Investigation of feature extraction and model selection in Natural Language Processing

Han Zhou [1], Hao Shu[2] and Gengyi Sun[3]

***Abstract*— In this project, we investigated approaches to accomplish natural language processing and classification. A data set with 100000 comments extracted from Reddit is used to train and validate the model accuracy. Some Python classes were used to help in the feature extraction and model building. Uni-gram Bag-of-words are used as the feature pattern when extracting the features from corpse. After investigation of various classifiers, we have constructed an ensemble of classifiers that's able to classify comments that acquired from a limited subreddits in Reddit. The result of prediction maintains over 58.7%, with no additional data required. Classifiers which has a outstanding performance in the classification of matrix produced from natural language has been recorded.**

I. INTRODUCTION

A. Preliminaries

Reddit is a famous public online forum with more than 330 million users and 1.6 billion visits up-to July 2019 [1]. It gathers people from all fields with different interests and separates the place of discussion into different "subreddits." As it records a huge amount of interactions between Reddit users and discussion of one specific area of knowledge, the comments from each subreddit could be used as a good resource of natural language process and classification.
Term-Frequency Inverse Document-Frequency (TF-IDF) is a common way of extracting features from natural language datasets. It weighted each term in a document by both its frequency in this document and its overall frequency among the corpse. This method has a significant effect on reducing the weight of frequently used words ('one,' 'like,' etc.) which are not included in the filtered stop words.

B. Related Work

The classification of natural language has always been a popular research topic in machine learning. The differences between models capable of classifying sentences and a typical machine learning model are that, the data provided is in form of raw sentence consisted of only words instead of selected and weighted numeric features waiting to be learnt. The focus in natural language processing comes out to be strategies in discovering the cohesion between documents and extracting them as numerical features. However, since not all the comments on Reddit are tied closely with the area of subreddit, to have a precise prediction for a random comment of which subreddit it comes from turns out to be a very challenging task. As the cohesion and quality of posts vary widely from one subreddit forum to another, model performance will be different regarding chosen datasets[2]. In the previous text classification contest, models with high complexity, which involve Latent Dirichlet Allocation and sentiment analysis, demonstrate worse performance compares to simpler uni-gram bag-of-words[2].

C. Task Description

In this project, we are given a dataset that's consists of 100k comments from 20 diverse popular subreddit forums. 70k of them come with a known subreddit tag, which can be used as a training dataset in supervised learning. The rest 30k is unclassified dataset waiting to be categorized and will be used as validation dataset in Kaggle contest.
The goal of this project is to build up a machine learning program that's capable of extracting features from the dataset, learn the featured trend and effectively classify a comment from a limited set of subreddits. The main task can be break down into two distinct phases: first,find a method to extraction features from corpse into matrices, the next step is to select a set of classifiers to learn the vectorized dataset as input features and output trained model for prediction.

II. DATASET AND SETUP

A. Data acquiring and prepossessing

The training dataset is a CSV file with 70k comments and three columns for each sample: the comment ID, the text of the comment, and the name of the target subreddit for that comment. Comments from each subreddits are distributed evenly. The pre-selection is assumed done by the provider so that each comment has some extends of correlation with the topic of its belonging subreddit. The misshape comments (ones which empty slots or meaningless content) have been filtered out before distributed, so the rest task is to tokenize and vectorize the data.

B. Lab design and environment setup

Python is used as the coding language for this project, in addition, we used some imported package to help us in extracting features and building the classifier. Mainly the numpy and pandas package for data import and organization, sklearn package for model construction, as well as nltk for word tokenization and lemmatization. In general, our model of prediction consists of two parts: the first part is to

[1]**Han Zhou** Third year Software Engineering student at McGill University.

[2]**Hao Shu** Third year Software Engineering student at McGill University.

[2]**Gengyi Sun** Third year Software Engineering student at McGill University

broke down and extract features from the provided corpse; the second part is to train models to classify and output predictions for the test datasets.

