COMP 551 Mini Project 2

Ian Benlolo (260744397), Anthony Maalouly (260532621), Christos Panaritis (260739936)

***Abstract*— The aim of this paper is to create a model to determine from which subreddit a comment from reddit most likely came from. There are twenty subreddits considered, with seventy thousand comments in total. Multinomial Naive Bayes, Complement Naive Bayes, Logistic Regression, Support Vector Machines, K-Nearest Neighbours, Decision Tree, and Ensemble method, were considered from the SciKit-Learn libraries. Four features were used with those models : preprocessing, VADER sentiment, TF-IDF, and Bi-gram inclusion. It was determined that the best model is the Ensemble of Multinomial Naive Bayes, Complement Naive Bayes and Logistic Regression which predicted 59.06 % with the exclusion of every feature except TF-IDF, converting to lowercase and removing stop words which was validated on the Kaggle Competition.**

I. Introduction

Reddit is an online forum colloquially known as the "front page" of the internet. The forum is divided into smaller forums called subreddits, each specializing in specific topic. For example, popular subreddits like /r/funny have upwards of 20 million subscribers. The dataset is composed of three headers: an arbitrary ID number, the comment, and the subreddit from which the comment originated. The task is to create a model to predict from which subreddit a comment most likely came from.

Bayes theorem is quite a basic theorem in probability. It is used to determine the probability of an event occuring using prior knowledge. It goes as follows

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where $P(A|B)$ is a conditional probability: the likelihood of event A occurring given that B is true and $P(A)$ is simply the probability of A occurring. In the Naive Bayes (NB) algorithm there is an assumption that all predictors are independent of each other. In other words, $P(x_j|y) = P(x_j|y, x_i)$ for any i, j. This assumption is useful (but usually "naive") as it brings the number of ways to describe the model from $O(2^m)$ to $O(m)$.

Generalizing Bayes theorem,

$$P\left(y|x_1, \ldots, x_n\right) = \frac{P(y)P\left(x_1, \ldots x_n|y\right)}{P\left(x_1, \ldots, x_n\right)}$$

which using our assumption can be seen as

$$P\left(y|x_1, \ldots, x_n\right) = \frac{P(y)\prod_{i=1}^{n} P\left(x_i|y\right)}{P\left(x_1, \ldots, x_n\right)}$$

When classifying, we simply take the class which maximizes the probability of that value, given all the inputs, ie $\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P\left(x_i|y\right)$.

Some errors can appear when this algorithm comes across predictors it hasn't seen before therefore a smoothing factor can be included.

To perform this classification task, multiple models were considered. First, the Bernoulli Naive Bayes model with a Laplace smoothing was implemented from scratch. Also, Logistic regression, Multinomial Naive Bayes, Complement Naive Bayes, Logistic Regression, Support Vector Machines, K-Nearest Neighbours (KNN), Decision Tree, and Ensemble method were called using the readily available packages from Sci-Kit Learn.

Since the data consisted of comments from the internet, naturally these had to be cleaned before any feature extraction was looking into. See Section V for more on preprocessing.

To evaluate the performance of the models, a pipeline using a 5-fold cross validation was implemented. Lastly, the final results presented in this report come from Kaggle using a held out test set.

Term Frequency-Inverse Document Frequency (TF-IDF) is the product of the number of occurrences of a word multiplied by the logarithm of the number of documents in a corpus over the number of documents containing the word [4]. Mathematically, it is described as

$$\text{TF-IDF}(t, Corpus) = t * log\frac{\text{\#of Docs in Corpus}}{\text{\#of Doc with term t +1}}$$

The TF-IDF is a metric which gives higher values for words that occurs only in specific documents, and appear more frequently. In the context of comment classification, it is expected for the word "hockey" to

appear more often in /r/hockey than in /r/nba, and a larger TF-IDF value for that word will most likely indicate that this is an /r/hockey-related comment.

II. Ethical Concern

Although many positive applications of text classification technology may be considered such as automatic tagging of content and products in websites, topic labeling, sentiment analysis, ML researchers must ensure that their work cannot be in a negative manner. Reddit is an online forum, where anonymity of its users reigns supreme. In some way, this forum allows user to express their true feeling, and without filter since the reddit user name is not directly linked to the identity of a person. Hence, it wouldn't be too far-fetched to think that based on reddit comments and social data mining, a model could be created to link a persons identity to a reddit user name. Such a model could have important real life consequences especially in subreddits such as /r/confessions, where redditors have confessed their wrong doings under a presumed veil of anonymity.

