Reddit Classification

Annan Liu, Dara Shahriari, Edward Son

***Abstract*— In this work, we investigated the performance of various classification models, logistic regression (LR), Multi Nomial Naive Bayes (MNNB), Naive Bayes (NB), and Support Vector Machines (SVM) on one uncleaned dataset: Reddit comments. We demonstrate the effect of optimizing hyper-parameter use, cleaning methods, and feature selection in order to get the best results from each model. We compare the performance (accuracy and runtime) of all the models, coming to the conclusion that MNNB is superior alongboth metrics. Finally, we explore the effect of stacking to further improve our results.**

I. INTRODUCTION

Text categorization is becoming a more common problem in machine learning and is important for text mining and information retrieval. There are several machine learning algorithms that can be applied to this kind of problem and can solve this efficiently, e.g.: Naive Bayes, Logistic regression, Support Vector Machines, Linear Discriminant Analysis(LSA), as well as many algorithm we did not try in this study, e.g decision tree, k-nearest neighbor forests(KNN).

From the many applications of text classification, we looked into a Reddit comment classification problem. There are 70000 comments for training and each comment is in raw word form. Each comment belong to one of the 20 subreddits. If we input this raw data into the models, it will have a large time complexity and poor performance, so the input has to be pre-processed and feature extraction needs to be applied before applying the text categorization algorithm. The pre-processing step makes the training faster and improves the accuracy of the classifier by removing the noisy features and avoidding overfitting.

In the paper, the comments are first processed with lemmatization and are represented using DF-IDF, as a vector of weights for each word. Then for feature extraction, among many methods, we choose PCA and Chi-square. Then, after feature extraction step, the selected features will be passed to different classification models like Naive Bayes, Logistic regression, SVM and LSA, etc. Finally, the effectiveness of each classifier is measured by running time and accuracy. Experiments are conducted on the data set for Reddit comments.

II. PREVIOUS WORKS

Text Classification research has received much attention and has had several studies done.

In the paper "Chi Square Feature Extraction Based Svms Arabic Language Text Categorization System", they used binary classification method and Support Vector Machines(SVM) classification method for Arabic Language text classification [1]. They did not use stemming for reprocessing because it is not always beneficial for text categorization, since many terms may be conflated from the same root. The study used DF-IDF to pre-possess the data for SVM and use Chi-square to do the feature selection.

In the paper "Improving Multi-class Text Classification with Naive Bayes", they believed that Naive Bayes is suited to perform multiclass text classification, but there are flaws with the commonly used algorithms and they suggest a new framework for text feature selection, which is whether or not the distribution is discriminative and gave a framework which exposes the free parameters in such a scheme [2].

III. DATASET AND SETUP

Our data was partitioned into a csv file with the first column housing the comment data and the second indicating the subreddit from which it came. Using pandas, we were able to read the data into an indexed array. Next, each comment was stripped of unnecessary characters using a regular expression, split into words, and lemmatized accordingly. After our data was cleaned, we used TF-IDF-vectorizor along with the TF-IDF-transformer to rid our data of stopwords, vectorizing the comments into a binary array of features.

IV. PROPOSED APPROACH

In this study of Reddit comment text classification, we will apply the following methods and approaches to our work in a composition of the following steps:

A. Collect and pre-processing raw Text

1) Lemmatization: Lemmatization usually refers to doing things properly with the use of vocabulary and morphological analysis of work. Both lemmatization and stemming can be used to reduce inflectional forms and derivationally related forms of a word to a common base form. (E.g am, is, are -¿ be). In our research we tried lemmatization.

2) Stop Words Removal: In every language we have those words that contains less information such as 'the', 'a', etc. Those pronouns, prepositions and articles do not affect the meaning of the comment. We used a built in stop word list in the python Natural Language Toolkit(NTLK) that contains all the stop words for English.

B. Feature Extraction

After pre-processing the data, we should have each comment represented as a bag of word. To fit the Reddit comments into our classification model, we need to create a vector representation for each comment. Inside the vector each word i is assigned a weight. There are several ways to calculating weight, in this study we used TF-IDF.

