PREDICTING REDDIT POST POPULARITY

by Jordan Segall and Alex Zamoshchin

1. INTRODUCTION

Reddit is a social news website where users create posts containing either links or text, which other uses can "upvote" or "downvote". Reddit has become increasingly popular in the past year having over two billion pageviews a month, with much of the traffic and discussion driven by its voting system. This project's purpose is to predict a post's future popularity it will receive at the time the post is created. More formally, this investigation hopes to predict the maximum score (number of upvotes minus number of downvotes) a post will receive in its lifetime.

While a post's popularity is usually related to the post's content, there are often other factors that determine how successful a post becomes. The focus of the investigation was to analyze how these factors – primarily, how a post is presented – play a role in predicting how popular a post will become. This included features such as the title of the post, the Subreddit of the post, and the time of day the post was created. While the problem of classifying a Reddit post without its content is quite difficult, an improvement in predictive ability would provide insight into the role of such features in determining a post's popularity.

2. APPROACH

A file consisting of a random sampling of 2,046,401 posts on Reddit was used to collect the data. Using a Python script and the Reddit API to connect to web pages corresponding to these posts, the features for these posts were captured in a separate output file. The following are features used in this investigation:

- Title: for each word *w* in the title of a post, the number of occurrences of *w*.
- Domain: the website for the url that is linked to in the post.
- Subreddit: the Subreddit (or category) that the post is posted to on Reddit.
- Over 18: a boolean indicating if the post is intended for people over the age of 18.
- Self: a boolean indicating if the content is on Reddit, or is a link to an external website.
- Author: the user who created the post.
- Created: the time of day (in 10 minute intervals) that the post was created.
- Thumbnail: the type of thumbnail (default, nsfw, custom) of the post.
- Self Text: if the post is not a link (ie. Self feature above is true), then for each word *w* in the content of the text, the number of occurrences of *w*.

3. EVALUATION METRICS

The algorithms were analyzed based on two evaluation metrics. For Multi-Class classification, the first was the classification error: for a post $(x^{(i)}, y^{(i)}, \hat{y}^{(i)})$ where $y^{(i)}$ and $\hat{y}^{(i)} \in \{1, \dots, K\}$ where *K* is the number of classes, $\hat{y}^{(i)}$ is the prediction, and $y^{(i)}$ is the actual label, then $classification\ error = \frac{\sum_i I\{y^{(i)} \neq \hat{y}^{(i)}\}}{m}$ where *m* is the number of posts.

The second metric was the Root Mean Squared Error (RMSE), where $RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2}$. Regression algorithms were evaluated solely on the RMSE, since classification error was not applicable.

Lastly, all implementations were compared to a baseline algorithm to compare relative improvements among algorithms. For the baseline algorithm, posts' scores were predicted as the median score of all posts within the dataset (three).

4. MULTI-CLASS NAÏVE BAYES

A Multi-Class Naïve Bayes Classifier (MCNB) was implemented, in which each classification corresponded to a different range of scores.

Version A: Full Feature Set (NB)

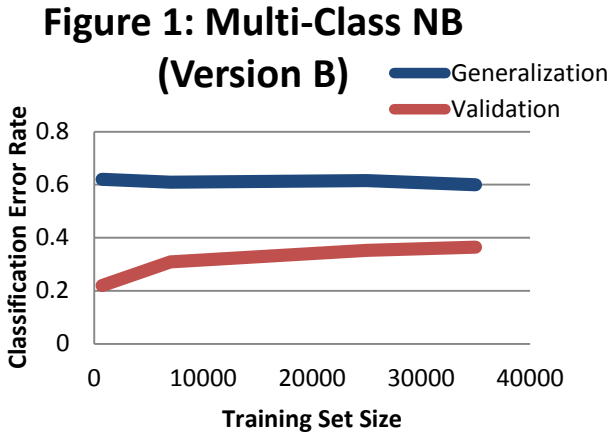
The first version of Multi-Class Naïve Bayes implemented utilized all of the features gathered in the data collection phase (refer to *Section 2* above). On a dataset of 7000 posts, the algorithm showed slight improvement over the baseline classification error. However, the classification errors decreased significantly when testing on the training set, indicating high variance and overfitting

in the MCNB algorithm. While the algorithm fit the training data very well, it was not extendable to data outside the training set. As a result, Version B was implemented to counteract the overfitting problem:

Version B: Reduced Feature Set (NB)
One method to reduce overfitting was to increase the size of the training set. While some attempts were successful in increasing the datasize, Matlab quickly ran out of memory when training on datasets greater than a certain theshold. As a result, the overfitting problem remained.

