# Classification of Reddit Comments

Jizhou Wang Nahiyan Malik Abby Leung

**Abstract**

In this project, we analyzed text data from Reddit, a popular social media forum, and built classification models to predict what subreddit a comment came from. Specifically, we investigated the performances of the Bernoulli Naive Bayes model, multinomial Naive Bayes, logistic regression, support vector machine (SVM), neural network and random forest on the Reddit comments dataset. By extracting word features from the comments, we have found that the self implemented Bernoulli Naive Bayes model had an accuracy of 0.4941. Out of all the other models tested, we observed that the neural network model had the best accuracy of 0.5757 and the SVM model had the fastest fitting and training runtimes.

## 1 Introduction

Text classification is defined as assigning text documents to predefined categories, in this case, comments to specific subreddits. We first start with training set $D = (d_1, \ldots, d_n)$, comments that are labelled with a category $L \in C$ , the set of all 20 subreddits. We then find the classification model $f : D \mapsto C$ that predicts the class of Reddit comments $f(c) = L$ at the highest accuracy. (Zhang et al. 2008)

The subreddits were as follows: hockey, nba, leagueoflegends, soccer, funny, movies, anime, Overwatch, trees, GlobalOffensive, nfl, AskReddit, gameofthrones, conspiracy, worldnews, wow, europe, canada, Music, baseball.

The Bernoulli Naive Bayes model is based off of word counts where when 1, a word exist in the comment when 0, it does not exist. For the best Bernoulli model, our corpus was preprocessed with lower casing, stemming as well as removal of punctuation, English stopwords and words that occurred only once. The highest accuracy for this model was 0.4941 with a Laplace smoothing of 0.5.

Additional models were trained on the dataset including multinomial Naive Bayes, logistic regression, SVM and a neural network. The best accuracy was achieved through a TF-IDF feature set using the neural network model, with an accuracy of 0.5757. Word2vec was also used to train word embeddings from the Reddit comments. The highest accuracy achieved based on word embeddings was 0.5696.

## 2 Related Works

Text classification is a well studied problem in machine learning. There have been many previous approaches in the area.

**Multi-word with support vector machine**

Zhang et al. proposed a method to implement multi-word extraction from document based on syntactical structure and k-mismatch for subtopic representation, determining the subtopic of the content of a document. They carried out a series of experiments on the Reuters-21578 dataset and found that subtopic representation outperforms general concept representation. Furthermore, representation using individual words outperforms representation using multi-word. (Zhang et al. 2008)

**Multinomial Naive Bayes**

Kibriya et al. worked on improving the multinomial Naive Bayes classifier using locally weighted learning. They showed that though the performance of multinomial Naive Bayes can be improved using locally weighted learning, SVM is still the best method to maximize accuracy. (Kibriya et al.)

## 3 Dataset and Setup

Given 70,000 comments as training samples separated into 20 equally sized subreddit categories, many of the entries are different in length, vocabulary used, etc. Before feature extraction, we had to preprocess these

entries. These methods include tokenization, removing stop words, special characters and punctuation as well as stemming or lemmatization to simplify our assumption of the corpus vocabulary and reduce the feature vector size. Some models like Bernoulli Naive Bayes simplify assumptions even further by removing infrequently occurring words, frequently occurring stopwords and even punctuations. This could provide a gain in its prediction performance, as the model itself is simplistic.

More complicated models such as neural networks could take in an entire corpuses without preprocessing and in theory, provide a prediction accuracy much higher than the simpler models, although it may would require millions of training examples. As a whole, all the models benefited from the preprocessing.

## 4 Proposed Approach

Many feature extraction methods were implemented, including word counts based on the bag of words model and TF-IDF for both words and characters (Ramos 2003). For further experiments, word embeddings such as word2vec and doc2vec (Goldberg et al., 2014) were used as well. Custom features were also tried: total number of words, upper cases, lower cases and characters in a comment.

The Bernoulli Naive Bayes model is very simple. It is based off of a binary 1 and 0 count associated with the presence of a word. Many runtime optimization strategies are used in the fitting and predicting function of this model as it is the only self implemented model. This includes sparse matrices (Davis et al. 2011) as they reduced memory usage by only saving the indices where 1's occurred in the feature vector. Vector operations on matrices also reduced the runtime of the implementation by a significant amount. For the Bernoulli model, the top search space is defined on different preprocessing methods with a default Laplace smoothing of 1. After finding the preprocessed feature vector with the with highest accuracy score, a grid search is conducted on the Laplace smoothing hyper-parameters of 0.5 increments from 0 to 3. We then chose the smoothing parameter with the highest scoring result.