1) TF-IDF: TF-IDF is a commonly used term weighting method called term frequency - inverse document frequency weighting. The weighting method uses numerical statistic measure to reflect how important each word is to a comment in certain class of corpus. TF(t,d) is the number of times that the word t occurred in the sample d, and IDF can be calculated by the form:

$$IDF(t, corpus) = log\frac{docsincorpus}{docswithtermt} \quad (1)$$

The measure of word important can be calculated by the product of the term frequency and the inverse document frequency(TF*IDF). So the term is important if it appears many time in the document and is relative rare word overall.

This algorithm looks at the dependency between features and classes, and ranks using this knowledge. The features that are independent of the classes are normally removed since they would be irrelevant in the classifier.

Using chi-square, we use the dependency ranking to keep only a subset of features. This method is faster than feature reduction, but may prove less viable as it is removing information from the data.

C. Feature Reduction

Initially our plan was to use PCA analysis and chi-square to reduce the number of features we had to work with. However, these both proved to lower accuracy so we chose to tamper with the hyper-parameters of the TFIDF-Vectorizer instead:

1) Min DF: Min DF is generally used to remove terms that appear too infrequently. For example, min df = 5 means ignore terms that appear in less than 5 documents. After some testing, we found a min df of 2 to be optimal.

2) Max DF: Max DF is generally used to remove terms that appear too frequently. Max DF = 0.50 means ignore terms that appear in more than half of the documents. Since there were 20 classes, we opted to use .05 to ensure that our features only represented words that were unique to the classification of one subreddit.

D. Text Classifiers Models

There have been many classifiers developed for text classification problem. In this study we tried many linear classifications as well as the following:

1) Bernoulli Naive Bayes: Bernoulli Naive Bayes is a family of simple "probabilistic classifiers" with strong independence assumptions between the features. Bernoulli means that the features are independent booleans describing inputs. In this research we implement the Multiclass Bernoulli Naive Bayes because we have 20 subreddits. Thus the likelihood of a comment given a subreddit k is given by:

$$log(\theta_k) + \sum_j (x_j log(\theta_{jk}) + (1 - x_j) log(1 - \theta_{jk})) \quad (2)$$

where θ_k is probability that the sample belongs to subreddit k from all samples. θ_{jk} is the probability that feature j in all samples belongs to subreddit k. After calculated the likelihood for every sample, we got a prdiction of the probability of this sample belong to each of 20 subreddit, we choose the one with largest probability. We also used Laplace smoothing when counting θ_k and θ_{jk} to solve the case when a frequency-based probability is zero. Laplace smoothing is a technique used to smooth categorical data and a way of regularizing Naive Bayes. After applying Laplace smoothing, no probability will be zero.

2) Multinomial Naive Bayes: In Multinomial Naive Bayes, the feature factors are the frequencies. Each sample is represented by a multinomial (p_1,...,p_n)

where each p is the probability that event i occurs. The likelihood for Multinomial Naive Bayes is

$$p(x|C_k) = \frac{(\sum_i (x_i))!}{\prod_i (x_i)!} \prod (p_{ki})^{x_i} (3)$$

where p_{ki} is the probability of class C_k generating the term x_i.

3) Logistic Regression: Logistic regression is a linear classifier that measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function. Logistic Regression directly models the log-odds of a dependent variable belonging to one of class. In this paper we used Logistic regression classifier from Sklearn linear with multi-class = multinomial, solver = "lbfgs". Lbfgs solver handles multinomial loss and support L2 penalties.

4) Support Vector Machines (SVM): For the SVM classifier, we have decided to go with a linear SVM as opposed to a kernel SVM, since a linear one is faster at optimizing a problem due to less parameters. Since we are working with a very large feature set, it made sense to use a faster model, as most of the work was done in the pre-processing steps. SGDClassifier from Sklearn is used with parameters loss='modified_huber', n_iter=1000 and tol=1e-3. The loss parameter smooths the loss of outliers and probabilities.

5) LSA: In this model we attempted to use LSA (latent semantic analysis), which does not center data like PCA does. Therefore, it can work with sparse matrices. This method allows us to reduce the feature space without removing any features. More concretely, we use the TruncatedSVD decomposition from Sklearn to reduce the number of features to 10 000 for SVM classifier, to gauge whether it would prove useful to use with other models.

6) Stacking: The stacking method aims to improve classification by combining the output of other classifiers, and using those probabilities as new features to train a new meta model. For this project, the StackingClassifier from the mlxtend library will be constructed using the output from the SVM output and the Multinomial Naive Bayes output, into a logistic regression meta model.

