Comparative Analysis on Classification Models for Reddit Comment Classification

Michael Li(260869379), Ethan Anderson(260569898) and William Zhang(260865382)

October 21, 2019

Abstract

Text classification is a key aspect of machine learning. In this paper, we discuss various methods of classifying Reddit comments to determine which subreddit they were posted on. The dataset is comprised of 20 subreddits and 70,000 comments. Various transformers were tested, notably Count Vectorizer and Term Frequency Inverse Document Frequencty (TFIDF) Vectorizer. Our final pipeline composed of a TFIDF Vectorizer, as well as a Voting Classifier that is comprised of a Linear Support Vector Classifier, a Multinomial Naive Bayes and Logistic Regression, we managed to obtain an accuracy of 58.3% on the test set.

1 Introduction

Text classification is a very key component of Natural Language Processing. The objective of this paper is to explore various methods in classifying text from popular social media forum Reddit. The website boasts a wide range of communities, ranging from music to sports. The objective in this case is to predict from which community a comment came from. The dataset is a collection of 70,000 comments with equal distribution from 20 subreddits.

In order to preprocess the data into vectors for our various models, a pipeline was employed using Scikit Learn's Term Frequency Inverse Document Frequency (TFIDF) Vectorizer in order to transform our data and classify the most important words.

After preprocessing the data, various models were tested, ranging from Naive Bayes to ensemble methods and deep learning models. In order to tune all hyperparameters, a grid search method was used to test out various permutations. After tuning, the highest accuracy was found using a Voting Classifier composed of three linear models: Linear Support Vector Classification (LinearSVC), Multinomial Naive Bayes (MNB) and Logistic Regression. Below is a table of the best accuracy achieved by each of the models we examined. All the accuracy values were generated from a 5-folds cross validation.

Model	Accuracy (%)
Logistic Regression	54.98
LinearSVC	56.22
Multinomial NB	56.75
Voting Classifier	56.94

Table 1: Comparison of model accuracy

2 Related work

Text classification is one of the most common problems in the field of natural language processing. The approaches used in this report represent, in general, overly simplistic or inefficient models such as Bernoulli naive Bayes or decision tree models (although these models can still work quite well in certain circumstances).[1] In comparison, there are several much more complex state-of-the-art natural language processing and text classification models such as Bidirectional Encoder Representations from Transformers (BERT) and XLNet,[2, 3] both recently published models by Google AI based on Transformer[?]. Transformer is notable because it provides bidirectional contextual information for each token in the sentence, allowing for a much greater predictive power over a range of applications including text classification. These transformer based models are the current state-of-the-art for NLP, however they come with massive computational and memory usage costs when compared to the models used in this report.

3 Dataset and setup

3.1 Dataset

The dataset is composed of comments from 20 subreddits on the Reddit website, an online discussion forum where users exchange comment on various subjects. The goal of this project is to correctly classify each comment to its corresponding subreddit.

3.2 Setup

Starting with raw text data, a preprocessing function was built. First, all sentences are transformed to lowercase, then each one of them is tokenized using the word_tokenize function from NLTK. Afterward, a regular expression was used to remove all punctuation, non-alphabetical char and whitespaces. Finally, each individual tokens is stemmed using the SnowballStemmer from NLTK. After the first round of preprocessing, we proceeded to feature exploration. We plotted the average word count and the 50 most frequent words for each subreddit (see example in Figure 1). It is possible to see that subreddits such as music, Canada and conspiracy clearly have longer comments on average (respectively, 72, 52 and 51 words) compared to other subjects. However, they all have a very high standard deviation which is why the word count was not used as a feature. In addition, by inspecting the 50 most common words for each subreddit, we noticed that certain words such as "nt", "like", "would" (see Appendix for the list of stopwords) appear regularly in the top 10 most frequent words. In fact, those words do not add any information. They can not be used to find the context, because they appear in every subreddit. Hence, these words are added to the list of stopwords from NLTK in order to reduce the dimensional space.

