Performance Comparison of Text Classification Algorithms

Alok Patel
Student ID : 260954024

Aishwarya Ramamurthy
Student ID : 260963956

Katyayani Prakash
Student ID : 260964511

Abstract

In this project, we investigated the performance of various classifiers on a mutliclass text classification problem. We developed the Bernoulli Naive Baiyes Algorithm from scratch, and compared its performance with five classifiers from SciKit-Learn package, namely Logistic Regression, Neural Network, Decision Trees, Random Forest and Support Vector Machines. Effect of varying the train/test set sizes on a model's accuracy were studied. We also explored different methods of feature construction in text classification and found TF-IDF to yield best generalization results. Lastly, we performed 10-fold cross validation on all models to conclude that Neural Network produces the best accuracy albeit at the cost of higher run-time.

1 Introduction

In this project, we have attempted text classification. We developed models to analyze text from the website Reddit – a popular social media forum where users post and comment on content in different themed communities, known as subreddits. Data was collected from five subreddits, which formed our five classes of classification, and shall be called as such henceforth in this report. These are: 1) Laptop, 2) Anime, 3) Samsung, 4) Science, and 5) Tennis.

The training data contained 1999 observations in all, with 399 observations from class Science, and 400 observations each from remaining classes (Laptop, Anime, Samsung, and Tennis). Our goal was to correctly classify a given post or comment into the subreddit it belongs to.

For training the data, we developed the Bernoulli Naive-Bayes classifier from scratch, and utilized SciKit-Learn's classifiers: 1) Logistic Regression, 2) Neural Network, 3) Decision Trees, 4) Random Forest, and 5) Support Vector Machine (SVM). Experiments on studying the effect of train/test set size in cross validation and different methods for feature construction were performed to arrive at the most optimal model for each classifier. After fitting the training data on each of these models, we performed 10-fold cross validation to select the best model. This model was then applied to test data – a label less collection of 1378 observations – and the consequent output labels were stored in a .csv file and submitted to the Kaggle Competition - Text Analysis.

We found that Neural Network gives us best results, closely followed by SVM and Logistic Regression. We also discovered that TF-IDF is the best method of feature construction in terms of generalization.

2 Dataset

Given the 5-class training data for text classification, Figure 1 visually validates that this dataset is majorly balanced since 4 out of 5 classes consists of 400 examples whereas class 'Science' has 399 instances. Vectorizing the training instances yields a vocabulary size (formally known as Bag of Words (BOW)) of 15365 features wherein 3000 features have been chosen for our experiments. Count Vectorization and Term Frequency-Inverse Document Frequency (TF-IDF) Transformation have been adopted for corpus data preprocessing. The former method has been used to convert the

Submitted as part of coursework for ECSE 551, Winter 2021. Do not distribute.

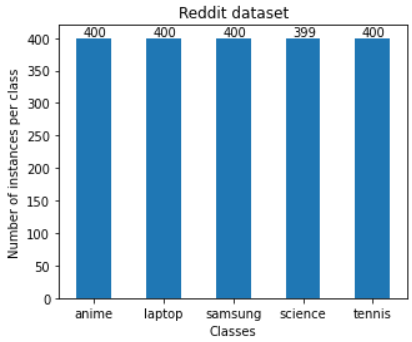

Figure 1: Subreddit Classes and Instances

corpus into a matrix (sparse) with token (word) counts (absolute frequency) with no limits imposed on stop words or maximum features. Term frequency refers to the absolute word frequency normalized to the document length whereas IDF is defined as:

$$IDF(t, Corpus) = log_2\Big(\frac{Number\ of\ Documents\ in\ Corpus}{Number\ of\ Documents\ with\ term\ t\ +1}\Big) \tag{1}$$

TF-IDF being a weighting scheme aims to highlight terms with low frequency (highly informative) and scale down the impact of less informative frequently occurring terms in a corpus of documents. In our experiments, TF-IDF is employed for max_features = 3000.

3 Proposed Approach

The task flow of this project can be briefly categorized into 3 major parts, as delineated below.

3.1 Development of Bernoulli Naive-Bayes Algorithm from scratch

Naïve Bayes is a probabilistic classifier based on Bayes theorem. It works particularly well in text classification problems [3]. For this K-class classification (where K = 5), we developed the Bernoulli Naive-Bayes classifier from scratch by computing the probability of each class, and selecting the class with the highest probability as our predicted label. For this, we used the probability equation as shown below:

$$\begin{aligned} P(y=k|x) \propto \delta_k(x) &= log(P(y=k)P(x|y=k)) \\ &= log(P(y=k)\prod_{j=1}^{m} P(x_j|y=k)) = log(\theta_k \prod_{j=1}^{m} (\theta_{j,k})^{x_j}(1-\theta_{j,k})^{1-x_j}) \end{aligned} \tag{2}$$

where $\theta_k = P(y=k)$ = (# of examples where $y=k$) / (total # of examples); $\theta_{j,k} = P(x_j = 1|y = k)$ = (# of examples where $x_j = 1$ & $y = k$) / (total # of examples where $y = k$).