E. 5-fold validation

To evaluate our models on the training set, each model will run a 5-fold cross validation using Sklearn's cross_val_score with a fold of 5. This will return an array of scores for each fold, and the mean will be used for the final accuracy score of the model.

F. Accuracy

The main metric we will be using to compare the different models is accuracy, simply because the competition uses accuracy to score. Therefore it will be easier to compare our results to our submissions.

V. RESULTS

Table I showcases the accuracy achieved for every class when running our different models. It can be seen that some classes were harder to identify, for example "AskReddit" and "funny" were almost always the most misclassified in any model. This might be due to the variety of words that could appear in these comments, as there is no particular subject. The following abbreviations are used for the models: Multionomial Naive Bayes (MNNB), Bernouli Naive Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM), Stacking Classifier (STCK)

Subreddit	**Model Accuracy**				
	MNNB	**NB**	**LR**	**SVM**	**STCK**
AskReddit	0.26	0.32	0.26	0.29	0.34
GlobalOffensive	0.64	0.56	0.62	0.65	0.72
Music	0.60	0.87	0.71	0.72	0.82
Overwatch	0.63	0.75	0.76	0.74	0.79
anime	0.62	0.73	0.64	0.60	0.72
baseball	0.73	0.50	0.65	0.70	0.78
canada	0.42	0.54	0.45	0.50	0.64
conspiracy	0.41	0.50	0.40	0.43	0.55
europe	0.48	0.56	0.50	0.52	0.64
funny	0.25	0.14	0.18	0.24	0.27
gameofthrones	0.74	0.89	0.79	0.84	0.86
hockey	0.70	0.68	0.65	0.70	0.74
leagueoflegends	0.76	0.76	0.67	0.68	0.82
movies	0.55	0.56	0.61	0.60	0.75
nba	0.66	0.55	0.70	0.72	0.78
nfl	0.65	0.82	0.64	0.70	0.74
soccer	0.76	0.50	0.64	0.70	0.77
trees	0.47	0.36	0.50	0.51	0.65
worldnews	0.32	0.38	0.32	0.36	0.51
wow	0.72	0.90	0.72	0.67	0.80

TABLE I: Accuracy of models for each class

Table II showcases the total accuracy achieved on the training data using 5-fold cross validation, the submission accuracy achieved on Kaggle, and finally the runtime for fitting.

	Models				
	MNNB	**NB**	**LR**	**SVM**	**STCK**
Test Accuracy	0.5646	0.4967	0.5489	**0.5812**	0.5808
Final Accuracy	**0.5788**	-	0.5292	0.5571	0.5618
Runtime (sec)	**0.2489**	0.2946	42.9391	2.0381	13.8072

TABLE II: Total accuracy and runtime of each model

A. Bernouli Naive Bayes

The Bernouli Naive Bayes model which was implemented from scratch was our lowest performing model on our training data. This is most likely due to the fact that the sklearn implementations are more optimized and have multiple hyper parameters to tune the performance. Also, since this was a multiclass prediction task, intuitively the bernouli model would not be the most appropriate, since it only checks if a single feature occurs or not. This model had a runtime of 0.2946 seconds, which was the second fastest out of the implemented models.

B. Multinomial Naive Bayes

The Multinomial Naive Bayes was one of our best performing model with an accuracy of 0.5646 using 5-fold cross validation on the training data. While it did not achieve the highest accuracy there, it was the top scorer on the submission accuracy with 0.5788. It also had the fastest runtime of 0.2489 seconds.

C. Logistic Regression

The Logistic Regression model performed better than Bernouli Naive Bayes with an 5-fold cross validation accuracy of 0.5489, and a slow runtime of 42.9391 seconds. The submission accuracy was 0.5292 which was lower than what we predicted, which could indicate overfitting.

D. SVM

The SGD classifier used implements a linear SVM which gave some promising results. It was the fastest algorithm with a fitting runtime of 2.0381 seconds, and an accuracy of 0.5812 using 5 fold cross validation. This was the highest accuracy achieved on the training data, however the submission score was only 0.5571. What this signifies is that the model is overfitting: it scores better on the training data than it does on actually testing data. This could be due to the use of chi-square variable ranking. In the case of this project, 15500 of the highest ranking features were used to train the SGD classifier. This number was found through trial and error, in increments of 500 features. The reason for overfitting could potentially be due to the fact that the feature ranking is different for the training set. Therefore, when the test data is fitted using the top features found in the training set, it does not perform as well.

