COMP 551 Project 2: Reddit Classification

Daniel Chernis	260707258
Archit Gupta	260720757
Chenthuran Sivanandan	260749843

Abstract

The goal of this project was to build a model to classify reddit comments into 1 of 20 subreddit categories. After testing Naive Bayes, Logistic Regression, Decision Trees, Support Vector Machines and Multinomial Naive Bayes, the latter proved to be the best model to predict unclassified reddit comments. Some of the pre-processing techniques we utilised were using the TF*IDF vectorizer and removing stop-words. We also implemented bagging (ensemble learning) which boosted our final accuracy to 57.64 % on the test data-set. All models attained an accuracy of at least 50 % with the exception of decision trees, which tend to overfit and could only reach a maximum accuracy of 35 %.

Introduction

In this mini-project we developed models to analyze text from Reddit. The goal of this project was to develop a supervised classification model that can predict what community a comment came from.
We implemented Naive Bayes Classification from scratch reaching an accuracy of 50.23%. Using SciKit Learn we modelled Logistic Regression (LR), Decision Trees (DT), Support Vector Machines (SVM) and Multinomial Naive Bayes (MNB). We reached an accuracy of 57.644 % for MNB (with an alpha of 0.2). The data set contained 20 communities and had 70000 comments.

Related Work

We read a paper by Jee Ian Tam [3] who classified reddit comments based on their subreddit. In the paper they investigated the results given by Recurrent Convolutional Neural Network (CNN), and Attention-Pooled CNN. According to their results Recurrent CNN with GRU performs the best.
We also read another paper by Bo Pang et al. [4] on sentiment classification using Naive Bayes. This helped us understand the performances of multiple models on a slightly different classification problem.

Dataset and Setup

There were 20 different subreddits. Words were derived from the data through splitting on spaces and new-line characters. Every word was stripped of punctuation and cast to lowercase. This was inspired by the cleaning methods in [1]. Additionally, any word less than 3 characters

or containing ‘http’ (a potential url word) was removed since it doesn’t add much value for classification. Finally the training data were shuffled and partitioned 90-10 for training and validation data respectively.

Proposed approach

On Preprocessing

Initially testing with a 50-50 split data for training and validation, we noticed having more training data increased our accuracies. To avoid overfitting, 90% of the data was used for training and the rest for validation. These numbers were decided upon as we had so much data so using only 10 percent of it for testing would still enable us to have an accurate insight as to how well our model was performing.

Since raw reddit comments might have typos in them, the data needed to be stripped of punctuation and uppercase characters converted to lowercase to increase the chance of the models recognising the same words even if they were initially differently written. Since urls and malformed words (having less than 3 characters) cannot be easily mapped to a corresponding class, those were completely ignored

Naive Bayes Model

For fitting we computed the probability of each word (feature) appearing in a given subreddit, the probability of each word appearing in the data-set, and the probability of a certain subreddit appearing. We applied Laplace smoothing to all three cases, to avoid probabilities of zero.

For prediction on the validation data, for every comment, the probability was computed the following way using Bayes Rule:
*P(class = subreddit | all words) = Max[P(all words_i | class = subreddit) * P(subreddit)]*
Where i = some comment for class = subreddit

The value of the probability of a word in validation data-set given some subreddit depends on cases:

1. If the word w from validation comment was found in our set of words from the training, then the probability of w is that from the training data set.
2. If the word w in validation comment was NOT found, then the probability of w is 1/ number of appearances of that word for that subreddit in question in the whole training data-set
3. If a word was not found in the validation comment, then the probability of that word is the complement of the probability of that word being in the training data set for that subreddit.

Multinomial Naive Bayes

We used the TF*IDF vectorizer to convert the words to numbers. TF*IDF captures the rarity of each of the words, enabling the model to associate the subreddits with very specific words. We also removed stop words as they appear across all topics deeming them redundant.
MNB works for this kind of classification task since there are 20 possible classes a test example can be attributed to. We used bagging to improve the accuracy of our model. By running our MNB model 25 times, we get 25 sets of predictions. We then used a voting mechanism to decrease variance and get the final set of predictions. We also made use of a Laplace smoothing parameter of 0.2. This was done to increase the zero probability values to a small positive number (and correspondingly reduce other values so that the sum is still 1).

Results

Performance of MNB - Our MNB model is the model resulting in the highest accuracy for the Kaggle competition (57.64 %, alpha=0.2). We implemented bagging (25 runs), wherein the model runs on a shuffled dataset split 90-10 between training and validation. We get 25 sets of predictions, after which we use a voting mechanism to form the final prediction set. We changed values of the hyperparameter alpha (smoothing parameter), the results of which are summarised below.