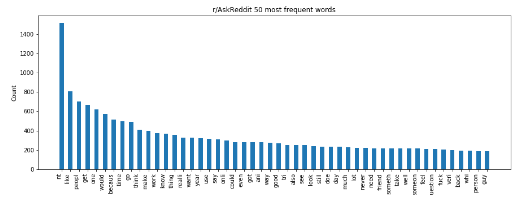

Figure 1. Word popular for a given subreddit

Following that, we employed a TFIDF Vectorizer in order to transform text into vectors. TFIDF is used to find unique words and weight them more relative to common words that would apply in every comment. The weight of a word is given by its Term Frequency and its Inverse Document Frequency. The following equations showcase the weight calculation where t is the term in question and d is the document.

$$TF(t,d) = \frac{Number\ of\ times\ t\ appears\ in\ d}{Total\ number\ of\ t\ in\ d}$$

$$IDF(t) = log\Big(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ t}\Big)$$

$$TFIDF = TF * IDF$$

4 Proposed Approach

Once the data was preprocessed, it was passed through a series of vanilla models, which were validated through a 5-fold cross-validation strategy. The result were results with accuracy ranging from 40% to 50%, depending on the model. In order to improve accuracy, hyperparameters had to be properly tuned.

4.1 Hyperparameter Tuning

In order to find the right set of hyperparameters, we performed a grid search for both the models and the transformer. trying out various permutations for not only the models, but also for the TFIDF Vectorizer. Due to the high number of permutations that the grid search had to go through, it took a long time to tune the hyper parameters. However, the result was a set of tuned models that increased the accuracy from 5% to 10%.

4.2 Regularization Strategies

Since most comments were fairly long, it was very important to regularize the features using Lasso (L1) and Ridge (L2) Regularization. Through our testing, we determined that Ridge regression worked better in almost all situations, the exception being when we employed N Gram for our TFIDF Vectorizer. This will be discussed further in section 5.7.

5 Results

The following table presents the accuracy reported after a 5-fold cross validation. The run time is for a single fold.

Model	Accuracy	Run time
Logistic regression	54.98	25.18 s
Bernoulli NB	50.7	50 s
LinearSVC	56.22	15.79 s
Multinomial NB	56.75	2.43 s
Voting Classifier	56.94	98.63 s
Decision Trees	33.3	1.56 s
Random Forest Classifier	46.80	51.6 min

Table 2: Comparison of model accuracy

5.1 Logistic Regression

Since we already worked on Logistic Regression for assignment 1, we were quite accustomed to its underlying logic, which allowed us to have an understanding of which parameters to tune. In order to fine-tune Logistic Regression, we ran a grid search with either L1 or L2 penalty, as well as the strength of the regularization, C. We also tested various solvers and maximum iteration. Finally, we concluded that the best hyperparameters were L2 for penalty, 4.45 for C and a *liblinear* solver with the default maximum iteration of 100.

5.2 Bernoulli Naive Bayes

We fully implemented a Naïve Bayes model with pipeline integration for testing different feature vectorizer parameters. Additionally we tried several different Laplace smoothing values and word stemming vs lemmatizing with this model. Overall results were good considering the simplicity of the model, with most reasonable parameter combinations giving an accuracy of around 49%. Some fine tuning of our model and vectorizer parameters gave us a best accuracy of 50.7%. Interestingly, stemming generally performed better than lemmatizing with this model, giving accuracies 1-3% higher depending on other relevant parameters. Lower smoothing values were also found to give better results, with 0.2 working best for us. Due to the simplicity of the model, runtimes were short ($\tilde{3}0$ seconds) even with hundreds of thousands of features. Overall this model is too simplistic to achieve a high accuracy on this dataset.

5.3 Multinomial Naive Bayes

The multinomial naive Bayes model we implemented outperformed the Bernoulli naive Bayes model significantly. This is to be expected since the Bernoulli naive Bayes model only accounts for the presence and absence of certain words whereas the multinomial naive Bayes model accounts for multiple appearance of words in a single comment as well as being receptive to feature processing like TFIDF. Our pipeline optimized multinomial naive Bayes model was able to achieve 56% accuracy.

5.4 Decision Tree and Random Forest

Both of these tree-based classification models were integrated into our pipeline and tested with a variety of hyperparameters. In general, tree-based models do not work well for text classification and in that regard our results were in line with what we expected. The decision tree classifier gave a best accuracy of only 33.3% after being run through our testing pipeline. The random forest model gave significantly improved results of 46.8% however this was at the cost of including 1000 trees in our forest which increased the runtime to 1 hour per fold (5 hours total to get the 46.8%). In comparison a pipeline optimized random forest model with the default 10 trees and a runtime more comparable to other models only has a 38.6% accuracy.

