# Analysis of Textual Data Classification with a Reddit Comments Dataset

**Adam Babs**
School of Computer Science
McGill University
Montreal, QC H3A 0G4

**Abstract**

Sentiment analysis became an area of intense research as it may be found very useful in various areas and may provide valuable information. Text classification is the problem of labeling natural language texts into a set of predefined classes.

The main objective of this project is to categorize comments from the American social news aggregation, web content rating, and discussion website – Reddit.

Various methods have been applied to develop the training algorithms and improve their accuracy, I experimented with different classification algorithms, ensemble methods, numerous text preprocessing approaches, various feature extraction methods. My experiments demonstrate how combining the aforementioned methods allowed me to train my final model with accuracy of 57.455%, which is approximately 5.5 percentage points increase in comparison to where I began.

## 1 Introduction

Reddit community does not have rules specifying how comments in particular categories should be composed and what type of vocabulary or structures they should comprise of. It is what makes the task of labeling comments a complex problem - successful predictions mostly depend on the words used in given categories and words used in various categories may be very similar.

My main goal was to achieve the highest possible accuracy of predictors on the Reddit comments. While developing the model which achieved the highest prediction performance I experimented with numerous combinations of feature extractions and preprocessing methods which I applied to learn algorithms. I applied some basic text data preprocessing methods such as lemmatizing, stemming, removing; stopwords, punctuation, numbers, characters outside of the English alphabet, converting the text to lowercase. At this point, I could observe a relatively high increase in the accuracy of algorithms.

Subsequently, I experimented with feature extraction and compared the accuracy of algorithms using the Count Vectorization and computing Term Frequency — Inverse Document Frequency. I tried training models including and excluding different features based on their frequency of occurrence and using different numbers of features that were included in the learning process.

Logistic Regression, Stochastic Gradient Descent Classifier, Support Vector Classifier, Decision Trees, Multinomial Naïve Bayes and Bernoulli Naïve Bayes algorithms were used to perform comments classification. To increase the prediction performance of classifiers, I applied the Grid Search and Random Search methods to find hyperparameters which provide the highest accuracy.

Ultimately, the highest accuracy was achieved as a result of using the Ensemble Methods of stacking three algorithms and using hard voting. These three algorithms were: Multinomial Naïve Bayes, Stochastic Gradient Descent Classifier and Logistic Regression.

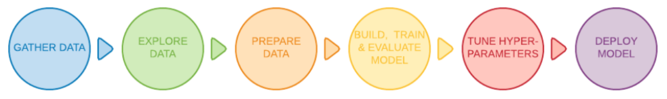

*Diagram 1. Workflow I applied to solve the labelling problem*
*Source: Google Developers Machine Learning Guide*

## 2 Related work

As stated before document classification became an area of intense research and found its' application in numerous fields. Tagging the content posted on the internet, analyzing customer reviews, spam filtering or extracting information from textual data for marketing purposes are only a few of them. Many scientific papers indicate how crucial and useful sentiment analysis became and authors discuss how significant in text labeling problems, are appropriate feature extractions and accurate data preprocessing.

In recent years, document classification has become important due to the advent of large amounts of data in digital form. For several decades now, document classification in the form of text classification systems have been widely implemented in numerous applications (Dino Isa et al., 2008).
The representation of a problem has a strong impact on the generalization accuracy of a learning system. For the problem of text categorization of a document, which is typically a string of characters, has to be transformed into a representation, which is suitable for the learning algorithm and the classification task (Li-Ping Jing et al., 2003).

Isabelle Guyon and Andre Elisseeff discuss other profits of appropriate feature selection and how it influences the accuracy of prediction algorithms. There are many potential benefits of variable and feature selection: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance (Isabelle Guyon and Andre Elisseeff, 2003).

Daniel Morariu, Radu Cretulescu, Lucian Vintan discuss how implementing an ensemble method based on the Support Vector Machine techniques and Naïve Bayes Theory may improve the prediction performance of predictors. They also describe the stacking method as a strategy that can improve the accuracy on the text classification problem. We combine multiple classifiers hoping that the classification accuracy can be improved without a significant increase in response time. Instead of building only one highly accurate specialized classifier with much time and effort, we build and combine several simpler classifiers. A usually approach is to build individual classifiers and later combine their judgments to make the final decision (D. Morariu, R. Cretulescu, L. Vintan, 2010).

