COMP551-A2

Aidan Sullivan 260733921

Devin Kreuzer 260803138

Tal Elbaz 260329068

October 18 2019

Abstract

Text analysis is a ubiquitous and complex problem for modern machine learning researchers. Naives-Bayes has been previously shown to be a simple and effective model for some text analysis tasks. As such, one of the primary objectives of this project was to implement a Multinomial Naives-Bayes (MNB) in order to better understand its ability to classify real-world data. In addition to testing this model, several other models were explored to determine which method(s) would be most effective in the multi-output classification problem at hand. Further investigated methods include: Linear Support Vector Machines (LSV), Logistic Regression (LR), Stochastic Gradient Descent (SGD) as well as multi-model voting schemes. The impact on prediction accuracy for several natural language preprocessing methods were also explored. Amongst the various methods, SGD produced the best individual results, whilst a combination of MNB and SGD produced the best overall prediction accuracy. Our findings ultimately suggest that implementing a combination of models and fine-tuning Laplace smoothing parameters for MNB can be beneficial for multi-output classification problems.

1 Introduction

This project contained a few primary tasks. This included the bare-bone implementation of a Multionomial Naive Bayes (MNB) classifier to run experiments on the dataset and developing a validation pipeline to test these models. The implementation of the Naive Bayes classifier was completed with no preexisting external machine learning libraries.

The dataset which was used for evaluating our model and others were a set of 70000 Reddit comments categorized by their originating subreddit. Further classification methods were compared on the dataset: Linear Support Vector Machine (LSV), Logistic Regression (LR), Stochastic Gradient Descent (SGD) and Multi-model Voting Schemes (ensembles). The impact of several more advanced natural language preprocessing tools from NLTK were also considered which include stemming, lemmatizing, part-of-speech tagging (POS) and chunking amongst others. After all methods were explored, it was determined that SGD yielded the best prediction accuracy on its own, while MNB and SGD classifiers combined in a Voting Classifier produced the best overall results. Our work suggests that Laplace smoothing and multi-model voting schemes have positive impacts on prediction accuracy for multi-output classification problems.

2 Related Work

SGD is the only item not covered in class so far. The idea with this model is gradient descent but with an incorporated probabilistic/random nature to its convergence, which may avoid ocnvergen local minima, and find the global minimum. SDGs are very sensitive to their inputs, and work best on large datasets. These models work best in the context of large scale learning. [1]

In addition to models, various preprocessing techniques for the data were investigated. Notably this included chunking, spelling correction, and natural language processing (NLP). Simple tokenization starts with removing bad characters splitting sentences apart. To clean the text further, text spelling correction can be applied so that later steps can use and interpret the intended word. To make this more effective more natural language techniques can be applied, such as stemming and lemmatization to get the intended root of the word, and part of speech analysis to get the intended context of the word. After these stages are applied, the data can be organized into inputs that can be more meaningful for model training and classification. No particular NLP technique is universally the best, and in some cases training without any NLP preprocessing provides the most effective results. [2]

3 Dataset and Setup

The dataset itself is a set of 70000 reddit comments. Each input was simply an id value and then a comment, and then their correct classification for training reference. Effectively the only raw input was a text field, which would be classified into 1 of 20 groups depending on which subreddit it originated from.

4 Proposed Approach

The Naive Bayes model we implemented takes advantage of some key yet basic concepts. For one, Naive Bayes models make the assumption that there are no correlation between input features. While this is not necessarily true for our case (synonyms and the presence of some words increase the likelihood of seeing others), some preprocessing steps were performed to increase the confidence of this assumption. The approach is relatively simple; fit the model by counting words throughout the comments and use the frequencies of these words across the given categories to assess the individual contribution of each word to the likelihood of it belonging or not belonging to the category. The data structure used to implement this idea was a set of categories whose values are sets themselves; these underlying sets point to key-value pairs of words to occurrence counts across the given category. The algorithm then computes twenty (one for each category) likelihood predictions and returns the one with the largest value. Since the outputs were relatively uniformly distributed, the main driver of prediction score is the log-likelihood of the frequency of the word in the category over the frequency of the word across all other categories:

$$likelihood \propto \sum^{words} \ln(\frac{f_1}{f_2})$$

Where the sum is computed over all words in the comment, f_1 is the frequency of the word in the given category and f_2 is the frequency across all other categories combined. A main issue that jumps out immediately from this formula is the failure of it under the condition that a word has never been seen in the category or has never been seen across any of the other categories. A hyperparameter (α) was implemented to circumvent this issue, which is the reward given to a word for being solely seen in the given category. Words that were never seen in the category or any category were not penalized (better predictions were observed). Preprocessing methods were also heavily explored in the MNB analysi and can be viewed as a form of normalizing and regularizing (removing useless words) the input data.
Beyond our own algorithm, we decided to explore SVR, LR, SGD and MNB from Scikit-learn with various tuning parameters. Furthermore, we combined the best-resulting models into a Voting system thanks to Scikit-learn's Voting Classifier model. Grid searches were performed to identify the most effective parameters and weights.