5.5 Linear SVC

The objective of a Linear SVC (Support Vector Classifier) is to fit a linear hyperplane that maximizes the margin, i.e. the maximum distance to the nearest points relative to both classes. The dimension of the hyperplane depends on the number of features. Maximizing the MARGIN distances allows more robust classification on future data points. The C parameter controls the tradeoff between a "smooth" decision boundary and classifying training points correctly. A larger C value signifies that more training points are correctly classified. On the other side, a lower C value represents a smoother decision boundary. The "hinge" loss function was used and we opted from a one-vs-all strategy. Hence, each data point is either from this class or not from this class.

5.6 Voting Classifier

A Voting Classifier combines different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.[5] Three Voting Classifiers were built using different combinations of models. The first uses to a combination Logistic Regression second and most accurate model is composed of 3 different classifiers, namely Logistic Regression, Multinomial Naive Bayes and LinearSVC. The latter uses "hard" voting which uses the majority rule to predict. Finally, the third Voting Classifier is composed of only two models: Logistic Regression and Multinomial Naive Bayes. It uses "soft" voting which means that it takes the argmax of the sums of the predicted probabilities. The respectively obtained 55.51%, 56.94% and 56.53% for cross validation accuracy.

5.7 Other preprocessing techniques

During preprocessing, different n-grams combinations were investigated using Multinomial Naive Bayes, Linear SVC and Logistic Regression. We found that without n-grams, the cross validation accuracy was at its highest when Ridge Regression regularization was used. However, when bigrams or trigrams are added to the features, Lasso Regression provides a better cross validation accuracy. Nonetheless, adding bigram and trigram added more noise than information and reduced by cross validation accuracy. Therefore, we opted out the use of n-grams in our preprocessing steps. In addition, we experimented the same models on a stemmed and lemmatized versions of the Reddit comments. In all three cases, stemmed features performed better than lemmatized features. Hence, stemming was used instead of lemmatization during text preprocessing.

5.8 Leaderboard test set accuracy

The highest leaderboard test set accuracy (58.244%) was obtained by using the second Voting Classifier.

Discussion and Conclusion Logistic Regression, Multinomial Naive Bayes and LinearSVC all have decent performance on this Reddit classification task. However, combining all three allows to use a majority vote or the average predicted probabilities (in the case of soft voting) to classify. Hence, this technique is useful to balance out each classifier's individual weaknesses. Since ensemble technique perform particularly well, bagging and boosting ensemble techniques methods should be further investigated, namely Random Forest, AdaBoost and XGBoost.

Although there was a light foray into deep learning models, this could definitely be a direction to take in order to increase the accuracy in text classification. Notably, the usage of a Long Short Term Memory (LSTM) model could be very promising due to its ability to retain past feature.

Another interesting direction would be to employ a form of meta-stacking, where multiple models would be developed to handle different cases. Since the accuracy increased using a voting classifier, it would be reasonable to expect stacking to also deliver positive results.

6 Statement of Contributions

Contributions were split evenly between group members. Ethan implemented the Bernoulli naive Bayes model and ran pipeline optimization for the decision tree and random forest classifiers as well as contributing to the results and related work sections of the report. William ran pipeline optimization for Logistic Regression, TFIDF Vectorizer and attempted a few neural networks unsuccessfully. He also wrote parts or all of sections 1, 3, 4 and 5. Michael implemented model selection and validation pipelines hyperparameter tuned Logistic Regression, Linear SVC and Multinomial Naive Bayes.

References

[1] M. Thangaraj, M. Sivakami, *Classification Techniques: A Literature Review.* 2018, vol 13, pp. 11-1357.

[2] Devlin, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* 2019. [Online]. Available: https://arxiv.org/pdf/1810.04805v2.pdf

[3] Yang, Z. et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding.* 2019. [Online]. Available: https://arxiv.org/pdf/1906.08237v1.pdf

[4] Prateek Joshi. ”How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models”. 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/

[5] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.