Other models were also trained including logistic regression, support vector machines with a linear kernel, multinomial Naive Bayes and multilayer perceptron (neural network). Similarly to the Bernoulli Naive Bayes analysis, grid search was used to tune the hyperparameters for each of the models. Two particular changes were effective; the Laplace smoothing parameter was changed to 0.6 for multinomial Naive Bayes and the penalty parameter C was changed to 0.2 for SVM. For the neural network, an adaptive learning rate was used with the solver 'adam' and 100 hidden layers, which is the default value in scikit learn. A random forest ensemble (Hedge et al. 2017) was done as well. All other hyperparameters were kept to scikit learn defaults.

In recent years, word embeddings have successfully been applied to text classification problems (Tang et al. 2014). As such, a Google pre-trained word2vec model was used to train a neural network. Additionally, a word2vec model was trained using the Reddit comments, which achieved better results. Doc2vec was also used as an unsupervised method to generate vectors using the Reddit comments.

Our model validation pipeline consisted of a K-Fold cross validation with 5 folds.

## 5 Results

For the Bernoulli Naive Bayes model, tests were conducted on 7 different preprocessed text models with 5 different Laplace smoothing values shown in Table 1 below. The best model from the grid search had stemming, lowercase words, punctuation and single word occurrences removed with an alpha of 0.5 for Laplace smoothing. Its validation accuracy was 0.4941 (sd. 0.00535). The average runtime for fitting was 0.1480 (sd. 0.00368) seconds and for predicting was 14.91 (sd. 0.632) seconds.

| Abbreviations | Preprocessing |
|---|---|
| Lo | Lowercased |
| P | Punctuations Removed |
| S | Stemming |
| SS | Space Separated |
| ST | Stopwords Removed |
| #W | #Word Occurances Removed |

Table 1: Preprocessing abbreviations depicting each step that was tested.

| Preprocess | Features | $\alpha = 0$ | $\alpha = 0.5$ | $\alpha = 1$ | $\alpha = 1.5$ | $\alpha = 2$ |
|---|---|---|---|---|---|---|
| SS | 235689 | 0.4309 | 0.4235 | 0.3940 | 0.3635 | 0.3296 |
| SS, Lo | 205930 | 0.4365 | 0.4397 | 0.4142 | 0.3852 | 0.3550 |
| SP | 112687 | 0.4730 | 0.4748 | 0.4549 | 0.4360 | 0.4191 |
| SP, Lo | 89320 | 0.4783 | 0.4892 | 0.4719 | 0.4557 | 0.4414 |
| SP, ST, Lo | 67391 | 0.4860 | 0.4918 | 0.4764 | 0.4641 | 0.4515 |
| SP, ST, Lo, 1W | 29362 | 0.4873 | **0.4941** | 0.4799 | 0.4683 | 0.4589 |
| SP, ST, Lo, 1W, 2W | 22202 | 0.4840 | 0.4931 | 0.4799 | 0.4685 | 0.4585 |

Table 2: Bernoulli Naive Bayes accuracy results. The model was fitted on features with various preprocessing steps.

For the rest of the models, both bag of words and TF-IDF feature sets were tested, with TF-IDF always having better accuracy. This makes sense because compared to bag of words that simply counts words as features, TF-IDF applied frequency-inverse document frequency, giving more importance to words appearing fewer times in the corpus. The following are the accuracy and runtime results for the models:

| **Model** | **Accuracy** | **Runtime (fit)** | **Runtime (predict)** |
|---|---|---|---|
| Logistic Regression | 0.5494 (sd. 0.003036) | 45.0169 secs (sd. 1.411083) | 2.7242 secs (sd. 0.299077) |
| SVM | 0.5643 (sd. 0.003854) | 15.6515 secs (sd. 1.684931) | 2.4066 secs (sd. 0.157584) |
| Multinomial Naive Bayes | 0.5509 (sd. 0.001818) | 22.8844 secs (sd. 1.477066) | 2.9354 secs (sd. 0.220009) |
| Neural Network | 0.5757 (sd. 0.003649) | 560.0027 secs (sd. 11.918611) | 2.5896 secs (sd. 0.302799) |

Table 3: Results from the scikit learn models. Four different models were used with the neural network achieving the best accuracy.