5 Results

Manual grid searching was performed to determine the most effective parameter for our Naive Bayes model. Figure 1 gives an insight behind our parameter selection of $\alpha = 5$. It was subsequently confirmed that stemming the input (55.65%) produced better accuracy than lemmatizing (54.6%), whilst keeping only nouns, verbs and adjectives for Part of Speech tagging even further decreased the predictive capacity of the model (53.2%).

Further model testing produced similar results for LSV and LR. However, Scikit-learn's MNB and SGD (with grid-searched optimized hyperparameters) produced stronger results. Separating the given sample data into train and test samples, we achieved the following optimized 5-fold training set validation scores for the individual models:

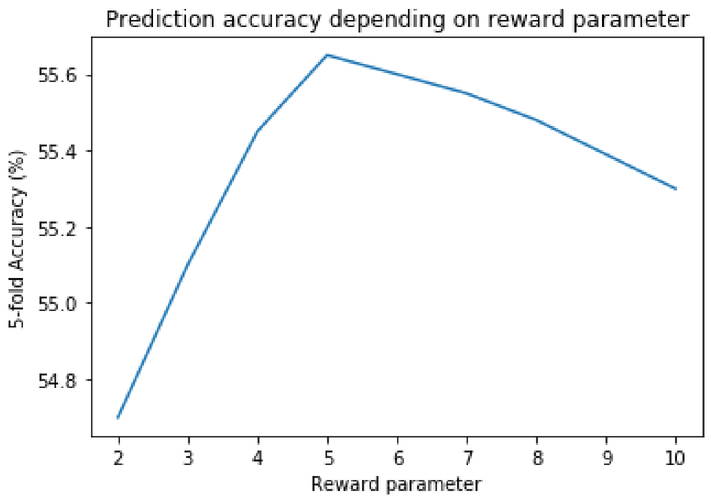

Figure 1: Plot of accuracy vs reward parameter

Table 1: 5-Fold Accuracies for each Model

Model	Accuracy (%)
Scikit LSV	55.37
Scikit LR	55.20
Scikit SGD	57.61
Scikit MNB	57.46
Our MNB	56.33 (Test set)

For the Kaggle submission, the final model that was used was a voting classifier. This consists of different models “voting” on what label has the highest probability of being the correct one. In addition, different models can be given weights based on how important the user believes them to be. For this problem, we tried many different combinations of models and respective weights, and the best K-fold accuracy was achieved by using only two types of models; the SGD and the MNB. This makes sense, as they are substantially better than the other models. The ideal configuration consisted of multiple versions of these models, with different alpha values. Alpha values in the MNB models correspond to an additive Laplace smoothing parameter, while being a constant that multiplies the regularization term in the SGD models. We used two MNB, with alphas 0.22 and 0.15, and three SGD, with alphas 0.00012, 0.00025, and 0.00032. In addition, the more accurate models were given three times larger weights than the others (one MNB and one SGD). This proved to be the most effective configuration, which enabled us to get a Kaggle accuracy

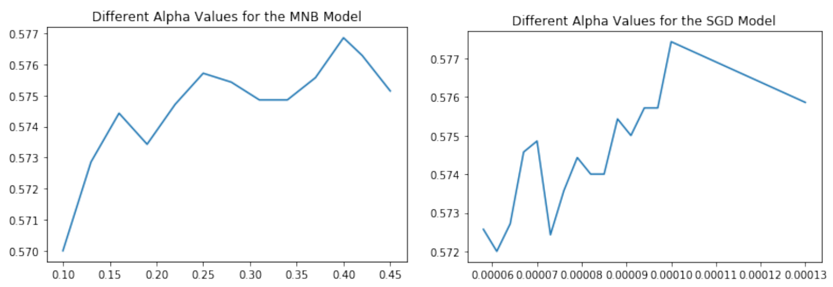

Figure 2: Different alpha values for both the SGD and MNB models

of above 59%. In Figure 2, we can see a plot of how the accuracy of the models change depending on their alpha value.

6 Discussion and conclusion

In conclusion, we learned a great deal about the Naive Bayes algorithm by implementing and testing it on a real world dataset. We observed the importance of fine-tuning Laplace parameters for optimizing prediction accuracy, as well as the impact that natural language preprocessing tools have on it. We also discovered a very powerful Voting Classifier tool, which drastically improved our results. This has piqued our interest on classification problems, and we look forward to learning about more powerful algorithms in the deep learning and neural network fields.

7 Statement of Contributions

Devin implemented the Naive Bayes script and investigated preprocessing method. Aidan did extensive research on Natural Language processing tools. Tal experimented with the Scikit-learn models and achieved our best prediction accuracy. All authors contributed equally to the write-up.

References

[1] F. Kabir et al. “Bangla text document categorization using Stochastic Gradient Descent (SGD) classifier”. In: *2015 International Conference on Cognitive Computing and Information Processing(CCIP)*. Mar. 2015, pp. 1–4. DOI: `10.1109/CCIP.2015.7100687`.

[2] Alper Kursat Uysal and Serkan Gunal. “The impact of preprocessing on text classification”. In: *Information Processing Management* 50.1 (2014), pp. 104–112. ISSN: 0306-4573. DOI: `https://doi.org/10.1016/j.ipm.2013.08.006`. URL: `http://www.sciencedirect.com/science/article/pii/S0306457313000964`.