naïve Bayes and Support Vector Machine classifiers, as Rohlin used these in his thesis, DataStories [3], and also utilized by students at Stanford. In addition, we examined a Decision Tree classifier, as we hypothesized that this classifier would also work well.

For experimentation, we utilized the Department of Defense's High Performance Computing cluster, Topaz, provided by ERDC, the U.S. Army Engineering Research and Development Center. The cluster has 3,456 nodes with 36 cores of Intel's Xeon E5-2699v3, clocked at 2.3GHz. Each node has 117 GB of RAM available to it. Since our approach was serial in its design, we utilized only one node with one core. However, the project's analysis speed certainly benefited from the increased memory available on Topaz. The cluster was running 64-bit SuSE Linux and Python version 2.7.10.

Table 1: Selected Features

Moderator Status	Hour of creation [3]
Sentiment of comment [3]	Controversiality
Bag of words [11]	-

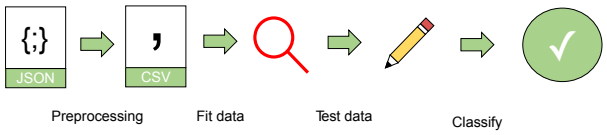

Figure 1: Preprocessing

3.1 Data and Feature Set

The data was derived from a post on `r/datasets` [14]. Uncompressed, the size of the comments was 30 GB and included 53,851,542 comments. The dataset represents only comments posted in reply to a post and did not include the original post. Each line of the file was in JSON and included all of the following fields: score_hidden, name, link_id, body, downs, created_utc, score, author, distinguished, id, archived, parent_id, subreddit, author_flair_css_class, author_final_text, gilded, retrieved_on, ups, controversiality, subreddit_id, and edited.

Table 3.1 shows the features selected for analysis. The bag of words was selected as it was the sole feature analyzed in Rohlin's thesis. Rohlin obtained better-than-random results using only the bag of words, suggesting it might also a good feature for comment popularity [11]. In the DataStories blog post, it was found that 0900 PST was the best time to obtain upvotes [3]. While this examined only posts, the same might also hold true for comments as well. While there was nothing in literature to suggest that moderator status or controversiality would affect popularity, we predicted they would be good features to examine. Moderators are usually vetted members on Reddit and frequently post in the subreddits they moderate. Given this, we hypothesized that their status as an authority figure within a subreddit may improve their chances of getting upvotes on a particular comment. The DataStories blog post also found that comments with polarizing sentiments, either overly positive or negative, yielded the most upvotes [3]. While it does not directly translate, this polarity in sentiment could imply that those comments are extremely controversial within a subreddit.

3.2 Preprocessing

For prepossessing, we selected two million comments randomly from the original dataset. The number two million was an arbitrary choice; it enabled us to collect enough data to fit the classifier without being overly time-consuming to process. First, the body of each comment is stripped of any function words and made lowercase. Function words were defined as anything in the Python nltk function word corpus. The sentiment of each post is also determined using the Python TextBlob library and the University of Pittsburgh's subjectivity Lexicon [16]. Each comment without at least two upvotes and at least twenty words in the body are filtered out. This ensures that comments entirely ignored by the community do not skew the classifiers. The created_utc field is stripped to leave just the hour [3].

A dictionary is instantiated such that there is a separate

key for each particular subreddit, and the value is an empty list. We iterated through each comment and appended the number of upvotes to the list corresponding to the comment's subreddit. At the conclusion of this step, the number of upvotes for each comment was contained within a list stored in the dictionary. To label popularity, each list was first sorted and then the index of the top quartile selected. The number at that index represented the threshold for popularity for that subreddit; anything above it was popular and anything below was unpopular. We then iterated over the comments again, checked the number of upvotes and properly labeled the comment as popular or unpopular. This was the true popularity label.

As in Rohlin's thesis, we defined popularity as the top 25% of posts within a subreddit [11]. Since popularity varies wildly between subreddits, this is an important distinction. If popularity was relative to the entire dataset, it is likely that a different subset of comments would be removed by the preprocessing step in our experiment. Each subreddit's comments were written to individual CSV files inside a directory with the final fields including: distinguished, body, sentiment, hour of creation, and controversiality. Finally, we removed a number of unpopular comments from each subreddit's dataset such that the number of unpopular comments equaled the number of popular comments. This was supported by Rohlin's thesis [11]. This ensured that we had the same number of unpopular and popular comments and also established a baseline for classification. To achieve better-than-random accuracy, each classifier needed a score better than 50%.