Another method of combating overfitting was to decrease the number of features. Not only would this help with classification error, but it would also enable use of larger datasets (by using less memory in Matlab). One way in which the number of features was decreased was by eliminating "Stop Words", which are short, commonly used words that are filtered-out in processing natural language text. Though this decreased the number of features, it did not provide substantial help in reducing overfitting. Nevertheless, all results shown are without stop words, due to the small improvement.

Ablative analysis, a process by which components are removed one at a time, was used to determine which features were the least helpful. The algorithm arrived at an optimal feature set consisting solely of Subreddit, Domain, and Author. *Figure 1* contains the error rates of this classifier. More detailed results can be found in *Table 1* of the *Appendix*.



While the validation error rates were closer to the generalization values, overfitting still existed in the MCNB classifier. However, little more can be done in this case: larger datasets are impractical due to limitations in Matlab, and decreased feature sets only hurt our generalization error.

Version B: Reduced Feature Set by Category (NB)
Since overfitting still remained in the dataset, the investigation shifted to determining if limiting the classification to a specific Subreddit would yield improved results. By running Version B's decreased feature sets on the Atheism Subreddit (now effectively 2 features), the algorithm was able to achieve a decrease in classification error by 21.487% and a decrease in RMSE by 16.138%. The results of testing on various Subreddits can be found in *Figure 2*.

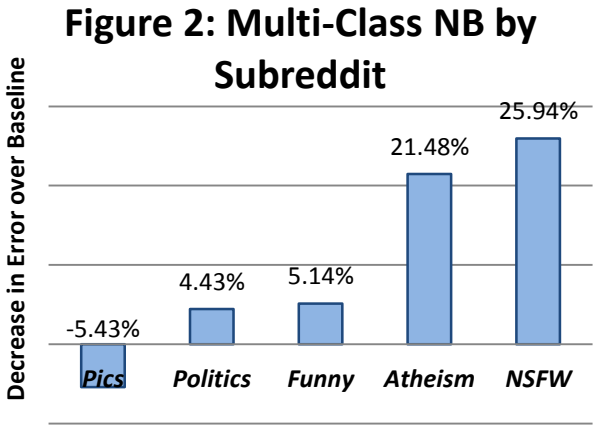


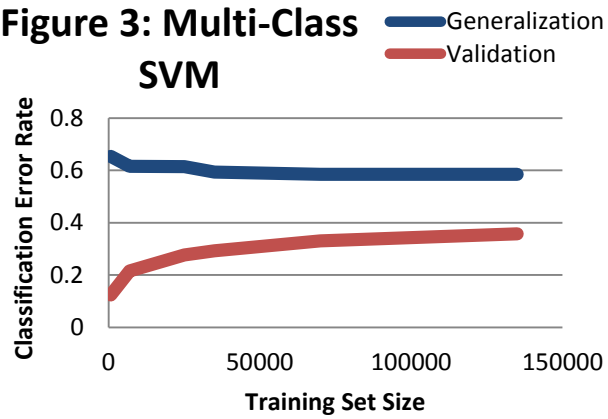
Figure 2 indicates that the success of the algorithm depends on the Subreddit. This may be because Reddit users that post consistently within a particular Subreddit follow trends that are easier to classify as popular, while other Subreddit categories are harder to predict. This is clearly evidenced by the fact that the algorithm performed the worst on the Pics Subreddit, where popularity relies primarily on the quality of the image, a feature inaccessible to our algorithm. On the other hand, Subreddits like Atheism or NSFW, in which very clear trends exist, were easier for the algorithm to classify and showed more substantial improvement over the baseline.

