Predicting Comment Karma by Subreddit

Yoav Zimmerman (304125151)
CS 260: Machine Learning Algorithms

December 11, 2015

1 Abstract

In this report, I approach the problem of predicting the score of a Reddit comment given the body text of the comment. To use as features, I experiment with a bag-of-words model, TF-IDF vectorization, and several more metadata features extracted from the text of each comment. I fit a Ridge Regression model on several different subreddits, achieving a 36.67 root squared mean test error on the subreddit */r/hiphopheads*. I achieve better results fitting Logistic Regression models, achieving an 82.4% precision with a binary classification model on the subreddit */r/music*, an improvement of 7.4% over the baseline model. I also achieve a 62.5% precision with a multi-class classification model on */r/music*, an improvement of 13.4% over the baseline model.

2 Motivation

Voting systems are a standard part of many online communities. Whether the measure is "likes" on a facebook status, hearts on a picture in Instagram, or upvotes on a submission from Reddit, these voting systems often define which online content receives the most attention. Understanding the dynamics of user behavior in voting systems has applications in the world of online advertising, voting algorithm design, and viral marketing. Reddit is one such website that aggregates link submissions, and allows users to either upvote or downvote other users comments on submissions. By attempting to formulate a text-based model for predicting Reddit *comment karma* on a given subreddit, I hope to gain insights into what text elements make a comment "good" or "bad".

3 Background

Although there has some previous academic research on the subject of popularity of online content, there has been very little work exploring the prediction of the popularity of user-generated text such as reddit comments, in particular. Szabo [2] was able to forecast the long-term (30 day) popularity of Digg content with relative success using short-term (2 hour) user access and popularity data. There have also been several student projects attempting to predict popularity of user-submitted comments of websites such as HackerNews and Reddit. Lamberson et. Al [1] is an example student project that uses a large variety of features to attempt to predict scores of Reddit comments by using a prediction model, with relative success focusing

on features other than text.

4 Data

a. Dataset

The dataset used in this project was the publicly available dataset of 1.7 billion reddit comments [5]. Although this dataset is very large, the features are relatively sparse; each comment only consists of basic features such as body text, author, time posted, and score of the comment. To narrow down to a more usuable data set, only comments from the month of September 2015 were used. Next comments were grouped by subreddit so that seperate models could be trained on each subreddit. Table 1 shows the size of various subreddits used through this project.

Subreddit	Number of Comments
/r/askreddit	1,000,000
/r/videos	550,192
/r/movies	288,522
/r/hiphopheads	137,363
/r/music	114,458
/r/askscience	27,747

Table 1: Size of subreddit datasets

b. Features

The most significant feature used in experiments was a vector representing the text body. First, the text body was tokenized on whitespace and stripped of non-alphanumeric characters. Two approaches were used to vectorize this list of tokens. One is a unigram **bag-of-words model** in which the value c_i corresponds to the amount of time token w_i appears in the comment body. Another representation often used on text corpus' is **Term-Frequency ·Inverse-Term-Frequency** (TFIDF), where the term frequency is the same counts value as in the bag-of-words model, and the inverse term frequency is a measure of how much information a "word" provides.

$$idf(t) = log\left(\frac{\sum_{d \in D} 1}{\sum_{d \in D; t \in d} 1}\right)$$

In addition to the body vector features, there were 7 metadata features extracted from each comment:

1. Character Count
2. Token Count
3. Average Token Length
4. Sentiment Analysis using the AFINN-111 dataset [6]
5. Number of question (?) marks
6. Number of exclamation (!) marks
7. Hour of Time posted

5 Models

There were two families of models applied to this problem.

a. Regression

The most intutive model to predict the score of a comment given it's features is to use a regression model. An advantage of a regression model is it naturally maps well to the dataset, as the score labels associated with each comment are continuous integers. The Ridge Regression algorithm was used, where the regularization parameter λ was tuned by cross-validation testing.

b. Classification

Another approach to modeling this problem is to bucket the comments into k score buckets and then run a classification algorithm on them. The two classification algorithms experimented with were Logistic Regression and Multinomial Naive Bayes.