E. Stacking Classifier

This meta-classifier was an attempt to improve on the existing classifier, by combining their output to train a new classifier. In this case, the MNNB and SGD classifiers were trained on the data as previously mentionned. The output of these classifiers were probabilities for each class, which were then fed to the meta classifier using logistic regression. The reason for this order was simply because the MNNB and SGD classifiers seemed to perform the best on the original dataset. Thus, the idea was to maximize the initial predictions with these classifiers, and then feed it into the third best classifier which was logistic regression. As we can see from the table, this stacking classifier achieved an accuracy of 0.5808, with a runtime of 13.8072 seconds. Once again however, this classifier did not score as well on the submission, with an accuracy of 0.5618. Like the SGD classifier, it seems the model is overfitting, most likely due to the same variable ranking implementation.

F. LSA

Latent semantic analysis was tested on the SGD classifier by creating 10 000 components from the TF-IDF features. The TruncatedSVD function from Sklearn was used to fit the data. The classifier fitting took 158.7149 seconds and achieved an accuracy of 0.5642. As we can see in figure 1, the accuracy tends to increase as the number of components increases. The experiment was run with 100, 1000 and finally 10 000 components. However, since these results were not better than most of our current pre-processed data, LSA was not used on the other models. This procedure was

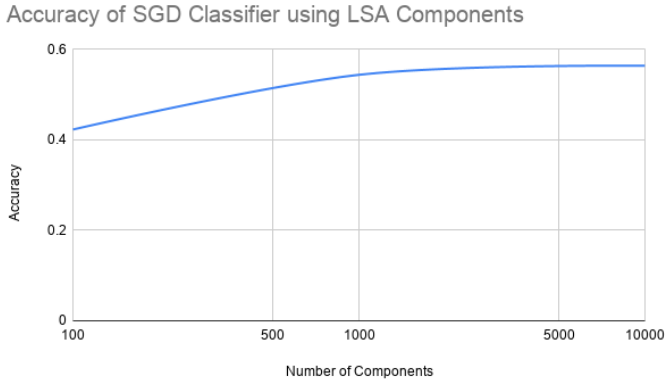

Fig. 1: Accuracy of SGD using LSA components

quite slow to run, as it took approximately 8 hours to create 10 000 components. The result was comparable to MNNB without LSA, and lower than SGD without

LSA. To improve this experiment, perhaps using more components would have yielded a better result, however due to time and processing power constraints, creating more components would have been infeasible.

VI. CONCLUSION

In conclusion, we have created a pipeline that implements data pre-processing such as TF-IDF vectorizing with stop word removal and lemmatization to create features. From these features, we applied variable ranking using chi-square algorithm and latent semantic analysis to reduce the number of features. Then, we trained 5 different models: Bernouli Naive Bayes, Multionomial Bayes, Logistic regression, SGD classifier and a meta-model using stacking. To evaluate our models, we use accuracy in order to easily compare with the submission score from Kaggle. On the training data, we use 5-fold cross validation to evaluate the accuracy of our models. Our highest scoring model on the training data was the SGD classifier with 0.5812, however our highest scoring model on Kaggle was Multinomial Bayes with a score of 0.5788. The latter also had the fastest runtime of 0.2489 seconds to fit the training data. An idea for the future would be to explore boosting as an ensemble method, to potentially give more weights to classes that are often misclassified.

VII. CONTRIBUTIONS

Edward worked mainly on feature reduction using LSA and feature extracting using chi-squared variable ranking, as well as the SGD classifier and the stacking classifier.

Dara worked on optimizing MNNB model and tampering with hyper parameters to get the best results as well as data clean and feature selection.

Annan worked on implementing the Bernouli Naive Bayes model, vectorizing the data and the logistic regression model.

REFERENCES

[1] A. Mesleh, "Chi square feature extraction based svms arabic language text categorization system," *Journal of Computer Science*, 2007.

[2] J. D. Rennie, "Improving multi-class text classification with naive bayes," *artificial intelligence laboratory*, 2001.