5. MULTI-CLASS SUPPORT VECTOR MACHINE

While SVMs are typically utilized in binary classification problems, they can be implemented as multi-class algorithms using a "one-vs-the-rest" strategy. The idea behind the "one-vs-the-rest" strategy is to convert the multi-class classification to a series of binary classification problems, where each sub-problem can be solved using existing methods. For example, suppose there are *K* possible classifications; the "one-vs-the-rest"

strategy would create K such classification problems, where for all $i \in K$ problems, a classifier separates the *ith* class from the remaining *K–1* classes. Therefore, the overall classification for a new $x^{(i)}$ test example is based on which of the K sub-classifications returned the largest output-value when testing on $x^{(i)}$.

A multi-class SVM was implemented using the above method to classify posts into a range of buckets, with each bucket representing a different range of scores. Performing a process of ablative analysis similar to that done above resulted in the same feature set of Naïve Bayes Version B. The classification errors of this SVM can be found in *Table 2* of the *Appendix* and are also shown below.



From above, the fact that the validation tests perform significantly better than the generalization tests indicate that the algorithm is still overfitting, despite the huge amount of data provided.

6. LINEAR REGRESSION

In contrast to the classification algorithms above, Linear Regression allowed a post to be predicted with a specific score.

Version A: Full Feature Set (LR)

Linear Regression was implemented with a Squared Loss objective function. For consistency, all training runs used a learning rate of 0.1 and ran for 500 iterations, which were necessary for the weights in the largest training set to converge. The generalization errors produced using all features can be found in *Table 3* of the *Appendix*.

It is evident that although at first the algorithm overfits the data, adding sufficient training examples is able to overcome this such that the validation and generalization errors are roughly the same. This is an example of high bias, meaning that the algorithm underfits the data. To provide more expressive power to the algorithm, a greater number of useful features would be required. However, since all possible features accessible by the Reddit API were already utilized, adding new features would require substantial work in scraping external-content links. In fact, it is reasonable that without access to post content – the main factor in a post's popularity – it would be very difficult to predict vote count with a high level of accuracy.

Version B: Expanded Feature Set (LR)

However, one of the primary problems of the existing model was that the existence of singular words (in the title or elsewhere) were not extremely indicative of popularity (the intuition being that humor, idioms, or complicated descriptions all involve combinations of words or phrases). Therefore, a new feature was created using a kernel involving the combination of pairs of words in the title. Meaning, a new feature *w1,w2* would be added for every *w1* and *w2* in the title (The corresponding permutation *w2,w1* would not be added). Therefore, it would be possible for the algorithm to detect trends using word combinations. However, only mixed results were achieved in exchange for an almost quadratic increase in time complexity. Since such a feature was deemed not successful, further improvement would require substantial effort to procure features outside of Reddit.

7. TF-IDF

One attempt to improve the results of the supervised learning algorithms above was to utilize *tf-idf*, or term frequency-inverse document frequency. *Tf-idf* assigns a weight to each word in a post, indicating the word's relative importance to the post. Normally, the *tf-idf* for a word *w* in a post *p* of a collection of posts *P* would be:

$$tf\text{-}idf_{w,p,P} = tf(w,p) * idf(w,P), \text{ where}$$

$$tf(w,p) = \frac{\#\ occurrences\ of\ w\ in\ p}{(max\ \#\ occurrences\ for\ any\ word\ in\ p)}$$

$$idf(w,P) = log\left(\frac{|P|}{\#\ documents\ containing\ w}\right)$$

However, alternative *tf-idf* approaches are more helpful for multinomial implementations such as the ones used in this investigation. As a result, an alternative *tf* was used, where:

$tf(w,p) = log(count(w,p) + 1)$ (Rener, 2003)

The idea behind applying *tf-idf* was to differentiate between the relative importance of words in the title and self-text features of posts, and thus better be able to differentiate posts based on their important words. However, with the inclusion of *tf-idf*, the algorithms showed only mixed results, indicating the substantial amount of noise present in the data.