3.3 Classification

After post-processing 239,099 comments were left spanning 8,259 subreddits. The top five subreddits with the most comments after post-processing were examined for classification. Each CSV file is read in separately. Each field containing text is label encoded, producing an integer that the classifier can read. For example, if a comment is popular, the popular field is set to 1, where false would be 0. The body of the text is turned into a vector where each index represents a distinct word. The number at that index represents the frequency that word appears in the comment.

As in Rohlin's thesis, 20% of the data was saved for testing. The remaining 80% was used for training the classifiers [11]. Each classifier was fitted with both the features and popularity label from the training data. Each classifier then predicted the popularity of the testing data without knowing the true popularity label. Finally, the accuracy, precision, recall, F1, and Cohen's kappa statistic were computed using the predicted values in comparison to the true popularity label for each prediction. The average value across each every prediction within that subreddit was used for the computation of the accuracy, precision, and recall values.

Accuracy is a measurement of how many predictions were accurately labeled. This is the metric by which we sought to rank each classifier. Recall is defined by number of true positives over the true positives and false negatives combined, as shown in equation 1. This shows the the ability of the classier to properly classify comments.

Precision is the total number of true positives over both true positives and false positives combined. It returns a percentage of the classifier's ability to not falsely label comments. The F1 score is a weighted average of both precision and recall.

$$recall = \frac{tp}{(tp + fn)} \tag{1}$$

$$precision = \frac{tp}{(tp + fp)} \tag{2}$$

$$F1 = 2 \cdot \frac{(precision * recall)}{(precision + recall)} \tag{3}$$

Cohen's kappa statistic is a measurement of the classifier's ability to use fitted data to properly label comments. Given that half of our dataset is classified as unpopular, a naïve classifier could easily achieve 50% accuracy simply by guessing that every comment is unpopular. If this is the case, the kappa statistic will determine that predictions are due to chance (closer to zero) and not due to accurate fitting (closer to one). If this were case, the recall score would also be low, as the classifier would be indicating comments as popular when they truly are not. All of these scores range from zero to one, with one being more favorable for our experimentation [13].

4. RESULTS

Accuracy ranged from 42.0% to 52.7%. The Cohen's kappa statistic ranged from -0.160 to 0.056. Anything close to zero indicates that our classifier's popularity prediction agreed with the true popularity label mainly on chance. In the case of the negative statistic, this represents certain features hurt the classifier's ability to make random guesses. These two scores indicate that every classifier yielded worse-than-random results. To reference Table 2 we found no pattern in determining which classifier performed the best. `r/destinythegame`, however, produced the highest kappa statistic with the Decision Tree classifier.

The F1 score in table 2 only shows the balance between recall and precision. The low score indicates a rather significant imbalance between the two. In every case, precision was higher than recall. Given the definition of recall in equation 1, this indicates that predictions included more false negatives than false positives. In the context of our experiments, false negatives correspond to labeling comments as unpopular when they truly are popular. This means that a naïve classifier could consistently label comments as unpopular and still achieve high accuracy.

The accuracy numbers are lower than expected and no classifier achieved better-than-random results. Given the way the data was split, with 50% of the data being classified as popular and 50% classified as unpopular, a better-than-random accuracy score would be greater than 50% with a high kappa statistic. We hypothesized that we would achieve better-than-random results with a much higher kappa statistic than what our experiment yielded. We based this hypothesis on Rohlin's previous work classifying the popularity of Reddit posts using solely a bag of words [11]. While The goal of this paper was not to validate Rohlin's results, we were attempting to apply his work to Reddit comments. However, this was not a perfect recreation. We did not use the same subreddits and we incorporated other features not used in Rohlin's thesis, mainly metadata. For certain subreddits, we included two to three times as many data points. We also examined each classifier's ability to utilize

	r/worldnews (3,031)			r/todayilearned (2,242)			r/news (2,058)		
Classifier	Accuracy	Kappa	F1	Accuracy	Kappa	F1	Accuracy	Kappa	F1
Linear SVM	0.471	-0.057	0.471	0.520	0.037	0.495	0.521	0.042	0.498
Decision Tree	0.525	0.051	0.524	0.520	0.045	0.538	0.457	-0.087	0.462
Naïve Bayes	0.503	0.007	0.506	0.511	0.021	0.495	0.516	0.032	0.514

Table 2: Statistics of "fact"-related subreddits 50%

	r/destinythegame (1,917)			r/leagueoflegends (1,862)		
Classifier	Accuracy	Kappa	F1	Accuracy	Kappa	F1
Linear SVM	0.507	0.009	0.469	0.527	0.045	0.470
Decision Tree	0.524	0.056	0.556	0.469	-0.071	0.408
Naïve Bayes	0.507	0.015	0.507	0.420	-0.160	0.409

Table 3: Statistics for "gamer" subreddits 50%

fitted data, rather than make random guesses, to classify popularity with Cohen's kappa statistic.