III. Related work

Sentiment analysis is the task of quantifying how positive, negative, or neutral a text is. It is a problem that is of interest for studies in social media, marketing, advertising, and psychology, to say the least. Hence, lexicons such as Linguistic Inquiry and Word Count (LIWC) were developed. LIWC was used to identify language style information. Using two central features, a preproessing component, and a dictionary. LIWC is a transparent text analysis program that counts words in psychologically meaningful categories, meaning that it recognizes words such as "See", "Hear", "Feel" to be associated with perceptual senses, and words such as "Anxiety", "Anger", "Sadness" to be associated with negative emotion.[3]. Pennebaker et. al concluded that language used is a key indicator of the social psychologies.

In 2014, VADER, a parsimonious rule based model for sentiment analysis was developed (Gilbert & Hutto, 2014, [2]). This model outperformed much of the current state of the art, and even scored better than individual human raters. A *gold-standard* list was constructed along with five general rules, namely punctuation, capitalization, degree modifiers, consideration of contrastive conjunction "but", tri-gram examination preceding a sentiment-laden lexical feature. VADER performed as well and in many cases better than eleven state-of-the-art sentiment analysis tools, such as SentiWordNet, ANEW, LIWC, to name a few.

Uysal and Ginnal ([1]) talk about having the right combinations of preprocessing is essential for any good text classification problem, though highlight that there are no 'best' combination for every classification task.

IV. Dataset and setup

Since the data come from an online forum, naturally we needed to clean it and remove any "junk". A few preprocessing steps were taken before parsing the comments: punctuation removal, lemmatization, tokenization, lower-casing all the comments, removal of stop words, and removal of extra spaces (see Section V for more). Afterwards, Scipy's CountVectorizer method was used to parse the comments, and create the vocabulary in a sparse matrix format due to its large size. This constituted as the input feature vector. The classifications were transformed to (arbitrary) numbers with the inverse transform saved for later use when predicting.

V. Preprocessing

Since this text was taken from the online forum Reddit, many steps had to be taken in order to "clean" it for better learning. Some common prepossessing such as removing stop words, putting everything in lowercase were used in all runs and were generally successful in improving the accuracy of our models. We first implemented a very rigorous cleaning which consisted of what was mentioned above as well as removal of any digits, punctuation, lemmatizing (standardizing different forms of a word) and stemming (reducing words to their stem) every word and we replaced all hyperlinks with the string 'https'. These were used as they seem to be standard practice in text classification [1] though some of these seemed to yield worse results (which will be discussed later).

These were all implemented using a python package called NLTK (Natural Language ToolKit) and some regular expressions to delete punctuation, digits and replace hyperlinks.

Another part of our preprocessing consisted of some feature extraction using a package called vaderSentiment SentimentIntensityAnalyzer. Valence Aware Dictionary for sentiment Reasoning (VADER) is a rule based model which extracts a 'sentiment' from the comments. As described in Gilbert & Hutto 2014, VADER is a sentiment analyzer which outperformed eleven highly regarded state of the art sentiment analyzer tools.[2]

VI. Proposed Approach(es)

The first model that was implemented from scratch and tested was Bernoulli Naive Bayes(NB). Since the current project is a text classification problem, adding Laplace smoothing was also added in order to add Bayesian prior to the model. This helps improve results because it accounts for words that do not appear in the training set.

During testing, four features were created and tested. As described in section IV, a vocabulary of the words in the dataset was created with both a count vectorizer and the TF-IDF method. Both methods were tested with bi-grams as well. Finally, a sentiment feature for each comment was tested as an additional feature.

Many models were compared to see which would perform best on the dataset. Three Naive Bayes implementations (Bernoulli, multinomial and complement), logistic regression, support vector machine (SVM),KNN and decisions trees were tested against each other. Five-fold cross validation was chosen as to measure accuracy because it gave realistic predictions on the performance of the model on the leader board and consists of a decent amount of training/testing data at every iteration.

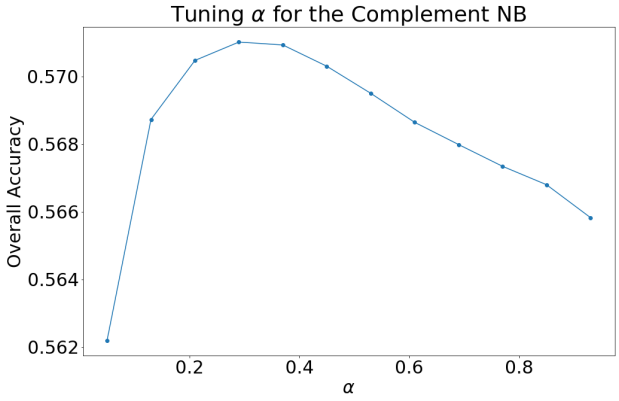

Fig. 1: Tuning the α parameter for Multinomial NB. This was done with only basic preprocessing as defined in Section V. The value that worked best with the ensemble method seemed to be at around 0.22.