8. STEMMING

Stemming is a technique used in natural language processing to reduce a set of words to a smaller subset of words consisting of base components of the larger set. For example, while the word *tree*, *tree's*, *trees'*, and *trees* are all different forms of the word *tree*, the differences between the forms are relatively insignificant in the process of predicting Reddit post popularity. Therefore, stemming would reduce the set of words that consist of *tree* to the single word *tree*.

The hypothesis behind implementing stemming was to reduce the feature sets for MCNB and SVM, since their validation error rates revealed that they were overfitting the training data. The NLTK (Natural Language Tool Kit) framework was utilized to help implement the stemming algorithm. Stemming decreased the feature sets for the algorithms significantly: for SVM on a 7000 size training set and 3000 size testing set, the number of features decreased from 39757 to 33070. Additionally, improvement from the baseline classification error increased from -1.32% to 2.49%, indicating that stemming helped considerably in the classification problem.

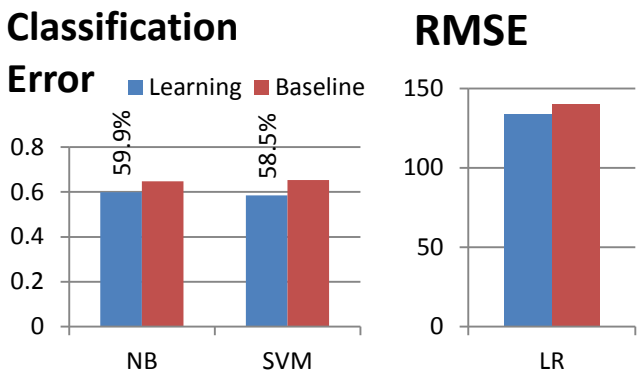
9. EVALUATION AND CONCLUSION

Version B of the Multi-Class Naïve Bayes Classifier achieved a 7.357% decrease in the classification error from the baseline metric, indicating that with large amounts of data simple predictive algorithms are preferable. Moreover, the fact that a decrease of 7.357% was even possible conveys how a considerable portion of a post's success is dictated by factors outside of its content (such as who posted it and where it was posted). Furthermore, when restricted to only the NSFW Subreddit, version B of the Multi-class Naïve Bayes Classifier was able to achieve a 25.935% decrease in the classification error from the baseline metric, indicating that certain Subreddits are much more predictable than others. However, the fact the MCNB classifier still overfits the data means more improvement in this algorithm might be possible.

Comparing the two classification algorithms reveals that the Multi-Class SVM results are very similar to those of Multi-Class Naïve Bayes. However, MCNB appears to learn more quickly than SVM on smaller data sets, with the classification errors for Naïve Bayes being superior for training sets of size 700 and 7,000. Nevertheless, the two algorithms appear to converge near a size of 25,000, and the SVM approaches an asymptotically lower error than the MCNB. However, when comparing the algorithms' validation errors, SVM's errors are much lower, indicating that SVM is overfitting the training data more-so than the MCNB.

While the SVM was marginally superior in reducing the classification error for large datasets, the MCNB algorithm's RMSE was slightly lower than that of the SVM for all training sizes. This indicates that MCNB may be marginally better than the SVM at making predictions that are closer to the actual score of the post, despite the prediction being correct or not. However, such a small increase may be attributed to just random sampling, and no strong conclusions can be made.

When comparing Classifier and Regression algorithms, a number of advantages and disadvantages must be considered. Although the advantage of a Regression algorithm is that it is able to produce a specific numeric prediction, rather than a categorical bucket, for this investigation such an advantage was not evident. In fact, since the problem of Reddit classification is so difficult, the Linear Regression algorithm was able to produce results only with a RMSE of 134.0524 even after a 4.1381% improvement. An exact prediction with such a deviation is not very useful, conveying the necessity of score ranges, or "buckets". In fact, after a 1.825% improvement, the Naïve Bayes Classifier was able to achieve a RMSE of 1.4468 buckets, meaning that most posts were classified within two buckets of the correct score range. This information may however be less useful, indicating a disadvantage of the Naïve Bayes classifier.