5. CONCLUSIONS AND FUTURE WORK

Most data analysis on Reddit has been preformed exclusively on posts. In this experiment, we examined solely comments and produced a novel serial implementation to preprocess and classify the popularity of two million Reddit comments. Through the fitting and testing of these comments on three different supervised machine learning classifiers, we determined each classifier's ability to predict Reddit popularity. With a mixture of features representing both metadata and content, we found no evidence suggesting that popularity is determined by this feature set. Each classifier produced worse-than-random accuracy scores and extremely low kappa statistics. The latter indicates that the classifiers guessed to produce their predictions. We were unable to validate the results produced by other authors conducting similar experiments, which may be attributed to a difference in context between posts and comments. Comments, we now hypothesize, are more sensitive to the original post and other comments in reply to the original post. Future work should attempt to quantify this context for further classification of comment popularity.

The results of this experiment do not mean that it is impossible to tailor comments to garner more upvotes from the community. Rather, it means that the same features that make posts popular are not the same features that make comments popular. Additional work should focus on identifying these features. Reddit users should not let their guard down and trust comments solely because of their popularity. Especially for fact-based subreddits, a myriad of different sources should be analyzed before trusting any particular comment.

For future work, one could vastly improve the rate at which comments are preprocessed. Although the implementation is functional, it is entirely naïve and inefficient. There are several instances where our preprocessor needlessly loops over the data. Furthermore, it would be entirely possible to parallelize the preprocessing step. Implementing both of these improvements would allow for a higher volume of comments to use for testing. This could potentially improve the accuracy of classification. If possible, the data should also be pulled immediately from Reddit. The dataset that we pulled, by the time is was aggregated and submitted for others to work on, was already outdated as it contained comments from 2015. Even if our results suggested machine learning could classify the popularity of Reddit comments, accuracy may not be as high on currently popular comments.

Given the nature of comments, we believe that future work should include three specific features in experimentation: the time relative to the original post, the number of comments previously posted, and the similarity in sentiment between the original post and the comment. With the results obtained in this experiment, we propose that comment popularity is influenced more by context, both relative to the original post and to other comments, and similarity to the original post. While this seems contrary to how comments are sorted on Reddit, as time is not a factor in the confidence scoring, it seems intuitive that if there exists only one comment to a post, that comment would receive more upvotes than a comment in a pool of several other replies.

The subjectivity lexicon used for determining sentiment could be more comprehensive. The University of Pittsburgh's lexicon was chosen because it was freely available under the GNU Public License and originally seemed like a good fit for our experiments. It classified specific words as being positive or negative. However, the lexicon includes only $8,000$ words. While this seems like a large enough sum, for a platform like Reddit, it may not be enough. Furthermore, many users of Reddit intentionally misspell or alter words. For example, in the popular subreddit `r/rarepuppers`, the word 'puppy' is intentionally spelled 'pupper.' The sentiment of 'pupper' would not be classified by the lexicon even though it should likely have the same sentiment as 'puppy.'

To assist others with training future classifiers, all of the work and results of these experiments are available freely on GitHub [4].

6. ACKNOWLEDGMENTS

This project was the work of an Honors Project for the United States Military Academy at West Point and the Department of Electrical Engineering and Computer Science. The opinions expressed in this paper are solely of the authors, and do not necessarily reflect those of the Department of Defense, the U.S. Army, or the United States Military Academy.

7. REFERENCES

[1] M. Barthel, G. Stocking, J. Holcomb, and A. Mitchell. Seven-in-ten reddit users get news on the site. *Pew Research Center*, February 2016.

[2] Brant Tedeschi. The best politics reddit iama's. Website, 2017. https://www.topiama.com/cat/politics/0.

[3] Data Stories. We analyzed 4 million data points to see what makes it to the front page of reddit. here's what we learned. Blog Post, 2 2016. https://blog.datastories.com/blog/reddit-front-page.

[4] S. Deaton. redditpopularity. https://github.com/FourMoreCups/redditpopularity, 2017.

[5] E. Gilbert. Widespread underprovision on reddit. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 803–808. ACM, 2013.

[6] S. Huffman. Tifu by editing some comments and creating an unnecessary controversy. *Reddit Announcements*, November 2016.

[7] H. Lakkaraju, J. J. McAuley, and J. Leskovec. What's in a name? understanding the interplay between titles, content, and communities in social media. *ICWSM*, 1(2):3, 2013.