## 3 Dataset and setup

The data used in the project is a file made up of Reddit comments that had been posted on the platform. The train dataset on which I conducted experiments and trained models comprises of 69 999 comments and their labels. The test dataset on which I tested the model and reached the accuracy of 57.455% in the competition comprises of 29 999 samples without labels.
Initially, I trained models before preprocessing the textual data and gathered the results. Subsequently, I started with preprocessing and experimented with various methods and different setups of lemmatizing, stemming, deleting words containing less than three characters, removing digits, numbers and all characters that are not in the English alphabet. I removed all hyperlinks, stopwords, punctuation and turned every character into a lower case.
Various combinations resulted in different accuracies, ultimately the approach that gave us the highest accuracy was: lemmatizing, turning every character to lower case, removing hyperlinks, stopwords, and punctuation. After applying the approach that gave us the highest accuracy, I deleted one sample (number 58011) - it contained an empty string.

| Category | Number of samples |
|---|---|
| baseball | 3500 |
| canada | 3500 |
| gameofthrones | 3500 |
| wow | 3500 |
| Music | 3500 |
| anime | 3500 |
| europe | 3500 |
| worldnews | 3500 |
| GlobalOffensive | 3500 |
| hockey | 3500 |
| nfl | 3500 |
| AskReddit | 3500 |
| nba | 3500 |
| soccer | 3500 |
| funny | 3500 |
| trees | 3500 |
| Overwatch | 3500 |
| conspiracy | 3500 |
| leagueoflegends | 3500 |
| movies | 3499 |

*Table 1. Categories and number of comments in each category*

## 4 Proposed approach

Initially, I trained models without preprocessing the data, I just applied the Tf-Idf weighting and Count Vectorization. Tables below present how these two techniques influence the classifying accuracy.

| Classifier | Accuracy |
|---|---|
| Multinomial Naïve Bayes | 55.7265% |
| Logistic Regression | 54.3222% |
| Stochastic Gradient Descent | 55.72507% |
| Support Vector Classifier | 55.31507% |

*Table 2. Accuracy before data preprocessing, Tf-Idf Weighting*

| Classifier | Accuracy |
|---|---|
| Multinomial Naïve Bayes | 53.07789% |
| Logistic Regression | 52.50789% |
| Stochastic Gradient Descent | 45.39635% |
| Support Vector Classifier | 49.02784% |

*Table 3. Accuracy before data preprocessing, Count Vectorization*

The difference in prediction performance is relatively high, thus I chose to use Tf-Idf weighting in further experiments. This indicates that rare words contribute more weights to the models I implemented, which is reasonable as comments in the same section refer to the mutual topics and similar vocabulary is used.

As I conducted the experiments with different preprocessing methods I found out that lemmatization, turning everything character to lower case and removing: punctuation, hyperlinks, stop words and removing samples that were empty after preprocessing (sample number 58011), yields the best predictions, hence I decided to use this approach while constructing the final model.

| Classifier | Accuracy |
|---|---|
| Multinomial Naïve Bayes | 56.07222% |
| Logistic Regression | 54.64506% |
| Stochastic Gradient Descent | 55.08364% |
| Support Vector Classifier | 54.74792% |

*Table 4. Accuracy after data preprocessing, Tf-Idf Weighting*

| Classifier | Accuracy |
|---|---|
| Multinomial Naïve Bayes | 54.44792% |
| Logistic Regression | 52.38788% |
| Stochastic Gradient Descent | 50.78357% |
| Support Vector Classifier | 49.28641% |

*Table 5. Accuracy after data preprocessing, Count Vectorization*

Contrary to my expectations, data preprocessing increased the prediction performance of only two models - Naive Bayes Algorithm and Logistic Regression while decreasing the accuracy of two others. This was a valuable finding and it helped me improve the final accuracy of my model.

To train my final model I decided to use the Stacking Ensemble Method. Results show that stacking has been proven to be consistently effective over all domains, working better than majority voting and that using the opinion summary can improve the performance further (Ying Su et al., 2012). I combined the performance of the Multinomial Naïve Bayes, Stochastic Gradient Descent Classifier and Logistic Regression.