The neural network had the best results, although the training time was drastically more than the other models. This is to be expected due to more than 22,000 features per sample. Despite the large number of features, the model converged after 11 iterations as can be seen in Figure 1. Any more iterations resulted in overfitting as the validation accuracy started to decrease.

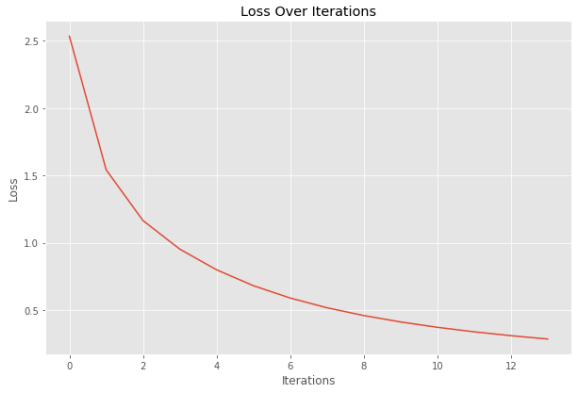


Figure 1: The loss from the neural network model over its iterations. The training was stopped at iteration 11.

Due to the large number of features and 20 different classes, the random forest classifier was a good ensemble method. Because each tree in the random forest would depend on independent features, it could be a good way to generalize given enough tree estimators. However, with 1000 trees, only an accuracy of 0.4724 was achieved.

In order to more closely examine the features and the most effective ones, a chi-squared test was used. Incremental features based on the chi-squared test were used to measure accuracy, as can be observed in Figure 2, which showed that the more than 22000 features were in fact necessary.

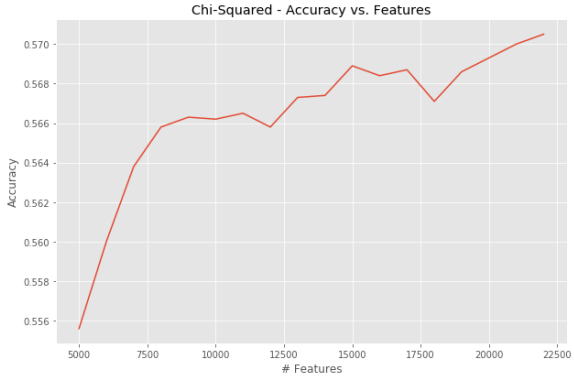


Figure 2: The number of top chi-squared based features used and the associated accuracy.

To reduce the number of dimensions from a high number of features, truncated singular value decomposition (SVD) from scikit learn was used. It is a memory efficient method for dimension reduction compared to principal component analysis (PCA). Many different numbers of components were attempted, but even with 2500 components, SVD only accounted for 0.69 variance.

We then tried word embeddings. Word embeddings are vector representations of words based on cosine similarities, which provides an effective way of comparing resemblances in words in a classification setting (Pennington et al. 2014). Two different pretrained models, Google News and Twitter, were used to train a neural network, which resulted in accuracies of 0.4496 and 0.4642 respectively. It is important to note that the Google News corpus and the Reddit comments vary a lot in context. Although the Twitter corpus may be more similar to the Reddit comments, Reddit comments are generally more structured, contextual and longer in length.

Due to the shortcomings of both pre-trained models, a word2vec model was trained using the Reddit comments from the training set. Because the training dataset was not very large, a continuous bag-of-words (CBOW) model was used, which tries to predict the current word based on the surrounding context (Mikolov et al. 2013).

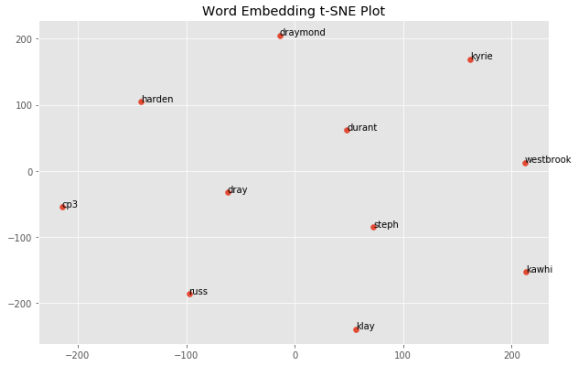


Figure 3: A group of similar words to 'kawhi', the basketball player - Kawhi Leonard. The other words are names of players that often discussed alongside Kawhi Leonard.

