Performance Analysis of Multiple Machine Learning Classifiers on Twenty-Class Classification of Reddit Comments

Christopher Zheng[1], Hongjian Gu[2] and Haoxuan Shi[3]
GROUP 89

***Abstract*— Text classification is one of the most salient applications of modern day natural language processing To undertake a twenty-class text classification task on tens of thousands of Reddit comments, we study and make use of various natural language processing techniques as well as widely utilized machine learning models. Instead of exhausting all possible pre-processing methods, we choose to handle raw texts with strategies such as lemmatization and vectorization which perform better than other methods based on observation. Besides, we implement the model of Bernoulli Naive Bayes and examine multiple other classifiers including Logistic Regression, Naive Bayes models with Lidstone smoothing and Linear Support Vector Classifier. Moreover, we attach a Long Short-Term Memory architecture and merge all base models into a synthetic ensemble classifier which is proven to outperform all the other base models in terms of robustness against instability and of accuracy, with a score of 58.61% on Kaggle and an increase of up to 4% over selected base models.**

I. INTRODUCTION

A. Task Description

Multi-class text classification has been a heated topic. In this work, we are expected to implement the model of Bernoulli Naive Bayes and utilize multiple models to classify the given texts to twenty categories.

The provided text dataset is a collection of Reddit comments. Reddit is a large-scale social media platform where users are able to post and comment on threads in communities, called subreddits, with distinct themes. Every comment in the training dataset is denoted by a unique ID number and one of the twenty subreddits.

We study how widely-used natural language processing techniques can help with the text pre-processing and to what extent they can contribute to the overall classification accuracy. Besides, we compare and contrast the overall performance of various models as well as their accuracies on specific categories of comments. Furthermore, we also investigate how to combine multiple models into an *ensemble* [1] model to enhance the overall result of classification by stacking and having all models to vote for each classification task according to a set of voting policies.

[1]Christopher Zheng (260760794, Corresponding Author): christopher.zheng@mail.mcgill.ca
[2]Hongjian Gu (260772647): hongjian.gu@mail.mcgill.ca
[3]Haoxuan Shi (260779480): haoxuan.shi@mail.mcgill.ca

B. Important Findings

In this work, we discover that the classification accuracies are dependent on a number of factors. At the stage of pre-processing, simultaneously applying all the techniques is not ideal since that may lead to severe overfitting or overly loses feature information as introduced in Section III. For classifiers, one model may appear to be dominant on specific tasks or datasets, but this excellence in accuracy is not necessarily guaranteed as shown in Section IV. Dynamically combining multiple models and taking advantage of base models' merits can make sure the meta classifier is both accurate and stable as discussed in Section V.

II. BACKGROUND AND RELATED WORK

A. Background

Categorizing texts can provide insights of the people who generate those texts and of the topics that the texts are associated with. Bernoulli Naive Bayes classifier is popular with document classification tasks, particularly short texts [2]. In a multi-variate Bernoulli event model, all features are assumed to be independent and binary instead of being continuous token frequencies. Formally, let x_i denote a boolean variable (expressing token occurrence or absence of the i^{th} token in the vocabulary list), and the likelihood of a document given the class C_k is that [3]

$$P(x|C_k) = \prod_{i=1}^{n} p_{k_i}^{x_i}(1 - p_{k_i})^{(1-x_i)}$$

where p_{k_i} denotes the probability of the token x_i being produced by the class C_k.

B. Related Work

Text classification is a field of natural language processing under intense studies [4].

One salient application of text classification is *sentiment analysis*, particularly in the domain of movie reviews, product reviews, blogs, etc [5]. (Pang and Lee [6] surveys previous research on sentiment analysis.) Previous works [7] apply machine learning algorithms to classify movie reviews. Apart from traditional pre-processing methods, an article [8] demonstrates that using emoticons as labels for sentiment analysis is useful for to some extent reducing dependencies in machine learning. A quite recent publication [9] proposes to analyze Tweet sentiments (of Twitter, a social media similar to Reddit) in an ensemble classification system, which is proven to have higher accuracy and lower variance scores.

Text classification techniques are also applied to fields such as *spam filtering*, *language identification*, and *readability assessment*. To fight against spams, relaxed online support vector machines [10] as well as Naive Bayesian classifiers [11, 12] are examined to be effective and easy-to-implement. Besides machine learning approaches, a case-based method [13] is also practical to filter spams and to track concept drifts during application. To identify a language in a context, decision trees [14] play an important role in maximizing the accuracy of language identification [15]. Further, an off-the-shelf tool, *langid* [16], is popular in this field. Readability assessment is widely applicable. On one hand, it can be used for text simplification [17] for various purposes. On the other hand, this assessment along with machine learning techniques can help to target different groups of people in terms of their regions, interests, incomes, education levels, numbers of languages mastered, etc [18].