To improve the prediction performance of the model, I tuned hyperparameters and experimented with changing the voting method and added the Support Vector Classifier. Ultimately, the majority voting method resulted in the highest accuracy and adding another base model, the Support Vector Classifier did not increase the successful predictions rate.

As I discovered earlier, the Stochastic Gradient Descent yields a higher accuracy on a dataset before preprocessing, hence I used this finding in modeling the stacked classifier.

Another method that I experimented with was implementing the Random Forest Classifier. To optimize its' performance I tuned hyperparameters using the Grid Search method, although the achieved accuracy of the algorithm was not satisfying.

## 5 Results

When constructing the final model with the highest accuracy, I built three individual classifiers and combined their performance. I stacked Multinomial Naive Bayes, Support Vector Classification and Stochastic Gradient Descent and merged their judgments to make the final decision based on hard voting. Base models in the final classifier were fitted on different datasets. Stochastic Gradient Descent and the final model were fitted on the text before preprocessing, Multinomial Naive Bayes and Logistic Regression were fitted on preprocessed comments. The algorithm used for Kaggle submission was fitted on the preprocessed training. It resulted in a performance presented below:

**Stacked Classifier Performance:**
Average accuracy from 5-fold cross validation on the training dataset: **57.4679%**
Accuracy on Kaggle (preprocessed data): **57.455%**
Accuracy on Kaggle (data before preprocessing): **53.677%**

I applied various techniques, tried numerous models and different combinations of stacking these models. My final model had the highest accuracy that I managed to achieve.

I experimented with:

- stemming,
- n-grams,
- removing all strings of length less than two,
- removing all digits and numbers,
- tuning max_df and min_df hyperparameters[1]
- tuning the max_features hyperparameter[2]

[1]Maximum and minimum document frequency across the corpus, which ignore terms that have a document frequency higher or lower than the given threshold,

[2]Maximum features hyperparameter which builds a vocabulary that only consider the top given features ordered by term frequency across the corpus.

All of the above techniques failed to improve the prediction accuracy of implemented classifiers.

To develop the Random Forest Classifier performance, I used the grid search method to find the best parameters. To expand the search, I manipulated the number of features included in the training process, however, I did not notice any improvements.

Parameters tuned to develop the classifier:

- number of estimators
- maximum depth of the tree
- minimum number of samples required to split an internal node
- minimum number of samples required to be at a leaf node

**Random Forest Classifier Performance:**
Accuracy: **44.29285%**
Best parameters: max_depth= 30, min_samples_leaf=1, min_samples_split=10, n_estimators= 1200

## 6 Discussion and Conclusion

Labeling textual data coming from twenty different categories, where the majority of samples comprise of similar language written using ordinary, everyday language is a demanding task and even tiny improvements in prediction performance result in a considerable increase in the number of correctly classified samples when working on big datasets.
Considering the fact that I started with an accuracy oscillating between 45% and 52% (depending on the model used) and improved the final accuracy of the predictor to around 55% only as a result of feature extraction and text preprocessing, I managed to correctly classify a few thousand more comments. *When predicting the labels on a dataset containing 100 000 samples. It proves that even a small increase in accuracy is meaningful.

The main conclusion drawn from this task is that experimenting with preprocessing, feature extraction, learning various models and tuning their hyperparameters significantly contribute to the improvement of predicting performance and makes room for continuous development. Despite the fact that I found some of the discovered results rather unexpected, they were valuable and I built my final model upon them. I also observed how effective combining the performance of several algorithms may be.

## References

[1] Isabelle Guyon and Andre Elisseeff. *An Introduction to Variable and Feature Selection*, 2003.

[2] Dino Isa, Lam H. Lee, V.P. Kallimani and R. RajKumar *Text Document Preprocessing with the Bayes Formula for Classification Using the Support Vector Machine*, 2008.

[3] Google Developers Machine Learning Guide on Text Classification – workflow diagram `https://developers.google.com/machine-learning/guides/text-classification`

[4] D. Morariu, R. Cretulescu and L. Vintan. *Improving a SVM Meta-classifier for Text Documents by using Naive Bayes*, 2003.

[5] Ying Su, Yong Zhang, Donghong Ji, Yibing Wang and Hongmiao Wu. *Ensemble Learning for Sentiment Classification*, 2012.

[6] i-Ping Jing, Hou-Kuan Huang and Hong-Bo Shi. *Improved feature selection approach TFIDF in text mining*, 2002.