Overall, it was found that Version B of the MCNB and SVM classifiers were slightly more desirable than Version A of the Linear Regression algorithm. With more useful results, and substantially faster runtime, the classification algorithms were able to achieve a better result with a much simpler feature set. This may convey the relative lack of indicative power of features such as the title in predicting a post's popularity (likely due to the substantial noise). Moreover, perhaps with large datasets, simple algorithms are better at providing more useful and accurate predictions.

10. SUGGESTIONS FOR IMPROVEMENT

As alluded to earlier, possible suggestions for improvement include larger datasets, runnable in environments with more memory, as well as adding features outside of the Reddit API. Possible features include determining the content of the link, or applying Computer Vision to categorize images. However, it is important to note that there are intrinsic characteristics of the Reddit community that make the problem of classifying a post difficult: as described in *Reference 1*, the popularity of a post is sometimes not even attributed to the content of the post, but instead the discussions that occur within the comments section of that post. However, since posts begin with no comments upon creation, this would not be possible.

11. REFERENCES

1. http://www.quora.com/reddit/How-do-you-get-to-the-front-page-of-Reddit
2. http://lingpipe-blog.com/2009/02/16/rennie-shih-teevan-and-karger-2003-tackling-poor-assumptions-naive-bayes-text-classifiers/
3. http://cs229.stanford.edu/proj2011/PoonWuZhang-RedditRecommendationSystem.pdf
4. The random sample of posts: http://www.infochimps.com/datasets/csv-dump-of-reddit-voting-data
5. The list of stop words removed: http://www.translatum.gr/forum/index.php?topic=24

12. APPENDIX

Table 1: Multi-Class Naïve Bayes Version B Results (CE = Classification Error, RMSE = Root Mean Squared Error in buckets)

Train Size	Generalization CE	Baseline CE	Decrease in CE	Generalization RMSE	Baseline RMSE	Decrease in RMSE	Validation CE	Validation RMSE
700	0.6200	0.6467	4.129%	1.6361	1.6299	-0.611%	0.2200	0.5014
7000	0.6097	0.6637	8.543%	1.5033	1.5000	-0.220%	0.3087	1.3171
25000	0.6150	0.6604	6.975%	1.5116	1.5220	0.683%	0.3526	0.6783
35000	0.5994	0.6470	7.357%	1.4468	1.4737	1.825%	0.3641	1.3264

* Larger data sets resulted in an *Out of Memory* error in Matlab

Table 2: Multi-Class SVM Results (CE = Classification Error, RMSE = Root Mean Squared Error in buckets)

Train Size	Generalization CE	Baseline CE	Decrease in CE	Generalization RMSE	Baseline RMSE	Decrease in RMSE	Validation CE	Validation RMSE
700	0.6533	0.6467	-1.02%	1.6693	1.6299	-2.42%	0.1243	0.8693
7000	0.6160	0.6637	7.19%	1.5339	1.5000	-2.26%	0.2157	1.1280
25000	0.6140	0.6604	7.03%	1.5191	1.5220	0.19%	0.2765	1.1972
35000	0.5934	0.6467	8.24%	1.4566	1.4737	1.16%	0.2921	1.2138
70000	0.5851	0.6495	9.92%	1.4502	1.4861	2.42%	0.3300	1.2442
135000	0.5850	0.6534	10.47%	1.4463	1.4900	2.93%	0.3577	1.2730

Table 3: Linear Regression Version A (RMSE = Root Mean Squared Error)

Train Size	RMSE	Baseline RMSE	Decrease in RMSE	Validation RMSE
700	160.3180	163.0698	1.69%	72.0005
7000	134.0524	139.8390	4.14%	126.727

* Larger data sets resulted in an *Out of Memory* error in Matlab.

Combined Project with CS221. Only basic versions of Multi-Class Naïve Bayes and Linear Regression were implemented for CS221.