Classification does introduce a complication that does not arise in the case of regression. The number of buckets brings another "hyperparameter" into the algorithm- with a larger amount of buckets the model will have predict a finer granularity of scores, but too many buckets will result in a too difficult of a model to learn.

b..1 Bucket Classification

To address the skewed nature of the dataset towards lower scores, comments were assigned to a score bucket according to a log scale.

$$bucket_n = \begin{cases} 0 & score_n < 0 \\ log_2(score_n) + 1 & score_n \geq 0 \end{cases}$$

b..2 Special Case: Binary Classification

Another approach was to decide on a score threshold to divide the dataset into "positive" and "negative" comments. Note that this is a special case of bucket classification above, where number of buckets is equal to 2. This simple model is the easiest to learn over, but gives us the least granularity in the task of predicting a new reddit comment.

6 Implementation

The large size of the dataset proved to be the most challenging part of this project. The feature processing and learning models were implemented using the Apache Spark [8] distributed computing framework. The Spark framework was a good fit due to it's capabilities for distributed computation and that it comes with the MLLib API for Ridge Regression, Multinomial Naive Bayes, and Logistic Regression. Spark also comes with helper scripts to set up clusters to run on Amazon's EC2 cloud computing services. A cluster of 4 machines on Amazon EC2 were used to train models with greater than 300,000 comments. When transforming comments into unigram vectors, the vocabulary size was limited to a maximum of 50,000 of the most frequently used words for scalability reasons. Simillarly, for TF-IDF vectorization the hashing trick [7] was used to restrict the feature vector to a maximum of 50,000 words.

The code written for this project is open-source and publicly available on github.[1]

7 Results/Evaluation

a. Regression

To evaluate the effectiveness of regression models, the **Root Mean Squared Error (RMSE)** can be used to compare test errors across models. Intuitively, this can be thought of as how far, on average, a score prediction is from the actual score of a comment.

$$\sqrt{\frac{\sum^N (\mathbf{w^T x_n} - \mathbf{y_n})^2}{N}}$$

Table 2 shows the testing RMSE after training and testing on 10,000 comment subset of four

[1] `https://github.com/yoavz/predict_reddit_comments`

different subreddits. Each set of comments was split into 70% training and 30% testing. The three variants of feature vectors used were:

1. Unigrams + Metadata features
2. TF-IDF + Metadata features
3. Only Metadata features

	Feature Variants		
Subreddit	Bag	TF-IDF	Metadata
/r/askreddit	144.70	147.24	118.76
/r/movies	85.39	84.39	53.43
/r/hiphopheads	36.67	36.71	38.63
/r/askscience	117.95	122.12	116.28

Table 2: Linear Regression on 10k comment subsets

None of these regression models achieve a satisfactory RMSE. The feature variant does not seem to have much of an effect; TF-IDF performs very similarly to unigrams, and metadata features even outperform both in some cases. With this small and noisy dataset, the models are *severely overfitting* to the training set, as evidenced by Figure 1, which shows training and testing RMSE against different values of regularization parameter λ.

Figure 2 shows the results of larger scale learning using the unigrams feature variant and the entire comment dataset for each subreddit. As can be seen, increasing the size of the dataset has very little effect on the final test RMSE of the model. In the case of the subreddit */r/askreddit*, for example, the test error actually increases, although the model is learned over 1,000,000 comments instead of the previous 10,000 comments.

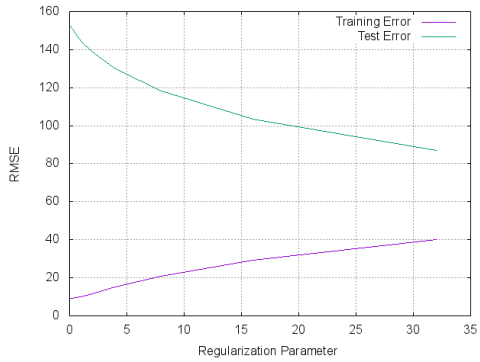

Figure 1: RMSE against λ for /r/movies

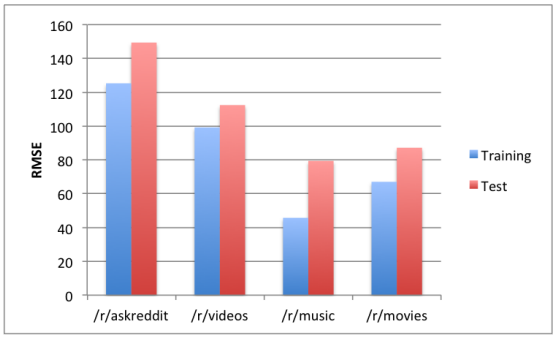

Figure 2: Linear Regression on full comment datasets

b. Binary Classification

The metric used to evaluate the effectiveness of classification models was the **precision**, or the percentage of comments that are successfully assigned to the correct class by the trained model.