Lemmatization [19] and stemming [20] are widely utilized methods of dimension reduction in the applications of text classification. Lemmatizing a word is to convert it back to its primary form (as in a dictionary) while stemming it is to truncate its tail and reduce it to its stem. Both methods' results depend on the specific datasets and are hardly applied simultaneously [21].

III. Dataset and Setup

The provided training dataset of Reddit comments contains 70,000 threads of comments along with their corresponding subreddit categories. Each category includes exactly 3,500 comments as illustrated by the Waffle Plot 1. Each example of the dataset contains its ID number, the content of the comment, and the subreddit that it belongs to.

TABLE I: Numbers of valid tokens in the provided training dataset with different NLP text processing techniques.

Preprocessing Techinique	No. of Tokens (unigram)
No operation	87000
Invalid token removed (Regex)	72990
Lemmatization (NLTK)	66658
Stemming (NLTK)	54040
English stopwords excluded (NLTK)	72685
Non-English excluded (NLTK)	31539

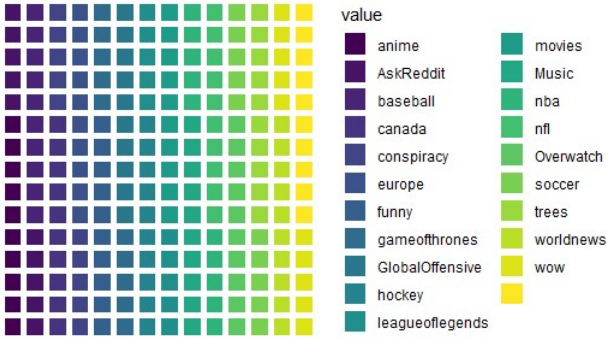

Fig. 1: Quantity distributions of twenty subreddits are equal.

A. Preprocessing

We start by cleansing all the non-English tokens in the dataset, converting them to lowercase, lemmatizing them, and excluding NLTK English stop words (e.g., *the, I, a*).

We present the number of valid tokens after applying

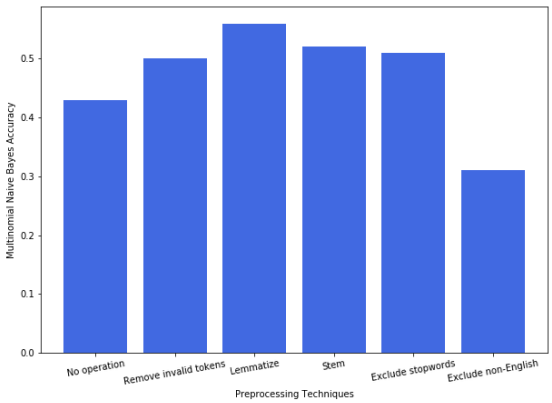

Fig. 2: The accuracy scores of Multinomial Naive Bayes classifier with default parameter values after being applied each pre-processing technique.

each text processing method in Tab I. The numbers may convey to what extent the methods are capable of shrinking the feature space, but it is crucial to note that small numbers do not necessarily suggest that the corresponding methods can produce accurate output since they may lose some significant features if their shrinking policies are not suitable enough in our case. As shown in Fig. 2, lemmatization alone demonstrates the best performance, whereas removing non-English words, which reduces the feature space to the largest extent, actually outputs the poorest result.

B. Features.

Feature Extraction. The Bag-of-Words (BOW) model is very prevalent in probability-based text categorization [22]. We vectorize the tokens of the corpus using *TfidfVectorizer* and obtain the N-Grams. (Using insights from the literature, uni-grams and bi-grams may be desirable, and our experiments demonstrate that unigrams are recommended in this case). Purely counting the numbers of word occurrences (using CountVectorizer) is relatively prone to problems, such as extremely skewed token distributions and irrelevant words.

After scrutinizing the tokenized word list, we notice that some typos do exist. (This is reasonable since the dataset is from Reddit where language formality is not required.) To recover badly-typed tokens to standard English vocabulary, we take advantage of the *SpellChecker* package, but the auto-correction has trouble recognizing certain trending slangs or domain-specific proper nouns due to quite limited sizes of corpora. Further, the function is unacceptably slow for a small project. Consequently, the attempt of typo correction does not produce promising result, therefore, we decide to rely on the result of *TfidfVectorizer* solely.