C. Ethic Consideration

Since the comments are acquired directly from the Reddit forum, inevitably, there will be some sensitive information related to some users' privacy. As we are using this dataset to train our model, we shall keep those acquired data in the black box and let the model do the algorithm alone. Since we are not supposed to interfere with the processing by any human inspections, those datasets should leave untouched during the entire process anyway.

III. PROPOSED APPROACH

A. Implementing Multi-class Bernoulli Naive Bayes

Before the experiment starts, we first implemented the Bernoulli Naive Bayes that supports multiple classes using the mathematical theory learnt from the class. After implementation, we imported the Multinomial Naive Bayes classifier from sklearn and make_classification to generate the dataset for validation. In a trial of 10, a dataset of 2000 samples was generated. Each sample had 20 informative features and output was separated into 5 classes. Two classifiers were using the same dataset training and validation, and after all, displayed similar predicting accuracy and variation.

	MNB	MNB_N
ACC	0.459%	0.481%
TIME	0.030s	1.968s
VAR	0.860	1.073

Where MNB_N represents the new Naive Bayes built by scratch, Acc is the average accuracy of the classification, and VAR stands for the variation of the predictions regarding the sample size. As the dataset is not especially generated for MNB classifier, the prediction for both classifiers are not satisfactory as they do in a typical dataset, but their accuracy indicates their similar functionality, as a random classifier will have its predicting accuracy around 20%. However, the time cost of MNB imported and MNB built is different: the MNB built by scratch costs significantly more time in fitting and predicting the results. The possible reason for the low time efficiency could be a better vector multiplication using numpy is implemented in MNB from sklearn instead of looping through the array. This low time efficiency increase as the size of the input dataset grows.

B. Pipeline design

The pipeline is imported from sklearn and used to construct a more directed machine learning system. The pipeline has three procedures: A CountVectorizer which breaks the sentence into Uni-gram bag-of-words, a TFidfTransformer that counts the weight of each word in documents and make up the weight matrix, and a classifier that takes the matrix and trains the model.

1) CountVectorizer: CountVectorizer tokenizes all words in the corpus passed as feature names. Then represents the occurrence of features for each comment in an array. The parameters for each CountVectorizer is adjusted differently to optimize the tokenization.

2) TfidfTransformer: TfidfTransformer takes the results of CountVectorizer and calculates the term frequency and inverse document frequency of each feature. In this project, we use the same imported TfidfTransformer from sklearn class for all pipelines we built.

3) Document tokenization and word lemmatization: During the process of tokenizing the document. Word_tokenizer is initially chosen to break down the sentence. However, this method is inefficient and only breaks sentenses at spaces, leaves many unnecessary signs and punctuation along with words, this greatly lower the accuracy of later fitting and classification. Hence, instead we import the string spliter from regular expression and customized the split deliminator to optimize the prediction. After tokenized the word, we applied lemmatizations for each tokenized word using WordNetLemmatizer imported from the sklearn. Using n-grams during word tokenization process has been considered initially. However, as the extracted feature grows exponentially and make the narrows down the selection of classifiers, we finally decided to use Uni-gram Bag-of-words only for feature extraction.

C. Model selection

The model we chose for prediction mostly came from the sklearn package, more precisely, we have tried almost all the common classifier that suits the dataset from sklearn package, the classifier with accuracy that's high enough has been kept for ensemble and are shown as following:

*1) **Multinomial Naive Bayes**:* Multinomial Naive Bayes Classifier implements the naive Bayes algorithm for multinomially distributed data [2], and is one of the two classic naive Bayes variants used in text classification.

*2) **Decision Tree**:* Decision Trees are a non-parametric supervised learning method used for classification. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

*3) **Logistic Regression**:* Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier [3]. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function.

*4) **SVC**:* Linear Support Vector Classification (SVC) was used for the large dataset. Compared to SVC with parameter kernel='linear', Linear SVC is implemented in terms of liblinear rather than libSVC [4], so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.