Figure 3 shows the precision of the Naive Bayes and Logistic Regression models on several different subreddits. It is also displayed next to the baseline model precision, which ignores the features and predicts a class based on the prior probabilities of the two classes. Logistic Regression consistently performed better than Naive Bayes and the baseline by a small amount. The best performing model is Logistic Regression on

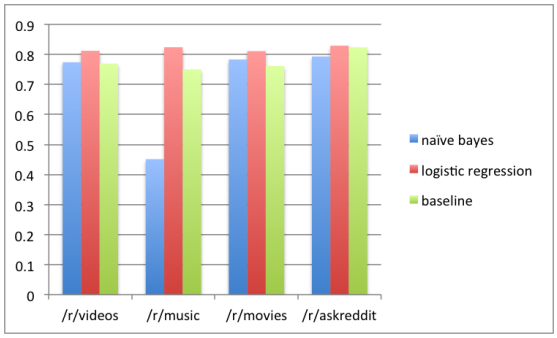

Figure 3: Binary Classification

the */r/music* subreddit, achieving a 7.4% improvement over the baseline model to achieve an 82.4% precision.

c. Multi-Class Classification

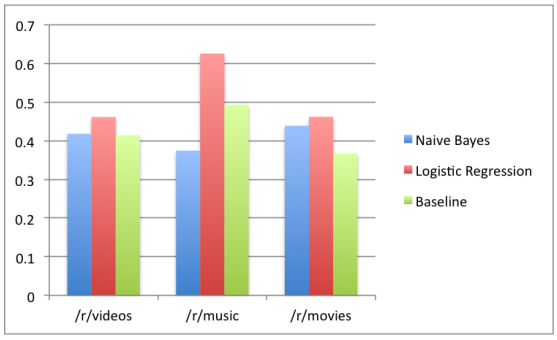

Figure 4: Multi-Class Classification

Figure 4 shows the precision of the Naive Bayes and Logistic Regression (with the three feature variants) next to the baseline model on several subreddits. In all cases, both prediction models do better than the baseline. The most successful prediction model is a Logistic Regression on */r/music*, where it improves the precision of the baseline by 13.4% to achieve a 62.5% precision over 14 score buckets.

8 Analysis

The performance of the models were varied, depending on subreddit. In general, the classification models performed better than the regression models, but neither performed spectacularly. There are several potential reasons for why this is. One reason may be that the simple tokenization and metadata feauteres do not correctly capture the inherent quality of a comment. The correlation between reddit comments and score may be mostly a function of other contextual comment features, or is embedded in a deeper semantics that are not properly conveyed by unigrams. Another potential reason is insufficient data; many ambitious text and image learning problems take days or weeks to train before any meaningful results are reached.

9 Future Work

In future work, it would be useful to augment the data set with a richer set of features that includes context about the parent post and surrounding posts. For example, perhaps the score of a parent post or sibling post(s) are correlated with the score of the given comment. Although richer features such as context are not part of the 1.7 billion reddit comment dataset, it is possible to build a web crawler to access additional features. There are also richer features that could be introduced over the original dataset, such as word2vec [4], which maps each token to a semantic vector space. Perhaps certain areas of the comment body, such as the last sentence or the first sentence, are more important in deciding the final score. Unfortunetely, these options could not be explored in this project due to time constraints.

References

[1] Daria Lamberson, Leo Martel, S. Z. Hacking the hivemind: Predicting comment karma on internet forums.

[2] Gabor Szabo, B. A. H. Predicting the popularity of online content. *Communications of the ACM 53*, 8 (August 2010), 80–88.

[3] Lakkaraju, K. H. Demystifying content popularity on reddit.

[4] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *CoRR abs/1301.3781* (2013).

[5] Reddit. I have every publicly available reddit comment for research. 1.7 billion comments @ 250 gb compressed. any interest in this? `https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment`, 2015.

[6] Sliwinski, A. Sentiment. `https://github.com/thisandagain/sentiment`, 2013.

[7] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., and Attenberg, J. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), ACM, pp. 1113–1120.

[8] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2010), HotCloud'10, USENIX Association, pp. 10–10.