Feature Selection. Promising feature selection can help reduce the dimensionality of the dataset, drive the training process faster, and, more importantly, improve the classification performance. For examples that contain small numbers of features, statistical Pearson cross-correlation analysis is a choice.

In this case, however, the large number of features suggests that we may consider methods such as Mutual Information [23], Principal Component Analysis (PCA) [24], and Pearson's Chi-squared test [25]. During our experimentation, we observe that the Chi-squared test witnesses the highest accuracy increase on our sample dataset, therefore, it is implemented for selecting the most representative and salient features. We apply the test, sort the features according to ascending p-values, and discard the features at the bottom of the ranking as shown in Fig. 3 and 4. Eventually, we select fifteen thousand features based on five base models' accuracies after fine tuning (Fig. 5).

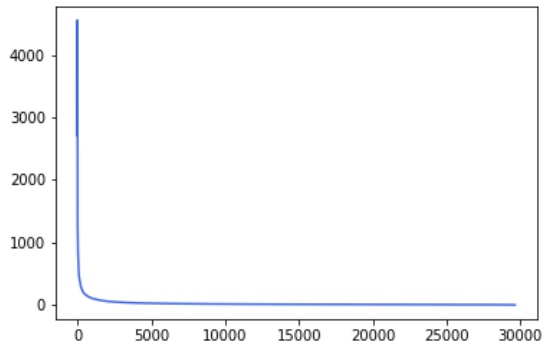

Fig. 3: Pearson's Chi Squared test ranking of features.

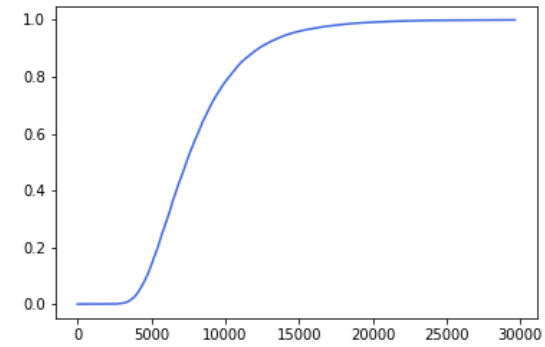

Fig. 4: P value ranking of features.

IV. Proposed Approach

We aim to properly combine multiple models into an *ensemble* model to achieve better classification performance. Therefore, to start with, we briefly introduce, compare and contrast our *base models*, i.e. the models (Logistic Regression, Naive Bayes models with Laplace/Lidstone smoothing, Linear Support Vector Classifier, and Long Short-Term Memory [26]) that we utilize as the building blocks of the final category predictor. In this section, for all the base models, we present their best hyper-parameters tuned using Grid Search [27] or 5-fold cross validation and their individual classification performance with proper feature selection.

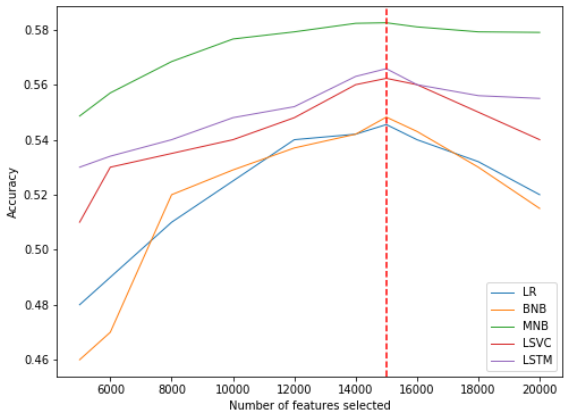

Fig. 5: The accuracy scores reach the max when we select fifteen thousand features of tokens.

A. Logistic Regression

Logistic Regression [28] estimates parameters of a logit model and classifies examples among classes based on probability. We explicitly denote that the feature value distribution is multinomial to match the real-world scenario and its other parameters are listed in Table II.

B. Two Naive Bayes

We incorporate two similar but different Naive Bayesian models: Bernoulli Naive Bayes and Multinomial Naive Bayes [29].

The BernoulliNB model, as discussed in Section I, is excellent in modelling token presence and absence. From our observation, this characteristic renders the model very capable of separating out categories that contain very specific terminologies or proper nouns (e.g. League of Legends and Game of Thrones). The most important parameter, *alpha* of the smoothing, is 0.0001 (Table II).

The Multinomial counterpart takes input values that represent the frequencies with which certain events are generated w.r.t a multinomial distribution. Based on past literature, this event model is typically selected for document classification [30]. The value of the Lidstone smoothing is 0.23 (Table II).