[8] A. Ohlheiser. Fearing yet another witch hunt, reddit bans 'pizzagate'. *The Washington Post*, November 2016.

[9] Reddit. Reflections on the recent boston crisis. 2013. https://redditblog.com/2013/04/22/reflections-on-the-recent-boston-crisis/.

[10] Reddit Help. Audience and demographics. Internet website, last accessed, 2 2017. https://reddit.zendesk.com/hc/en-us/articles/205183225-Audience-and-Demographics.

[11] T. Rohlin. Popularity prediction of reddit texts. Master's thesis, San Jose State University, SJSU ScholarWorks, 2016.

[12] A. Salihefendic. How reddit ranking algorithms work. Medium.com, 2 2016. https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9#.a8l1ow50b.

[13] scikit-learn-developers. sklearn.metrics. API, 2016. http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.

[14] Stuck_In_the_Matrix. I have every publicly available reddit comment for research. 1.7 billion comments @ 250 gb compressed. Reddit Post, 2016. https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/.

[15] R. Talreja, M. Fang, and J. Baena. Classifying reddit post titles to subreddit. PDF:http://robots.stanford.edu/cs221/2016/restricted/projects/rtalreja/final.pdf, 2016.

[16] Theresa Wilson, Janyce Wiebe and Paul Hoffmann. Subjectivity lexicon. University of Pittsburgh, 2005. http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.

8. APPENDIX

```
from __future__ import absolute_import
import csv
import sys
import time
import json
import re
import os
import datetime
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
import sklearn
from textblob import classifiers

def read_JSON_as_dict(file_name):
    dict_of_subs = {}
    for line in open(file_name, 'r'):
        json_object=(json.loads(line.lower()))
        subreddit = json_object["subreddit"]
        dict_of_subs.setdefault(subreddit, []).append(json_object)
    return dict_of_subs

def remove_function_words(dict_of_subs):
    dict_of_all_words = {}
    for sub, list_of_comments in dict_of_subs.iteritems():
        for comment in list_of_comments:
            comment_list = wordList = re.sub("[^\w]", " ",  comment["body"]).split()
            comment["body"] = comment_list
            for word in comment_list:
                dict_of_all_words[word] = dict_of_all_words.get(word, 0) + 1
            comment["body"] = ([word for word in comment_list if word not in stopwords.words("english")])
    return dict_of_subs

def filter_votes_length(dict_of_subs):
    dict = dict_of_subs
    for sub, list_of_comments in dict.iteritems():
        dict[sub] = [comment for comment in list_of_comments if int(comment["ups"]) \
        >= 2 if len(comment["body"]) >= 20]
    return dict

def determine_if_popular(training_dict):
    dict_of_upvotes = {}
    for sub, list_of_comments in training_dict.iteritems():
        sub_list = []
        for comment in list_of_comments:
            sub_list.append(int(comment["ups"]))
        sorted_list = sorted(sub_list)
        dict_of_upvotes[sub] = int(sorted_list[(len(sorted_list)/2)])
    for sub, list_of_comments in training_dict.iteritems():
        for comment in list_of_comments:
            comment["body"] = " ".join(comment["body"])
            comment["created_utc"] = datetime.datetime.fromtimestamp(int(comment["created_utc"])).strftime("%H")
            comment["sentiment"] = str(sentiment_clf.classify(comment["body"]))
            if comment["distinguished"]!="":
                comment["distinguished"] = "not_distinguished"
            if (int(comment["ups"]) >= dict_of_upvotes[sub]):
                comment["popular"] = True
            else: comment["popular"] = False
    return training_dict