Based on the results, an ensemble model will be created with the best performing models in order to get more accurate and stable predictions [5]. In the end, an ensemble was made with logistic regression, multinomial naive Bayes and complement naive Bayes. Each of the models were modified with the best hyperparameters by using grid search on a subset of them to find optimal values.

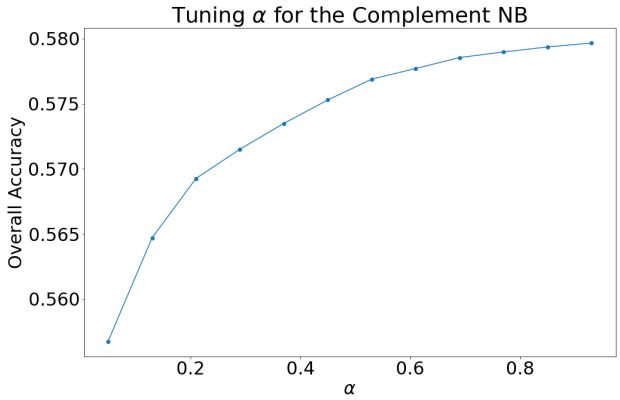

Fig. 2: Tuning the α parameter for Complement NB. This was done with only basic preprocessing as defined in Section V. The value that worked best with the ensemble method did not seem to follow this distribution and was observed to be quite small.

VII. Results

In order to determine the best features and the best models, it was best to run feature tests on all the models. From there, it would be easy to determine the best feature set and models to use for the ensemble model. Important note is that Bernoulli naive Bayes (BNB) was not considered as a model since losing the word count caused a loss of accuracy in every other model. Multinomial naive Bayes (MNB) was used instead since it consistently outperforms BNB for most text classification problems [6] despite BNB having an accuracy of 56.3485%. However, the runtime of the implemented BNB was magnitudes slower than that of SkLearn (323 seconds) as shown in Table I. This was due to the training function using for loops instead of vectorized calculations.

As shown in Figure 3, the best model is the Ensemble method which is composed of the MNB, CNB, and logistic regression, with a final accuracy of 58.96% using our pipeline, while on the Kaggle leaderboard using a held-out test set, the model had a final prediction of 59.066%. It also became clear that using the dataset with only TF*IDF word matrix performed the best throughout all the models. Note that all other models were disregarded due to poor accuracy (less than 30%) and extremely slow runtimes (see Figure 5 in appendix for more details).

SVM was also considered when creating the ensemble since it performed better than CNB and logistic

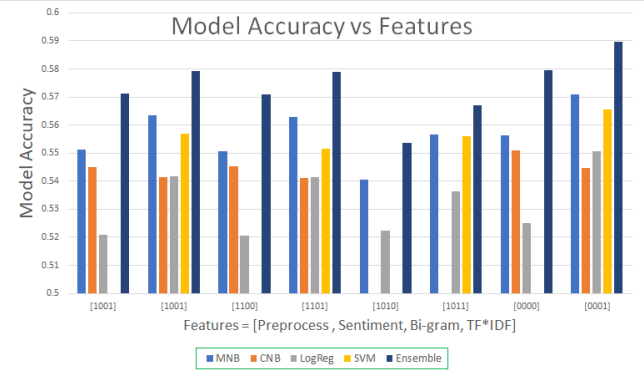

Fig. 3: Bar graph showing the accuracy of the best models for different feature vectors.

Model	Runtime (s)
MNB	0.34
CNB	0.35
LogReg	75.08
SVM	7.39
Ensemble	69.19
Our BNB	323.82
Ensemble + SVM	78.65
Ensemble - CNB + SVM	74.63

TABLE I: Runtime of the different models

regression (56.571% vs. 54.461% and 55.056% respectively). However, when testing it in the ensemble, lower accuracy was achieved. The same ensemble with SVM included scored an accuracy of 58.82% and another with CNB removed scored and accuracy of 57.56%. Both these models performed marginally worse than the best found model.

Finding the best hyperparameters for the ensemble model was an issue due to the many possibilities but tuning them separately generally resulted in a better ensemble model. Figure 1 shows that values around 0.3-0.4 works best for Multinomial NB and higher values for Complement NB but these values did not seem to reflect exactly in the model using the ensemble method model. The same thing occured with Complement NB. Figure 2 shows that higher α values yield higher results when ran on TF*IDF vectorized data with basic cleaning but for some reaon after manually tuning this parameter by hand in the ensemble method model, 0.08 yielded our best accuracy on cross-validation.

As for runtimes, generally the major increase in time was due to logistic regression as shown in Table I. This is due to the fact that it running gradient descent with a maximum of 2000 iterations is significantly slower with the amount of weights we are computing. However, in order to achieve low runtimes, it would be advisable to simply use naive Bayes. However, this project was simply to achieve the best accuracy possible, so runtime was not taken into account. There is a discussion to be had to whether increasing accuracy by 2% is worth increasing the runtime by 200 seconds.