C. Linear Support Vector Classifier (Linear SVC)

Linear SVC is similar to Support Vector Classifier with linear kernel to some extent, whereas it is relatively scalable and performs better with samples of large sizes. As for the parameters, all other parameters are default except for that the crucial penalty parameter C of the error term is tuned as 0.23 (Table II).

D. Long Short-Term Memory (LSTM)

LSTM is a recurrent neural network [31] architecture widely used in deep learning. We decide to add a neural network into our ensemble model because we notice that the aforementioned models individually have quite similar prediction patterns and a new type of model may mitigate this issue. After fine tuning, our LSTM model has one layer of 128 LSTM units (with the activation function as *tanh*) followed by twenty dense softmax units. The dropout ratios (the spatical dropout, recurrent dropout, etc) are all set to 70%. The word embedding dimension is 256 and the stochastic gradient descent batch size is 256 as default.

E. Ensemble method

We generally follow the *Stacking* Ensemble method. However, instead of feeding immediately the prediction results of the base models on one portion of the training set to the meta classifier, we choose to consider them as a new feature and horizontally stack them to the remaining portion of the training set. We again train the novelly modified examples on our base models, and pipeline the results to the meta classifier.

Our meta classifier controls the voting procedure and is responsible for generating final predictions. Based on a *hard-vote* scheme (i.e., each model has one vote and the mode of all votes is selected as the final prediction), the meta classifier also considers about different models' fields of expertise. If a base model makes a prediction of its domain, its prediction is granted exceptionally more weights when the meta classifier summarizes the votes. This field of expertise is manually constructed by us scrutinizing the model performance on twenty subreddits. For instance, as mentioned in IV-B, Bernoulli Naive Bayes is excellent in making predictions of specific categories such as WoW, Game of Thrones and League of Legends. (Please note that a base model does not necessarily have an expertise.)

V. Results

We present the classification accuracies and training time duration of five base models and the ensemble models based on 5-fold cross-validation in Table II. In terms of the accuracy scores, the base model Multinomial Naive Bayes obtains the highest score of 58.01% but when we examine solely this model on

Model	Base model accuracy	Parameter(s)	Training time on the train set
Logistic Regression	54.55%	solver='saga', multi_class='multinomial', C=3, max_iter=99999	31.84 s
Bernoulli NB	54.82%	alpha=0.0001	1.20 s
Multinomial NB	58.01%	alpha=0.22	1.33 s
Linear SVC	56.23%	C=0.26, loss='hinge'	8.22 s
LSTM	56.58%	word embedding dimension = 256, batch size = 256, dropout = 0.7, 1 layer of 128 LSTM units, activation func = tanh, 1 layer of 20 softmax units	∼3 h
Ensemble w/o LSTM	57.00%	/	40.30 s
Ensemble	**58.61%**	/	∼3 h

TABLE II: Accuracy scores, parameters, and training time of base models and the ensemble model.

Kaggle, its accuracy drops by almost two percent. We suspect that it may suffer overfitting with a Lidstone smoothing of 0.22 because it is not robust enough against unforeseen patterns in the test set. The ensemble method enclosing all the base models achieves an accuracy of 58.61% while its counterpart without LSTM losses 1.61% in accuracy.

We include two ensemble methods with and without LSTM to illustrate the point of time efficiency (specs listed below[1]). The LSTM base model requires approximately three hours of training and this time is added to the overall training time, which is consequently more than three hours, as the ultimate ensemble method includes LSTM as one of the bases. For other base models, both Naive Bayes variants are rapid and take slightly more than one second for one training. Besides, Linear SVC's runtime is decent and this model consumes 8.22 seconds per training. Logistic Regression, however, is relatively slow as it is relatively easily influenced by the colossal number of token features in text classification tasks.

To summarize, the ensemble method with Logistic Regression, Bernoulli Naive Bayes, Multinomial Naive Bayes, Linear SVC, and LSTM produces the optimal accuracy. Although the training time is quite long, it is negligible since long offline training time generally does not negatively impact online classification efficiency based on our observation.

VI. Discussion and Conclusion

Admittedly, certain parts of this work are not exhausted nor complete, and some procedures are not fully sound. As the future work, we aim to explore more types and structures of neural networks (e.g., convolutional neural networks and other categories of recurrent neural networks) and examine their performance on text classification. Currently, we are unable to implement such models in this project due to the limited computing resources. Next, we may also further study other types of ensemble methods such as Bagging and Boosting.