```

```
def get_sentiment(training_dict):
    for sub, list_of_comments in training_dict.iteritems():
        for comment in list_of_comments:
            comment["sentiment"] = 0

def write_csv(file_name, training_list_of_comments):
    headers = ["ups", "popular", "subreddit", "body",  "controversiality", "created_utc", \
    "distinguished", "sentiment"]
    with open(os.path.join("subreddit_twomillion_50_2_csv", file_name), "wb") as file:
        w = csv.DictWriter(file, fieldnames=headers, extrasaction="ignore")
        w.writeheader()
        for comment in training_list_of_comments:
            w.writerow(comment)
        file.close()

lexicon = "lexicon.txt"
lexicon_csv = csv.reader(open(lexicon, "rb"), delimiter=" ")
sentiment_list = [(l[2].replace("word1=", ""), l[5].replace("priorpolarity=", "")) for l in lexicon_csv]
print("training the sentiment classifier")
clf_timer = time.time()
sentiment_clf = classifiers.NaiveBayesClassifier(sentiment_list)
print("It took {0}s to train the sentiement classifier".format(time.time() - clf_timer))

if __name__ == "__main__":
    print("Starting the timer.")
    file_name = str(sys.argv[1]) if len(sys.argv) > 1 else ("logs/RC_2015-01")
    print file_name
    start_time = time.time()
    dict_of_subs = read_JSON_as_dict(file_name)
    print("It took {0} to read the file.".format(time.time() - start_time))

    dict_of_subs_no_function_words = remove_function_words(dict_of_subs)
    dict_of_subs_stripped = filter_votes_length(dict_of_subs_no_function_words)
    dict_reduced = {k: v for k, v in dict_of_subs_stripped.items() if v}

    dict_tagged = determine_if_popular(dict_reduced)
    for k, v in dict_tagged.items():
        write_csv(str(k)+".csv", v)
    print("total time: {0}.".format(time.time() - start_time))
```

```
import sys
from os import listdir
from os.path import isfile, join
import os
import numpy as np
import pandas
import random
import copy

from sklearn import neighbors, datasets, preprocessing
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn.metrics import accuracy_score, cohen_kappa_score, precision_recall_curve, average_precision_score,\
                                                    recall_score, f1_score

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB

classifiers = [
    SVC(kernel="linear", C=1),
    DecisionTreeClassifier(criterion="entropy", random_state=0),
    GaussianNB()]
names = [
    "Linear SVM",
    "Decision Tree",
    "Naive Bayes"]
'''
We need to get the number of lines in the file to ensure we have enough comments for our training set.
http://stackoverflow.com/questions/845058/how-to-get-line-count-cheaply-in-python
'''
def file_len(fname):
    with open(fname) as f:
        for i, l in enumerate(f):
            pass
    return i + 1

if __name__ == "__main__":
    sub_path = "subreddit_twomillion_csv"
    files = [f for f in listdir(sub_path) if isfile(join(sub_path, f)) if file_len(join(sub_path, f))>1750]
    for f in files:
        data = pandas.read_csv(os.path.join(sub_path, f))

        '''
        Init an encoder to turn our data into something the classifier can read.
        '''
        popular_encoder = preprocessing.LabelEncoder()
        data.popular = popular_encoder.fit_transform(data.popular.tolist())

        distinguished_encoder = preprocessing.LabelEncoder()
        data.distinguished = distinguished_encoder.fit_transform(data.distinguished.tolist())

        sentiment_encoder = preprocessing.LabelEncoder()
        data.sentiment = sentiment_encoder.fit_transform(data.sentiment.tolist())

        '''
        Convert the text into a vector where each index represents a word.
        The number in that index represents the frequency that word appears.
        '''
        cv = CountVectorizer()
        list_of_body = data.body.as_matrix()
        bag = cv.fit_transform(list_of_body)
```

```
        word_freq = bag.toarray()

        '''
        Select our features and stack it with the bag of words.
        Our labels indicate our popularity.
        '''
        columns = ["distinguished", "sentiment", "created_utc", "controversiality"]
        features = np.hstack((data[list(columns)].values, word_freq))
        labels = data.popular.values

        '''
        Split our data for testing and validation.
        Create/open a file to append results to.
        Iterate through the classifiers to fit the data and obtain predictions.
        '''
        X_train, X_test, y_train, y_test = train_test_split(features, labels, train_size=0.80, \
        test_size=0.20, random_state=random.randint(1,10000))
        scaler = preprocessing.StandardScaler().fit(X_train)
        X_train = scaler.transform(X_train)
        X_test = scaler.transform(X_test)
        file = open("results/clf_results_1750.dat","a+")
        print("\nTesting subreddit: {0} with {1} comments.\n".format(f.replace(".csv", ""), \
        file_len(join(sub_path, f))))
        for (name, clf) in zip(names, classifiers):
            clf.fit(X_train, y_train)
            y_pred = clf.predict(X_test)

            '''
            Obtain measurements to see how well our classifier works.
            '''
            precision, recall, thresholds = precision_recall_curve(y_test, y_pred)
            acc_score = accuracy_score(y_test, y_pred)
            avg_precision = average_precision_score(y_test, y_pred)
            avg_recall = recall_score(y_test, y_pred)
            f1 = f1_score(y_test, y_pred)
            kappa_stat = cohen_kappa_score(y_test, y_pred)

            print("Cohen Kappa: %0.3f, Accuracy:%0.3f, Precision:%0.3f, Recall:%0.3f, F1:%0.3f\n" \
                        % (kappa_stat, acc_score, avg_precision, avg_recall, f1))
```