First, training data was converted into binary vectorized form, and divided according to its class. Next, θ_k and $\theta_{j,k}$ were calculated using the equation (2). We also applied Laplace smoothing to tackle the problem of zero probability (by adding 1 to the numerator and 2 to the denominator in equation (2)).

Lastly, using the predict function, accuracy of the training set was calculated by using the class with highest probability as the predicted label for a particular training comment. We achieved a very competitive accuracy of 90.69% on fitting this model to our training data set.

3.2 Comparing performance of our Naive Bayes model with SciKit-Learn models

Based on our literature review and theoretical knowledge, we decided on 5 models from SciKit-Learn package listed below. We have attempted to fine-tune each model by experimenting with varying train/test set sizes, varying regularization parameter (C) etc. Please see below for a brief summary of each model implementation.

3.2.1 Logistic Regression

Logistic Regression is a simple and powerful linear classifier that uses a weighted combination of input features and passes them through a sigmoid function to sample the output in 2 classes – 0 and 1.

For improving accuracy, we experimented with varying test/train split sizes and regularization parameter, C (smaller C implies stronger regularization). We achieved the best accuracy score for 25% test split, and C in the range of 0.01 to 0.001.

3.2.2 Neural Network - MLP Classifier

MLP stands for Multi-layer Perceptron classifier that relies on an underlying Neural Network to perform the task of classification. MLP utilizes a supervised learning technique called backpropagation for training. Since it is a powerful technique in distinguishing data that is not linearly separable, we intuitively believed it to work better than most other algorithms.

To achieve a better accuracy, we varied the train/test set size and observed that by decreasing test split size from 25% to 10%, accuracy increased from 90.4% to 93%.

3.2.3 Decision Trees

A Decision Tree is a simple representation for classifying examples. It is a Supervised Learning Algorithm where the data is continuously split to arrive at the terminal leaf nodes that predict final class labels. Aliwy & Ameer (2017) [1] stated that decision trees' "capability to learn disjunctive expressions and their robustness to noisy data seem convenient for document classification".

We achieved an accuracy of 74% with 75-25 train-test set size.

3.2.4 Random Forest

Random Forest is an ensemble method of decision trees generated on a randomly split dataset. It overcomes the problem of overfitting encountered in single decision tree. Thus, to see how well a Random Forest performs against a Decision Tree on a real-life multi-class text classification, we implemented this model.

We achieved an accuracy of 88% with 75-25 train-test split which was a definite improvement from the Decision Tree model's performance.

3.2.5 Support Vector Machine (SVM)

SVM is a non-probabilistic linear classifier that maps training examples to points in space so as to place "a maximal margin separating hyper plane" between two classes [8]. SVMs have a key advantage in text-based classification due to their effectiveness in handling high dimensional spaces. Aliwy & Ameer (2017) [1] have called SVM as one of the "fundamentally common text classification methods". Hence, we implemented this model to test its performance on our dataset.

On varying the train/test set sizes, we observed that decreasing the test split yields better accuracy.

Table 1: Effect on SVM model's accuracy by varying test set size

Test Set Size	Accuracy
0.75	0.765
0.25	0.890
0.10	0.910

3.3 Selection of best model and submission to Kaggle Competition

The performances of all above models were tested through a series of experiments and 10-fold cross validation to arrive at the best model, which would then be submitted to the Kaggle Competition. The details of these experiments, along with their results, are summarized in the Results section below.

4 Results

As stated earlier in Section 2, we have employed 2 different methods of feature constructions – Count Vectorization (Bag of Words Approach) and TF-IDF.

For Bag of Words (BOW) approach, we have 2 methods: Binary Representation and Absolute Frequency. By setting the parameter (binary = True) in the CountVectorizer() method, we only fetch the binary representation of a word in a document (1 corresponds to the word being present, and 0 to it being absent). If we set (binary = False), it will yield us the absolute frequency of the word in a document. Below table and graph summarizes the results of our 10-fold cross validation performed over these 3 different training sets.

Table 2: 10-fold cross validation results for different methods of feature construction

Accuracy fetched by each method of Feature Construction			
Model	Absolute Frequency	Binary Representation	TF-IDF
Logistic Regression	0.878568	0.884604	0.916631
Neural Network	0.916604	0.922613	0.918609
Decision Tree	0.744479	0.759163	0.733812
Random Forest	-	0.884600	0.867915
Support Vector Machine	0.843884	0.849893	0.905266

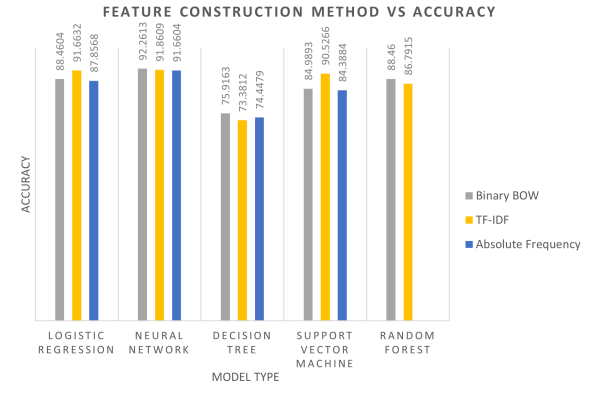

Figure 2: Feature Construction Method vs Accuracy

We also tested our own Naive Bayes model's accuracy with that of SciKit-Learn Naive Bayes, and discovered that our model outperforms the library model in 10-fold cross validation.