The CBOW word2vec model was used to train a neural network, which resulted in an accuracy of 0.5696. Another method, doc2vec, was used to train a neural network using the Reddit comments from the training set, which resulted in an accuracy of 0.52.

In an effort to acquire more training data, Google BigQuery was used to query for Reddit comments that have been scraped by Google. For each subreddit and month in 2018, 5,000 comments were queried and saved as additional training data, which resulted in 1,270,000 total training samples. A word2vec model was trained and validated against the larger dataset. Despite this, however, the accuracy did not improve as the best accuracy from the larger dataset was 0.5409.

Both the word2vec and doc2vec models were outperformed by the neural network results based on the TF-IDF features from the original, smaller dataset.

## 6 Discussion and Conclusion

From the experiments above, we have shown that a simple model like Bernoulli Naive Bayes may not be the best performing model compared to other complex models in sentiment classification. More involved models such as SVM, neural networks and even word embeddings do not produce great results. There are various reasons for this. A 20 class classification task is quite complex, especially when the Reddit comments can also vary in quality and length. A larger dataset and a custom Reddit trained word embedding model could result in better accuracy. Because of the transient nature of some of the subreddits, the larger dataset would preferably need to be from the same timeframe as the test set for maximal results.

Other sentiment analysis work were conducted in predicting popularity of Reddit comments across different subreddit communities (Horne et al., 2017) This can be translated to a type of classification problem if the predictions surpassed a threshold for it to be considered popular within that subreddit (Terentiev et al. 2014). In their work more features were used in making predictions than just the text corpus, including timestamps, flair, user activity. A future direction could be to extract and include similar features in our classification task.

Another suggestion to try is to increase our training data sample size, as stated above, in order to train more complex models such as Convolutional Neural Networks (CNN) as suggested by (Aich et al., 2019). Compared to other neural network models such as a Recurrent Neural Network (RNN), CNN is suggested to have higher accuracy for text classification tasks.

## 7 Statement of Contributions

Jizhou worked on the implementation and evaluation of Bernoulli Naive Bayes and conducted tests on accuracy, runtime and feature selection for that model.

Nahiyan worked on analyses of logistic regression, SVM, neural network, multinomial Naive Bayes, random forest, word2vec, doc2vec and feature selection.

Abby worked on the implementation of the neural network model and the report.

## References

1. Horne, Benjamin D., Sibel Adali, and Sujoy Sikdar. "Identifying the social signals that drive online discussions: A case study of Reddit communities." 2017 26th International Conference on Computer Communication and Networks (ICCCN). IEEE, 2017.

2. Terentiev, Andrei, and Alanna Tempest. "Predicting Reddit Post Popularity Via Initial Commentary." nd): n. pag (2014).

3. Aich, Satyabrata, Sabyasachi Chakraborty, and Hee-Cheol Kim. "Convolutional neural network-based model for web-based text classification." International Journal of Electrical & Computer Engineering (2088-8708) 9 (2019).

4. Kibriya A.M., Frank E., Pfahringer B., Holmes G. (2004) Multinomial Naive Bayes for Text Categorization Revisited. In: Webb G.I., Yu X. (eds) AI 2004: Advances in Artificial Intelligence. AI 2004. Lecture Notes in Computer Science, vol 3339. Springer, Berlin, Heidelberg

5. W. Zhang, T. Yoshida, X. Tang. Text classification based on multi-word with support vector machine. Knowledge-Based Syst., 21 (2008), pp. 879-88

6. Ramos, Juan. "Using tf-idf to determine word relevance in document queries." Proceedings of the first instructional conference on machine learning. Vol. 242. 2003.

7. Goldberg, Yoav, and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method." arXiv preprint arXiv:1402.3722 (2014).

8. Davis, Timothy A., and Yifan Hu. "The University of Florida sparse matrix collection." ACM Transactions on Mathematical Software (TOMS) 38.1 (2011): 1.

9. Hedge Yashaswini, Padma S.K. "Sentiment Analysis Using Random Forest Ensemble for Mobile Product Review in Kannada." 2017 IEEE 7th International Advance Computing Conference (IACC). IEEE, 2017.

10. Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, Bing Qin. "Learning Sentiment-Specific Word Embedding for Twitter Classification." Proceedings of the 52nd Annual Meeting of the Association of Computational Linguistics. ACL 2014.

11. Jeffery Pennington, Richard Socher, Christopher Manning. "Glove: Global Vectors for Word Representation" Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). EMNLP 2014.

12. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space" arXiv preprint arXiv:1301.3781v3 (2013).