VIII. Discussion and Conclusion

The preprocessing implemented seemed to improve results initially, but when the models became more complicated (using ensembles), anything more than the most basic preprocessing (removing stop words, convert everything to lower case, removing accents) seemed to reduce accuracy. The preprocessing steps taken may have been a bit too much for this classification problem. A problem with our implementation is that we did not test many different subsets of preprocessing (such as only lemmatizing without stemming and vice versa, trying different n-grams) mainly due to time constraints.

We did not expect the Bernoulli NB model to work very well as much information is lost when the features are made binary, which is also what was observed in the results. The Multinomial NB model worked quite well with Laplace smoothing ($\alpha = 1$) and after testing various forms of Lidstone smoothing ($\alpha < 1$) values it seemed to work much better. Another avenue which was not explored but may have yielded good results would have been categorizing the subreddits into larger classifications such as 'sports', 'video games', etc and running a two-step algorithm which would first attempt to categorize the general class and then have sub-models to classify each subreddit.

In the end, though unintuitive, extensive preprocessing did not yield the results we expected. Only simple preprocessing yielded the best results. It seems like reducing the number of features, whether seemingly trivial or not, did not increase accuracy in general.

IX. Statement of Contributions

Ian mostly dealt with all the data loading and preprocessing. Anthony implemented new features and helped with some model testing. Christos mostly dealt with testing different models. Everyone contributed equally to the write-up.

References

[1] Uysal, Alper Kursat, and Serkan Gunal. "The impact of preprocessing on text classification." Information Processing & Management 50.1 (2014): 104-112.

[2] Gilbert E. & Hutto C.J. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text, Eight Inernational AAAI Conference on Weblogs and Social Media
[3] Pennebaker, J. W., Francis, M., & Booth, R. (2010). Linguistic Inquiry and Word Count: LIWC 2010. Mahwah, NJ: Erlbaum.
[4] Salton, Gerard & Buckley, Christopher (1988), Term-Weighting Approaches in Automatic Text Retrieval, Information Processing and Management, pp.513-523
[5] Budzik, Jay (2019). "Many Heads Are Better Than One: The Case For Ensemble Learning", KDnuggets.
[6] McCallum, Andrew & Nigam, Kamal (1998). "A Comparison of Event Models for Naive Bayes Text Classification", Conference Proceeding s.

APPENDIX

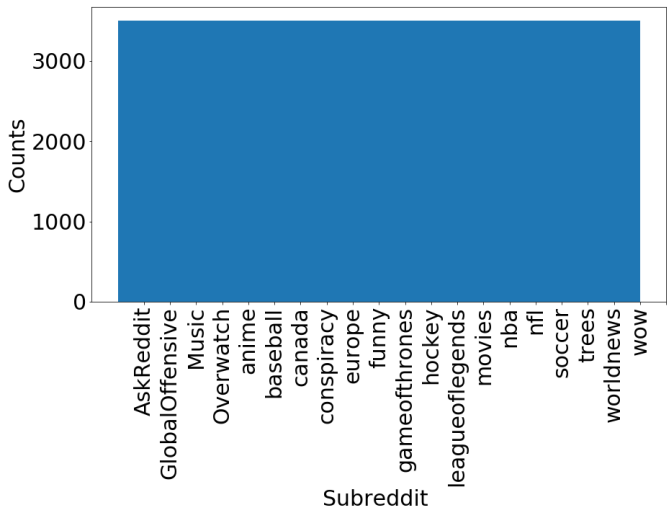

Fig. 4: Counts per subreddit in training data. Clearly there were equal amount of comments for each subreddit.

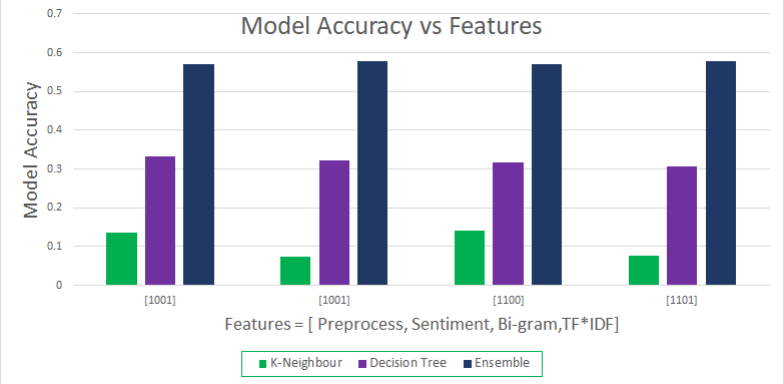

Fig. 5: Model accuracy of the Decision Trees and K-Neighbours. Since those models performed poorly compared to the other, they were not presented in main document. The ensemble method is also shown for comparison.