Table 3: Comparison of our Bernoulli Naive Bayes model with SciKit Learn's Naive Baiyes model

Model	Accuracy
Our Bernoulli Naive Bayes Model	0.8263
SciKit-Learn's Naive Bayes Model	0.7859

Next, we explored the effect of vocabulary size on classification accuracy. Figure 3 depicts a decreasing trend in classification accuracy as vocabulary size (i.e. number of attributes considered for feature extraction) increases beyond 3000 features. Further, as the feature count (specific to this training corpus) approaches 13000, accuracy attains constancy (75.3%) in the context of the Bernoulli Naive Bayes model developed from scratch.

Finally, we explored the run-time of all our models and compared it to the accuracy fetched by them. Figure 4 provides an overview of time taken to execute code snippets computing accuracy for each text classifier. Bernoulli Naive Bayes model developed by us has the longest runtime of 249.92s

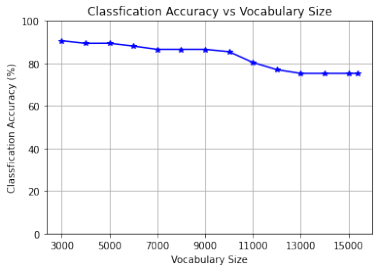

Figure 3: Classification Accuracy vs Vocabulary Size.

proving its high computational cost. This is followed by Neural Networks (NN) at 121.313*s*, Logistic Regression (LR) at 19.342*s*, Support Vector Machine (SVM) at 18.108 *s*, Random Forest (RF) at 9.943 *s* and Decision Tree being computationally fastest at 2.054 *s*.

From the graph, we can see that even though decision tree is the fastest algorithm, it suffers in accuracy. This is largely overcome in Random Forests, which take slightly longer but improve the accuracy. Neural Network is the most time-consuming SciKit-Learn model from our selection, but it brings the best accuracy.

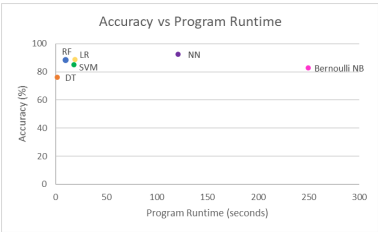

Figure 4: Accuracy vs Program Execution Time for all text classifiers.

From all above experiments, we can conclude that Neural Network is the best model for our dataset. Since it was fetching very similar accuracy with TF-IDF and binary feature representation, we tested both models on test data. We found that TF-IDF fetched us better accuracy. We are getting **92.736%** accuracy on 30% test data in the Kaggle competition.

5 Discussion and Conclusion

After implementing the Bernoulli Naive Bayes and SciKit-Learn models, we found that Neural Network works best for this text classification problem, closely followed by Logistic Regression and Support Vector Machine. We also explored different feature construction techniques such as using the Absolute frequency (word count), Binary Representation of BOW, and TF-IDF. While all methods gave good accuracy on training data, TF-IDF outperformed them all on test data. Therefore, we can conclude that TF-IDF is the best at generalization.

For future investigation, other models such as AdaBoost, QDA, Guassian Naive Bayes, RBF SVM, and K-NN should be tested. Ensemble learning methods such as Bagging and Boosting should also be explored. Lastly, Receiver Operating Characteristic (ROC) can be plotted to better visualise the models' performance.

6 Statement of Contribution

The project was a team effort. Alok dealt with preprocessing of training data, developing the Bernoulli Naive Bayes model and creating ReadMe file. Katyayani explored various methods for feature construction and segregated training instances according to their class label. Aishwarya performed 10-fold cross validation on Naive Bayes model and SciKit models with suitable tuning.

References

[1] Aliwy, A.H., Abdul Ameer, E.H. (2017) Comparative Study of Five Text Classification Algorithms with their Improvements. *International Journal of Applied Engineering Research*. 12(14):4309-4319.

[2] Lecture Slides of ECSE-551: Machine Learning For Engineers by Prof. Narges Armanfard

[3] https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f

[4] https://medium.com/swlh/decision-tree-classification-de64fc4d5aac

[5] https://www.geeksforgeeks.org/decision-tree/#:~:text=Decision%20Tree%20%3A%20Decision%20tree%20is,node)%20holds%20a%20class%20label.

[6] https://machinelearnings.co/text-classification-using-neural-networks-f5cd7b8765c6

[7] R. Lippmann, "An introduction to computing with neural nets," in *IEEE ASSP Magazine*, vol. 4, no. 2, pp. 4-22, Apr 1987, doi: 10.1109/MASSP.1987.1165576.

[8] Xu, Z., Li, P., & Wang, Y. (2012). Text classifier based on an improved SVM decision tree. *Physics Procedia*, 33, 1986-1991.

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

[10] https://getthematic.com/insights/5-text-analytics-approaches/

[11] https://iq.opengenus.org/bernoulli-naive-bayes/