Reddit Comment Classification Analysis

Xiaohui Wang, Zhenghua Chen, and Zijun Yu

Abstract - Reddit is one of the most popular discussion websites, which aggregates various comments from different topics. In this work, we aim to analyze text from Reddit by developing a classification model that can predict which of 20 communities a comment belongs to. We preprocessed our data by a series of procedures (tokenizing, lemmatizing and tf-idf method) to exclude unimportant information and convert the selected features into a sparse matrix. To find the optimal classifier for this task, we evaluated the performance of multiple models with K-fold cross validation pipeline. By comparing the accuracy and runtime of all these models comprehensively, we found that the Multinomial Naïve Bayes model achieves a satisfactory score using much less time; however, although time-consuming, the BERT model yields the highest accuracy among all the attempts.

I. INTRODUCTION

REDDIT is a network of communities which brings people who have the same interests together. Every day about 2 million comments are posted on reddit and those comments are attributed to 1.2 million subreddits. Building a model to classify these comments can be useful to filter spams or resort comments that are mistakenly posted on another topic. Our idea is to realize this model by comparing the performance of a few different classification models, which have been used frequently on categorical classification problems.

A. Preliminaries

Lemmatizing is the process of converting a word to its base form. TFIDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Bag-of-worlds model is a way to extract features by representing the text using a vocabulary of known words and a measure of the presence of known words.[1] Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.[2] Bernoulli Naïve Bayes is a naïve bayes method for binary features only. Multinomial naïve bayes uses a multinomial distribution for each of the features, which works well for data which can easily be turned into counts, such as word counts in text. Logistic regression is a discriminative method of probabilistic classifier which models the log-odds with linear function. Decision Tree is a non-parametric supervised learning method which predicts the value of a target variable by learning simple decision rules inferred from the data features. Here we use classification trees where the decision variable is categorical. K-nearest-neighbors method is a non-parametric lazy learning which has the input of the k closest training examples in feature space. SVM is a supervised learning model which creates a line or a hyperplane which separates the data into classes. BERT(Bidirectional Encoder Representation from Transformers) is to apply the bidirectional training of Transformer, a popular attention model developed by researchers at Google AI Language, to language modelling.

B. Task and Findings

In this work our dataset includes 70000 comments, each has one column indicating the subreddit it comes from. The data is assumed to be balanced among all categories. The task of finding the most outstanding model could be divided into four steps: first, we transformed the data by turning categories into numbers and tokenizing; second, we selected features in two ways, bag-of-words or lemmatizing based on what models we used, also applying stop-words and tf-idf method to filter out commonly-seen words; third, we implemented Bernoulli Naïve Bayes model from scratch and used Scikitlearn to build other models; last, we developed a k-fold-cross-validation pipeline to test the performance of these models. After analyzing their performance, we found that that among all these models in a given time, multinomial naïve bayes predicts best when we adjusting the parameter carefully, reaching an accuracy of 0.57499, but if regardless of time cost, BERT achieves the highest accuracy than any other model, which is up to 0.61022.

II. RELATED WORK

Through the years data scientists have never stopped their attempts to sort documents based on their subjects utilizing technical methods. Bo Pang, Lillian Lee and Shivakumar Vaithyanathan[3] have studied on sentiment classification using machine learning techniques. They compared the three models (Naive Bayes, maximum entropy classification, and support vector machines) and showed people the classification accuracies resulting from using only unigrams as features reached up to 82.9 when utilizing the SVM model.

Also Prem Melville, Wojciech Gryc and Richard D. Lawrence[4] published their study on sentiment analysis of blogs. They suggested that knowledge-based approaches, which is a dictionary defining the sentiment-polarity of words and simple linguistic patterns, would need more human effort on annotation of documents. Then they presented a more effective way by combining background lexical knowledge with supervised learning. Their empirical results demonstrate that when provided with even a few training examples, we can combine background lexical information with supervised learning in our framework to produce better results than using a lexicon or the training data separately, as well as an approach to using a lexicon and unlabeled data in a semi-supervised setting. Though the primary focus of this paper is sentiment analysis, the approach developed is applicable to

any text classification task in which some relevant background information is available.
On the aspect of multiple categorical classification, Bharath Sriram, David Fuhry, Engin Demir, Hakan Ferhatosmanoglu and Murat Demirbas[5] have shared their insight for short text classification on Twitter. They proposed to use a small set of domain-specific features extracted from the author's profile and text instead of BOW (Bag-Of-Words model). The 8F approach they developed (extracted 8 features which consist of one nominal (author) and seven binary features) performs significantly better than BOW.

III. Dataset and setup

The training dataset we processed consists of 7000 comments on Reddit with each comment given a subreddit category.

A. Tokenization

To begin with, we divided the contents into pieces, with each being a 'token'. We removed punctuations from the text and lowercased all alphabets.

B. Removal of stop-words

Through nltk, we imported the list of most frequently used words in English, which includes 'the', 'a', 'me' and so on. We made the assumption that stop words have low predictive power, so we removed these tokens from our training data.

C. Lemmatization

With WordNetLemmatizer in nltk, we lemmatized the words based on POS tags of each token, so that we used their root form to form word dictionary (bag of words).

D. Tf-idf

We applied term frequency–inverse document frequency method to select the words that are most important in the collection for training.

E. Vectorization

After cleaning the training data, we converted the cleaned data into a matrix. In this case, we utilized a sparse matrix since the majority of elements in the matrix is 0. Each vector in the sparse matrix tells which tokens in our bag of words are present in a comment.

F. one-hot label

To make our training set validate for neural networks, we needed to transfer all the input data and target data into one-hot labels. One-hot label means having all the data in one matrix, and inside the matrix, there are only 1s and 0s which indicates whether they contain certain elements or not. To implement this, we used a Keras[6] function $keras.utils.to\_categorical$

IV. Proposed approach

A. Bernoulli Naïve Bayes

In this model our features are conditionally independent given the prediction of training set, and the parameters that we use to predict the class take up only 2 values (0 indicates a word occurs in text while 1 indicates a word does not occur). To implement this model, we utilized Bayes Theorem to calculate the probability of A happening (A is a subreddit) given that B has occurred (B is a comment) by the probability of B happening given that A has occurred and probability of A. $P(B|A)$ and $P(A)$ can be calculated by maximizing the log-likelihood function using derivative:

$$\frac{\partial L}{\partial \theta_1} = \sum_{i=1}^{n} \left(\frac{y_i}{\theta_1} - \frac{1 - y_i}{1 - \theta_1} \right)$$

To avoid the situation that words not included in training set will give a 0 probability, we added Laplace smoothing to the maximum likelihood estimator, by adding 1 to the numerator and 2 to the denominator.

B. bag-of-words model

For bag-of-words model in text classification, the text is represented as a collection of its words, disregarding word order and grammar dependencies.

1) Multinomial Naïve Bayes

Multinomial Naïve Bayes is a variation of Naive Bayes classifier. It assumes that every feature is independent of the others in classification. We extract words from the text and apply the Bayes' theorem on probability calculations with word frequencies. This variation considers the number of occurrences of feature in the training set from a class.
We experiment with different choice of hyper-parameter. In Multinomial Naïve Bayes, the alpha parameter is a hyper-parameter, as it defines the model itself. Alpha here is the additive smoothing parameter (for Laplace or Lidstone smoothing); when alpha=0, there is no smoothing applied. It can be problematic when a frequency-based probability is zero, thus we apply smoothing technique and pick one that yields the best accuracy model.

2) Logistic Regression

Logistic regression is a discriminative method that predicts a categorical target using the logistic function. There are several types of logistic regression including binary logistic regression, multinomial logistic regression and ordinal logistic regression.
For this task, we apply multinomial logistic regression for predicting multiple categories without ordering. We preprocessed data to remove noise as logistic regression assumes that there is no outliers in training data. With our experiment, we apply L2 regularization with 'lbfgs' solver to avoid overfitting.

3) Support Vector Machine(SVM)

An SVC/SVM model is a representation of data points in space, where points of separate categories are divided by a clear gap. The support vector machine takes in a kernel parameter for mapping data points and for this experiment we specify kernel type to be linear. Support vector clustering

is an unsupervised learning approach to categorize unlabeled data and it is a widely used clustering algorithm.

4) Decision Tree

A decision tree is used to classify record by assigning it to most likely category based on a tree representation. Decision tree is one of the fastest ways to perform classification. At the beginning of the algorithm, decision tree assumes that the whole training set is considered as the root, and that the training data is distributed recursively on the basis of attribute values.

5) K-Nearest Neighbor

K-nearest neighbor algorithm (KNN) is a non-parametric lazy learning algorithm that can be used for text classification. Given an input, the algorithm finds the K nearest matches in training data to predict the category.

C. Deep learning methods

Applying bag-of-words model to our data ignores the actual structure of our sentences, which is a big loss of information. Using Neural Networks to train and predict our data could be a solution to this. There are two popular ways to implement neural networks, TensorFlow and Theano. But in this project I used Keras[6], which is an easy-to-start, powerful API using TensorFlow and Theano as backend.

1) Convolutional neural network

In a text processing task, the same word sometimes represents totally different meanings among two sentences. This situation demonstrates noise in bag-of-word models, especially when we lemmatize words in our sets.
To solve this problem, a convolution neural network(CNN) could be useful. CNNs are regularized versions of a fully-connected, multilayer neural network. Since we are dealing with texts, and 1-dimension CNN is enough.
There are two ways to construct a TextCNN[7]. First is a 'deep' neural network, which means we have three fully connected layers series, with each layer's output is the input of the next layer. The second one is a set of layers running in parallel, where each convolutional layer going through different numbers of words each time then we combine their output together.

2) Recurrent neural network

When in a sentiment test, a sentence with positive in the first half and negative in the second half could cause big trouble. A recurrent neural network(RNN) could be really helpful in this case. RNN will go through the whole sentence which means it can actually memorize the words it saw before. Two famous RNN for NLP are LSTM(Long short-term memory) and GRU(Gated recurrent unit)
There are several ways to implement an RNN.[8]

i Using a LSTM directly
ii Using a bi-LSTM directly
iii First use CNN to analyze the data, then output to a LSTM layer.
iv Use a CNN and a LSTM in parallel then combine the output.

3) Design our own network

Since Keras is a powerful tool, we decide to design our own neural network. We observed three best performing neural networks and tried to combined them together. First, is three models in parallel: a 'deep' CNN, a CNN-LSTM, a bi-LSTM model. Then we used a concatenate layer to merge the output. The structural graph of the model is in the appendix.

D. Bert–Transfer learning

Sometimes the training set provided cannot assure us a perfect output because it is not big enough for our model to get enough information. BERT (Bidirectional Encoder Representations from Transformers)[9] is a deep learning model developed by Google. Which is pre-trained and is using a fine-tuning technique when we are training this model using our dataset.
We found a perfect wrapper for BERT that makes BERT reachable, ktrain[10]. Within a few lines of code, ktrain can actually help us pre-process the dataset and start the training process.

V. Results

Model validation pipeline: Cross-validation

When evaluating a trained model, we want to hold out part of the available data as a test set. Predicting on the data that has been trained is problematic and would cause overfitting. Thus, we adopted a pipeline that produced less-biased evaluation results. For our experiments and results, we implemented k-fold cross validation to test effectiveness of our model. For this k-fold cross validation, the given dataset is split into K folds, where each fold is used as a testing set at some point.

A. Bernoulli Naive Bayes

Before utilizing tf-idf to select more meaningful features, we ran 5-fold-cross validation on the training dataset which only excluded the stop-words with nltk. It gave an average accuracy of 0.5175. Then we implemented lemmatizing and tf-idf to see if we could gain improvements. This time we got an average validation accuracy of 0.5219, which is an obvious increase. Thus we validated that for Bernoulli Naïve Bayes, stop-words are not efficient for filtering out data; by removing lemmatized data with low tf-idf, the model would achieve better results. However, Bernoulli Naïve Bayes is still not the most outstanding model for predicting subreddits of comments in this problem, but it takes only around 0.357 seconds to predict outcomes of 10000 data based on 60000 training data.

B. bag-of-words model

1) Multinomial Naïve Bayes

As a result of the experiment on Multinomial Naïve Bayes model, alpha = 0.32 yields the best-performance model. The average runtime for prediction, using training set size of 60000 records and validation set size of 10000 records, was 0.3071 seconds.

TABLE I
ALPHA PARAMETER AND ACCURACIES

alpha	0.10	0.20	0.30	0.32	0.34
accuracy	0.5723	0.5755	0.5766	0.5772	0.5768
alpha	0.40	0.50	0.60	0.70	0.80
accuracy	0.5756	0.575	0.5736	0.5728	0.5714

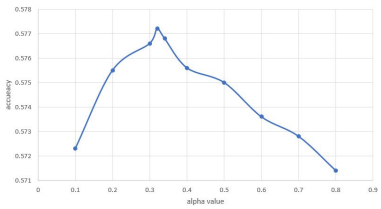

Fig. 1. Experiment of hyperparameter alpha on Multinomial Naïve Bayes model

We evaluated the model performance with K-fold cross validation. We recorded the runtime, mean value and standard deviation of accuracies with respect to each chosen K. As a result, the unbiased accuracy for our Multinomial Naïve Bayes model is 0.569.

TABLE II
MODEL PERFORMANCE WITH COMPARISON TO K VALUE IN CROSS VALIDATION

	K=5	K=8	K=10	K=15	K=20
Accuracy($\pm$0.01)	0.563	0.567	0.568	0.569	0.569
Runtime(s)	1.636	2.578	3.278	4.887	6.614

2) Multinomial Logistic Regression

We have run the experiment with the Multinomial Logistic Regression classifier. We set max_iter=4000 and solver='lbfgs' with L2 regularization applied by default.
With K-fold cross validation where K=4, the mean accuracy is 0.539 (+/- 0.01), with each iteration giving the accuracy [0.5366 0.5433 0.5371 0.5397].
The training took time 4:34.692869, meaning the process took roughly 4.5 minutes.

3) Support Vector Machine(SVM)

We have run the experiment with the SVM classifier.
With K-fold cross validation where K=3, the mean accuracy is 0.392 (+/- 0.00), with each iteration giving the accuracy [0.3938 0.3924 0.3897].
The training took time 13:06.075, meaning the process took roughly 13 minutes.

4) Decision Tree

We have run the experiment with the Decision Tree classifier.
With K-fold cross validation where K=4, the mean accuracy is 0.321 (+/- 0.01), with each iteration giving the accuracy [0.3169 0.3247 0.3192 0.3223].
The training took time 6:42.757, meaning the process took nearly 7 minutes.

C. Deep learning methods

To test the deep learning model, we split the training set and test set into 9:1. And also set the length of sentence to 150, the batch size to 100 to compare within different models.

TABLE III
ALPHA PARAMETER AND ACCURACIES

Model name	time/step	1-epochs	n-epochs	highest
CNN	655us	37.94	44.52	44.52
TextCNN	681us	19.81	39.76	39.76
LSTM	3ms	36.34	overfit	48.61
BiGRU	10ms	50.89	overfit	54.28
CNNRNN	7ms	52.06	overfit	55.61
CNNLSTM	659us	16.04	overfit	50.13
Trinity	13ms	65.74	53.15	53.15

D. Bert–Transfer learning

When running the Bert model, we set the size of validation to be 1.5% because with more training set, we are expected to have a better estimation. The drawback to this is that the validation result can't perfectly match the real test result.
For other parameters, we set the maximum length of each sentence to be 250. Batch size is 6 due to the size of the memory of our GPU. Number of epochs 4 and updating parameter $2e-5$ to help us better converge.
Among the variables, we chose the number of max features to be our testing parameter. The number of max features stands for the size of word-bank, our assumption is when the dataset is not large enough, shrinking the word-bank could be helpful. But with BERT, a training set of size 70k is large enough, so as we increase the size of word-bank, the outcome should be better. And here is the testing result:
Here we can observe that as we include more words into our dictionary, the accuracy increases. But in the appendix, there are two screenshots showing the validation accuracy where as the word-bank increases, the accuracy decreases. The reason for this is that our validation set is small so the amount of bias inside the validation set.

TABLE IV
NUMBER OF FEATURES & KAGGLE ACCURACIES

max-feature	35000	36000	40000
Kaggle accuracy	0.60511	0.60900	0.61022

VI. DISCUSSION AND CONCLUSION

In this work we investigated various supervised machine learning models such as Bernoulli Naïve Bayes, Multinomial Naïve Bayes, Logistic regression, Decision trees, knn, SVM and a state-of-the-arts method BERT and how they perform in classifying texts into multiple topics. In preprocessing dataset and selecting meaningful features, we utilized a combination of techniques such as tokenizing, lemmatizing and tf-idf weighting. We found that lemmatizing can be an especially helpful tool to filter out useless data. Instead of tokenizing text into single words and removing stop-words, lemmatizing and selecting features according to tf-idf weighting is more effective. Lemmatization transforms a word to its dictionary form with consideration of the context. Applying this we can ignore the grammar and make the words context-free,

thus reducing the feature size. Also removing unimportant information based on their tf-idf weights is more efficient than removing stop-words, for some of the stop-words may play a greater role than we imagined in recognizing subreddits.

Future work

Here we present three ways to improve the work and for future studies:

1) When pre-processing the sentences, one can try to deal with synonym, which means for a set of synonym, we replace them with the same word(or number). By applying this, we can highly increase the efficiency of each word appears in the text. This is a kind of pre-training process, which we are actually adding the amount of information into our dataset, therefore the accuracy could highly increase while we are having a relatively small dataset. But if the training set is large enough, it is not recommended to use this approach.
2) When we are testing the neural networks, we are always suffering from overfitting. To deal with this problem, we recommend to decrease the learning rate and maybe add regularization to each layer.
3) In the result of BERT model, we can see that the accuracy of some subreddits have been up to 80 percent. Our idea is that for each validation result of BERT model or other models, if some of their subclass accuracy is high, we collect those results and build a fully-connected layer to concatenate the results to achieve a higher accuracy.

Statement of Contributions

All team members have made certain contributions to the project, works were equally divided. Detailed division of work is as follows:

- Xiaohui Wang: Dataset acquiring, pre-processing, searching for the best model, writeup contribution.
- Zhenghua Chen: Related work researching, report writeup
- Zijun Yu: Writing and training for divergent models, writeup contribution

Appendix

Three figures:

	precision	recall	f1-score	support
hockey	0.69	0.58	0.63	62
nba	0.69	0.70	0.69	60
leagueoflegends	0.63	0.75	0.69	44
soccer	0.81	0.69	0.75	55
funny	0.22	0.31	0.26	51
movies	0.65	0.80	0.72	46
anime	0.79	0.68	0.73	44
Overwatch	0.68	0.68	0.68	44
trees	0.56	0.63	0.59	46
GlobalOffensive	0.66	0.56	0.61	62
nfl	0.63	0.64	0.64	53
AskReddit	0.36	0.36	0.36	59
gameofthrones	0.80	0.67	0.73	52
conspiracy	0.46	0.52	0.49	46
worldnews	0.42	0.49	0.45	49
wow	0.84	0.69	0.76	54
europe	0.63	0.57	0.60	58
canada	0.53	0.49	0.51	59
Music	0.71	0.78	0.74	45
baseball	0.70	0.64	0.67	61
accuracy			0.61	1050
macro avg	0.62	0.61	0.61	1050

(a) Kaggle public score 60.9

	precision	recall	f1-score	support
hockey	0.78	0.69	0.73	61
nba	0.54	0.62	0.58	45
leagueoflegends	0.69	0.56	0.62	43
soccer	0.70	0.71	0.70	52
funny	0.38	0.47	0.42	64
movies	0.61	0.61	0.61	56
anime	0.70	0.67	0.69	55
Overwatch	0.72	0.74	0.73	58
trees	0.57	0.67	0.62	46
GlobalOffensive	0.79	0.72	0.76	58
nfl	0.66	0.68	0.67	56
AskReddit	0.34	0.38	0.36	45
gameofthrones	0.72	0.65	0.68	51
conspiracy	0.44	0.50	0.47	44
worldnews	0.35	0.37	0.36	41
wow	0.89	0.80	0.84	50
europe	0.70	0.65	0.68	69
canada	0.76	0.61	0.67	61
Music	0.67	0.67	0.67	46
baseball	0.65	0.67	0.66	49
accuracy			0.63	1050
macro avg	0.63	0.62	0.63	1050

(b) Kaggle public score 60.511

Fig. 2. The validation accuracy of Bert

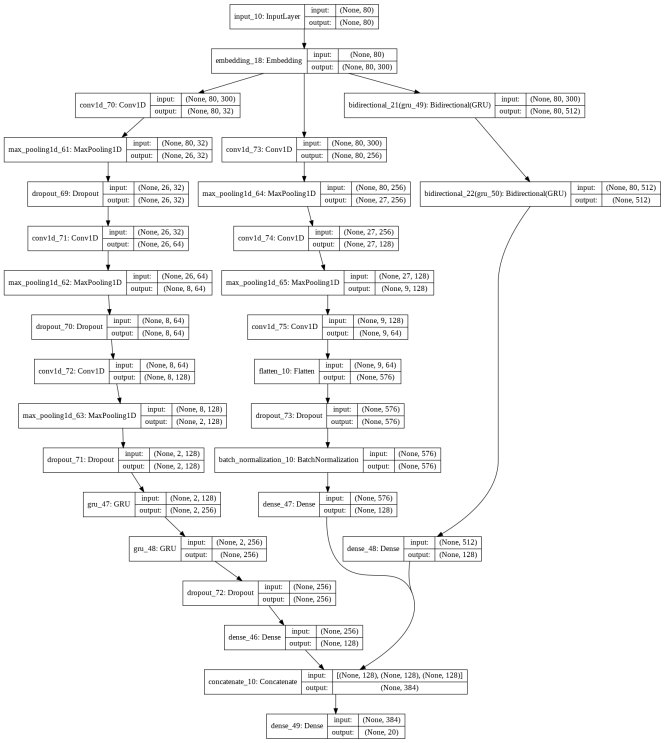

Fig. 3. The structural graph of a neural network We designed

References

[1] J. Brownlee, "A gentle introduction to the bag-of-words model." https://machinelearningmastery.com/gentle-introduction-bag-words-model/, oct 2017.

[2] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[3] 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*, EMNLP '02, (Stroudsburg, PA, USA), pp. 79–86, Association for Computational Linguistics, 2002.

[4] P. Melville, W. Gryc, and R. D. Lawrence, "Sentiment analysis of blogs by combining lexical knowledge with text classification," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, (New York, NY, USA), pp. 1275–1284, ACM, 2009.

[5] B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas, "Short text classification in twitter to improve information filtering," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, (New York, NY, USA), pp. 841–842, ACM, 2010.

[6] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[7] Asia-Lee, "Textcnnwenbenfenlei(kerasshixian)." https://blog.csdn.net/asialee_bird/article/details/88813385, mar 2019.

[8] wang-yi lei, "Keraszhiwenbenfenleishixian." https://zhuanlan.zhihu.com/p/29201491, sep 2017.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[10] A. S. Maiya, "ktrain." https://github.com/amaiya/ktrain, 2019.