*5) **k-Nearest Neighbors**:* k-Nearest Neighbors Classification implements learning based on the nearest neighbors of each query point [4], and the optimal choice of the value k is highly data-dependent: in general, a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

*6) **SGD Classifier**:* Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to discriminative learning of linear classifiers under convex loss functions such as (linear) Support Vector Machines and Logistic Regression [5].

*7) **MLP Classifier**:* Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function by training on a dataset. Given a set of features and a targe, it can learn a non-linear function approximator for either classification or regression [4].MLP Classifier implements an MLP algorithm that trains using Backpropagation.

*8) **Random Forest Classifier**:* A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

D. Model training and adjusting

We found that the models we selected could perform differently depends on the CountVectorizer. For Multinomial Naive Bayes, Decision Tree, Logistic Regression,K-Nearest Neighbors and linear SVC, the model performed better when CountVectorizer is set with LemmaTokenizer as well as $max\_df = 0.5$, $min\_df = 1$. However, for MLP Classifier, SDG Classifier, Random Forest Classifier, the models gave the best accuracy when the CounterVectorizer was set to default.

*1) **Multinomial Naive Bayes**:* The Multinomial Naive Bayes model carried out the best performance with the default setting. No more parameter adjustment was made in this classifier.

*2) **Decision Tree**:* Decision Tree did not have very high accuracy when testing with large datasets. However, the testing showed that the default parameters gave a relatively greater accuracy, so no adjustment was made to the parameters.

*3) **Logistic Regression**:* The multiclass option was adjusted to ”multinomial” to minimize the multinomial loss fit across the entire probability distribution. The solver was set to ”lbfgs” with the maximum iteration number to perform with the best accuracy.

*4) **SVC**:* All the parameters except for the parameter were set to default in linear SVC. The penalty parameter C was adjusted to 0.2 to reach fairly high accuracy.

*5) **k-Nearest Neighbors**:* The number of neighbors could influence the performance of the k-Nearest Neighbors Classifier. After several testing, the number of neighbors was set to 250, and the algorithm used to compute the nearest neighbors was set to ”auto” to achieve the maximum accuracy.

*6) **SGD Classifier**:* The SGD Classifier gave the highest accuracy when all the parameters were set to the default values, so there is no change in the parameter settings.

*7) **MLP Classifier**:* In MLP Classifier, the initial learning rate and the maximum iteration would heavily affect the accuracy performed. Because overtraining can easily occur in the neural network, it is very crucial that the training time is well controlled. With the maximum iteration = 1 and the initial learning rate = 0.004, MLP Classifier was able to perform the best accuracy of $58.10\% \pm 0.85\%$

*8) **Random Forest Classifier**:* The number of trees in the forest and the number of jobs to run in parallel are two crucial parameters that affect the performance of a Random Forest Classifier. According to the testing completed, the classifier gave the highest accuracy when the parameter setting was: n_estimators=100, n_jobs=3

E. Ensemble

After we have the prediction from each classifier, we ensemble and vote for a final prediction using stack method. The vote weight for each classifier included in the ensemble is decided by its individual accuracy, the one with outstanding performances like MLP gets more weight in vote. Diversity is a property of an ensemble of classifiers on some set of data. Diversity tends to be greater when models producing uniformly distributed incorrect decision results [6]. Increasing diversity will certainly have an impact on accuracy boosting, as the speared out wrong predictions are corrected by the right predictions by voting. In this case, all models which are able to produce an accuracy greater than 50% are included as part of the ensemble.

F. Other ways to improve the results

By observing results produced by models trained with different numbers of data, adding more informative data helps improving accuracy in all the case. However, importing more unfiltered data does not always help improving accuracy since the data acquired from elsewhere may not have the same cohesion towards the subreddit. As we tried to add new dataset consists of 2000 new documents for each subreddit for training, the output accuracy surprisingly drops.

Moreover, the result can also be improved by better feature selection techniques, as the tokenizer we used simply filters out common punctuation, there's definitely better feature extraction for natural language exist. For example, stemming the suffix from words. It's worth noticing that eliminate those "unnecessary features" may not always results in improving of the result, for example, eliminating parentheses somehow lower the accuracy for MNB classifier.
Additionally, twerking parameters within a model also has an impact on the result produced(i.e. learning rate and etc.), so to optimize the prediction for each classifier, more effort could be done in adjusting and twerking parameters.

IV. RESULTS

The best result our group obtained with the ensemble is the fifth submission with an accuracy of 58.722%, as shown in Figure. 3. Decision tree and random forest are not included in the ensemble of classifiers due to both insufficient accuracy and long runtime, as Figure. 1 displays the time cost and accuracy of each classifier. Score improvement before the fifth submission is due to adjusting parameters and adding new classifiers in the ensemble. After MLP was added(5th submission), our algorithm hit the ceiling. In the ensemble method, MLP was given triple the weight of Kn and LR models. The decision was made based on the fact that, when running individually, MLP provides significant greater prediction accuracy, as shown in Figure 3. Besides, MNB, SGD and SVM are given double the weight of Kn and LR models since they are close to reaching the TA baseline.

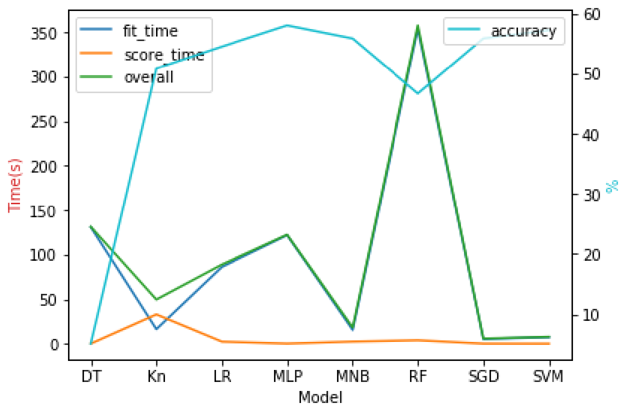

Fig. 1: Run-time and accuracy comparison

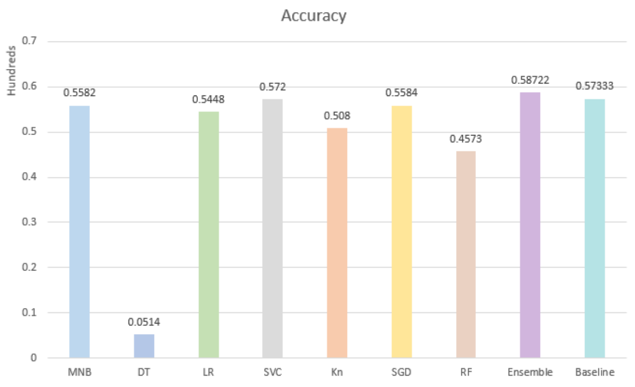

Fig. 2: Models' accuracy and baseline accuracy

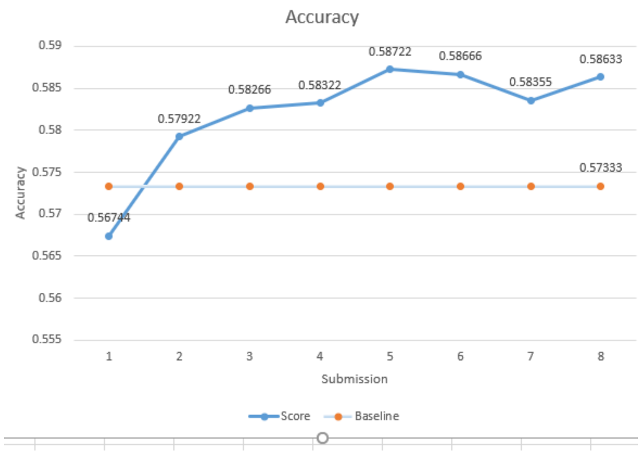

Fig. 3: Submission results improvements

Name	Accuracy[1]	Runtime
Multinomial Naive Bayes	55.82%	18.52s
Decision tree	5.14%	131.23s
Logistic Regression	54.48%	88.82s
Support Vector Machine	57.20%	8.08s
K's Nearest	50.8%	49.74s
Stochastic Gradient Descent	55.84%	6.06s
Random Forest	46.66%	357.36s
Multi-layer Perceptron	58.01%	122.35s

[1] All accuracies refers to 7 folds cross-validation accuracy

V. DISCUSSION AND CONCLUSION

A. *Conclusions and takeaways*

This project allows us to take different approaches to the solution of natural language processing and classification. After investigating several ways of feature extraction and feature classification, we have built up the model with prediction accuracy up to 58.7% in the Kaggle contest. Some takeaways we gain from these projects are shown as follows. If only one model is provided to complete the task with no time constraint, the multi-layer perceptron (MLP) would be the best choice based on its performance. Otherwise, if time is limited, SGD and SVM would be the most suitable choice based on their overall performance. As contrast, decision tree is the least recommended since finding ”good” tokens as nodes in text classification is difficult. The decision tree works badly with a high sparse dimension feature as well, whereas in this the dimension of the feature space is too complex for it to find the best decision boundary. From observing the cross-validation results, we can conclude that the ensemble can improve accuracy by approximately 1.5% higher than the best performance model (i.e. MLP), which is a significant improvement.

B. Further research direction

During this project, only classifiers from sklearn are used, there exist various sources of classifiers out there and many of them are designed specifically for word processing. Sklearn is a general machine learning library built on top of NumPy, with fewer neural network libraries available. Neural network and deep learning might be another approach to classify comments by using PyTorch or TensorFlow. As the class proceeds, students will know more about all different kinds of machine learning approaches. Meanwhile, students will be able to decide the most suitable approach for certain tasks. Whereas in this stage, members in our group make decisions based on exhausting possibilities.

VI. STATEMENT OF CONTRIBUTION

All members have made significant contributions towards this project. The amount of work for each member is described as follows:

Han Zhou : Naive Bayes Implementation, Ensemble, RESpliter, extra-data cleaning, report write up.

Gengyi Sun : Pipeline construction, cross-validation, extra-data cleaning, report write up.

Hao Shu :Multi-layer Perceptron, SGD, SVC, parameter adjust, report write up.

VII. SOURCE CODE

The code used for this project can be found publicly at `https://github.com/JimShu716/COMP551-Group-Projects`.

REFERENCES

[1] Statista, “Combined desktop and mobile visits to reddit.com from february 2018 to july 2019,” *online:* *https://www.statista.com/statistics/443332/reddit-monthly-visitors/*, 2019.

[2] R. N. Jacqueline Gutman, “Text classification of reddit posts,” *online:* *https://jgutman.github.io/assets/SNLP_writeup_gutman_nam.pdf*, 2015.

[3] H. K. Andrew Giel, Jonathan NeCamp, “Cs 229: r/classifier - subreddit text classification,” *online:* *http://cs229.stanford.edu/proj2014/Andrew%20Giel,Jon%20NeCamp,HussainKader,rClassifier.pdf*, 2015.

[4] Scikit-learn, “Supervised learning in scikit-learn,” *online:* *https://scikit-learn.org/stable/supervised_learning.html*.

[5] D. Z. Daniel Poon, Yu Wu, “Reddit recommendation system,” *online:* *http://cs229.stanford.edu/proj2011/PoonWuZhang-RedditRecommendationSystem.pdf*, 2011.

[6] K. W. W. K. Robert E.Banfiel, Lawrence O.Hall, “Ensemble diversity measures and their application to thinning,” *online:* *https://www.sciencedirect.com/science/article/pii/S1566253504000387*, 2004.