In this work, by attempting to classify the Reddit comments into twenty classes using multiple models along with various Machine Learning and Natural Language Processing algorithms and techniques, we discover that the classification accuracies depend on a lot of factors, particularly the dataset and its corresponding strategies. For text pre-processing, simultaneously applying all the techniques is not necessarily the optimal solution since that might causes overfitting or overly loses feature information. For classifiers, certain model may appear to be dominant on specific tasks or datasets, but this excellence is not guaranteed. Combining multiple models and taking advantage of base models' merits can make sure the meta classifier is both accurate and stable.

VII. Statement of Contributions

- **Define the question:** Done as a group.
- **Analyze and develop the methodology:** Done as a group.
- **Data preparation and analysis:** Christopher
- **Implement the solution by coding:**
  - Text pre-processing and the LSTM architecture: Christopher.
  - The series of comparison experiments, the meta classifier, and the overall pipeline: Hongjian.
  - The model of the Bernoulli Naive Bayes classifier and the ensemble model (stacking): Haoxuan.
- **Writing up the report:** Christopher

[1]Implementation specs: models are implemented in Python 3.7.3 and our experiments are conducted on a PC with Intel i7 CPU and 8 GB RAM.

REFERENCES

[1] T. G. Dietterich *et al.*, "Ensemble learning," *The handbook of brain theory and neural networks*, vol. 2, pp. 110–125, 2002.

[2] A. McCallum, K. Nigam, *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, pp. 41–48, Citeseer, 1998.

[3] S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, "Some effective techniques for naive bayes text classification," *IEEE transactions on knowledge and data engineering*, vol. 18, no. 11, pp. 1457–1466, 2006.

[4] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[5] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[6] B. Pang, L. Lee, *et al.*, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[7] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.

[8] J. Read, "Using emoticons to reduce dependency in machine learning techniques for sentiment classification," in *Proceedings of the ACL student research workshop*, pp. 43–48, Association for Computational Linguistics, 2005.

[9] N. Saleena *et al.*, "An ensemble classification system for twitter sentiment analysis," *Procedia computer science*, vol. 132, pp. 937–946, 2018.

[10] D. Sculley and G. M. Wachman, "Relaxed online svms for spam filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 415–422, ACM, 2007.

[11] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, "An evaluation of naive bayesian anti-spam filtering," *arXiv preprint cs/0006013*, 2000.

[12] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam filtering with naive bayes-which naive bayes?," in *CEAS*, vol. 17, pp. 28–69, Mountain View, CA, 2006.

[13] S. J. Delany, P. Cunningham, A. Tsymbal, and L. Coyle, "A case-based technique for tracking concept drift in spam filtering," in *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 3–16, Springer, 2004.

[14] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[15] S. Jarvis, Y. Bestgen, and S. Pepper, "Maximizing classification accuracy in native language identification," in *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pp. 111–118, 2013.

[16] M. Lui and T. Baldwin, "langid. py: An off-the-shelf language identification tool," in *Proceedings of the ACL 2012 system demonstrations*, pp. 25–30, Association for Computational Linguistics, 2012.

[17] S. Aluisio, L. Specia, C. Gasperin, and C. Scarton, "Readability assessment for text simplification," in *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pp. 1–9, Association for Computational Linguistics, 2010.

[18] S. E. Petersen and M. Ostendorf, "A machine learning approach to reading level assessment," *Computer speech & language*, vol. 23, no. 1, pp. 89–106, 2009.

[19] M. Gross, "Lemmatization of compound tenses in english," *Lingvisticae Investigationes*, vol. 22, no. 1, pp. 71–122, 1998.

[20] J. B. Lovins, "Development of a stemming algorithm," *Mech. Translat. & Comp. Linguistics*, vol. 11, no. 1-2, pp. 22–31, 1968.

[21] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of finnish text documents," in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pp. 625–633, ACM, 2004.

[22] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization.," tech. rep., Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[23] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[24] I. Jolliffe, *Principal component analysis*. Springer, 2011.

[25] S. Lemeshow and D. W. Hosmer Jr, "A review of goodness of fit statistics for use in the development of logistic regression models," *American journal of epidemiology*, vol. 115, no. 1, pp. 92–106, 1982.

[26] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[27] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification," 2003.

[28] S. H. Walker and D. B. Duncan, "Estimation of the probability of an event as a function of several independent variables," *Biometrika*, vol. 54, no. 1-2, pp. 167–179, 1967.

[29] A. M. Kibriya, E. Frank, B. Pfahringer, and G. Holmes, "Multinomial naive bayes for text categorization revisited," in *Australasian Joint Conference on Artificial Intelligence*, pp. 488–499, Springer, 2004.

[30] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 616–623, 2003.

[31] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.