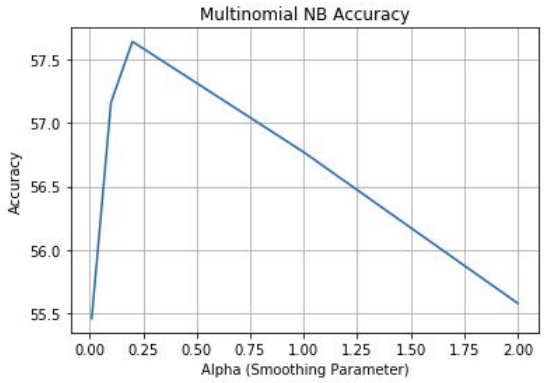

Figure 1 - Graph depicting the MNB model accuracy with respect to the smoothing parameter

Performance of SVM - The hyperparameter we changed to get different models for SVM was the tolerance, which is essentially the stopping criteria. We used 5 fold cross validation, the results of which are summarised below. A tolerance between 0.1 and 1 works best for the SVM model.

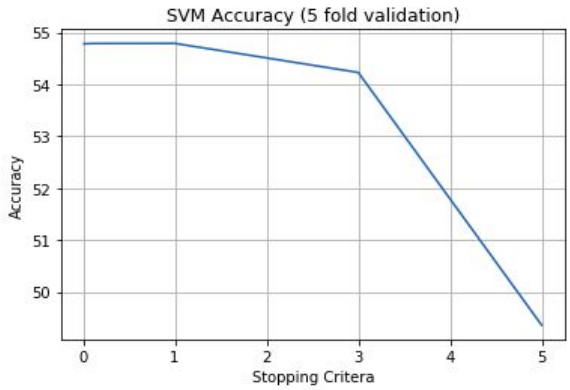

Figure 2 - Graph depicting the SVM model accuracy with respect to the Alpha parameter

Performance of Decision Trees - The hyperparameter we changed to get different models for the Decision Tree was the maximum depth, which is the maximum depth that we want our tree to be. We used a held out validation set to test our results. It seems that as the maximum depth increases we get a better value for the accuracy on the validation set.

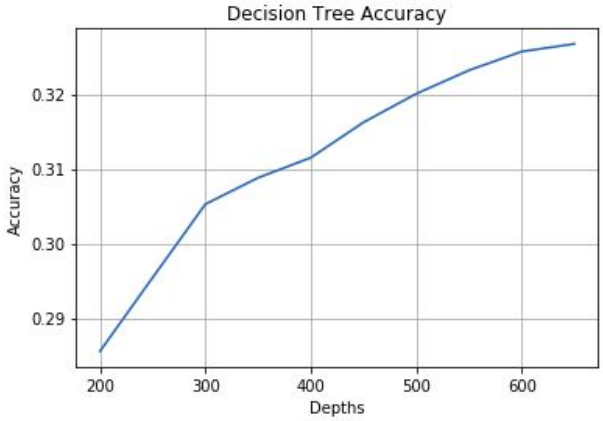

Figure 3 - Graph depicting the Decision Tree model accuracy with respect to the Depth of the tree

Performance of Logistic Regression - The hyperparameter we changed to get different models for LR was the tolerance, which is essentially the stopping criteria. We used a validation test set of 25 %. A tolerance of 0.001 gives the best accuracy.

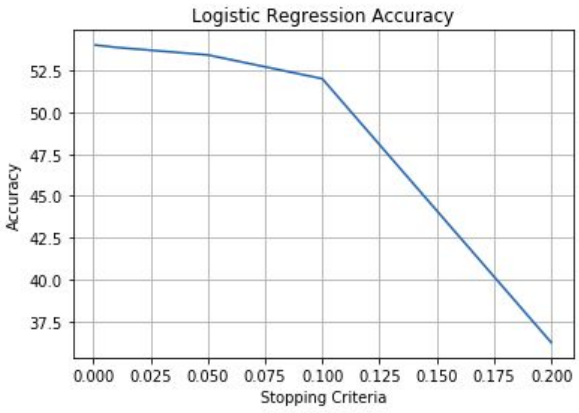

Figure 4 - Graph depicting the Logistic Regression model accuracy with respect to the Stopping Criteria

Discussion and Conclusion

Throughout this project, we explored different models and the effects of feature engineering on a model's performance. We learned more about what approaches to take to solve a problem pertaining to natural language processing. We also learned more about which of the current models that we've learned about perform well on text classification tasks. In the future we would likely utilize deep learning. We can also test different methods to add new features (combining different features) or to filter out less relevant features (chi-squared test).

Statement of Contributions

Daniel did the preprocessing and decision tree model. Daniel and Chenthuran worked on naive bayes. Chenthuran worked on the logistic regression and generation data results for decision tree model. Archit worked on the SVM and Multinomial NB.
The report was written collaboratively.

References

[1] Jason Brownlee -
https://machinelearningmastery.com/clean-text-machine-learning-python/?fbclid=IwAR12HjCGSboNmPjz8439KTdNhIMKRwuIEfugeWY3wrDjSALddUmYBdq_pE0

[2] William L. Hamilton -
https://www.cs.mcgill.ca/~wlh/comp551/slides/05-linear_classification_cont.pdf

[3] Jee Ian Tam
https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2735436.pdf.

[4] Bo Pang et al.
https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf