Reddit Text Classification

Code available at https://github.com/tonamuhang/Reddit-Comment-Classification

***Abstract*—In this project, we considered the problem of classifying the comments into 20 difference subreddits by applying classifiers, including Logistic Regression(LR), Decision Trees, Support Vector Machines, Bernoulli Naive Bayes', and Multinomial Naive Bayes'. We also experimented with Recurrent Neural Network with the Keras library run on the TensorFlow backend. We investigated how changing the parameters and preprocessing texts can affect the performances of the classifiers. In the end, with the help of Gridsearch for hyperparameter tuning, and model validation pipeline, we found that among all of the classifiers, Support Vector Machines performs the best, while the Decision trees method performs the worst in our environment.**

I. Introduction

The common classifiers for text categorical classification are Logistic Regression, Bernoulli Naive Bayes, Multinomial Naive Bayes, Decision Trees, Support Vector Machines, and Recurrent Neural Network. In this project, we compared the abilities as 20-class classifiers between the above models by training them on the Reddit comments dataset.

The Reddit comment dataset contains 70000 comments that belong to 20 different subreddits. The subreddits class distribution is balanced as there are 3500 comments of each subreddit. The comments are randomly picked, and the formality of the texts can not be guaranteed. Thus, data preprocessing, for example, converting text to lowercase, removing meaningless URLs, and numbers, is required. For this specific dataset, we extracted the critical information such as the title of the URLs, since it would be a waste to just throw out all the websites that might contain useful intels to help us classify the subreddit class. We also experimented with n-grams and tested its effect on different models.

We compared the performances of the models under different circumstances. By applying a model validation pipeline with k-fold cross-validation and gridsearch, we were able to find the best parameters for the specific dataset with ease.

The report is organized as follows. Section II introduces the related work to the text classification problem. Section III describes the dataset and how it's loaded, analyzed, and cleaned. Section IV explains our approach of the different models, and the features we designed. Section V details the results obtained by running the experiments on different models. Finally, Section VI briefly discusses the key findings and concludes the whole project.

II. Related Work

In terms of text classification in machine learning, the performance of a classifier between can be measured by a variety of methods such as metrics, accuracy, precision, and recall.

Parametric and Non-parametric models are both implemented in this project. k-nearest neighbour and rule induction are two non-parametric models that are not implemented in this project but can solve the same problem.

Estimating conditional probability is essential for a Naive Bayes model. In order to obtain a higher accuracy of conditional probability, metaheuristics methods such as Genetic Algorithms and Differential Evolution can be used for the estimation process.

Since the linear support vector machine provided the best estimations in this project. Customizing kernels is a better method to improve the performance since it contains the background details for the text. One example is named Class Meaning Kernel, which addresses class-based meaning values of terms for the purpose of smoothing. (Goudjil, Koudil, Bedda & Ghoggali, 2016)

III. Dataset

A. Dataset description

In this project, a dataset taken from the website Reddit (https://www.reddit. com/) is provided. The raw dataset contains 70, 000 comments, and each comment belongs to one of twenty unique subreddits. Also, the dataset is balanced.

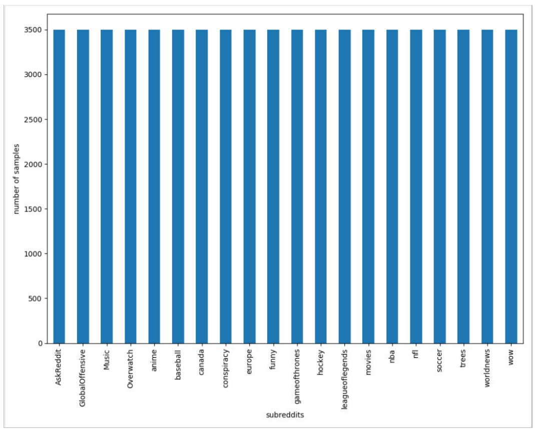


Fig. 2 Class Distribution

B. Data loading and cleaning

We used the built-in function from pandas to load our dataset. In order to make use of the text data and improve the accuracy of our models, the following process steps were performed:

- Load data: when the data was loaded, we also removed the numbers, redundant white spaces, punctuation, and converted the comments to lowercase.
- Text tokenization: We split the comments into smaller pieces, i.e. words, as tokens.
- Text Lemmatization: We converted each word into its root form, assuming that they are all nouns.
- Stopwords deletion: We removed all the unnecessary words contained in the nltk package, along with our custom stop words, such as month words, and quantity words.
- TF-IDF Encoding: We encoded the text data into numerical values, and adjusted their weights according to the frequency they appear.
- Data Normalization

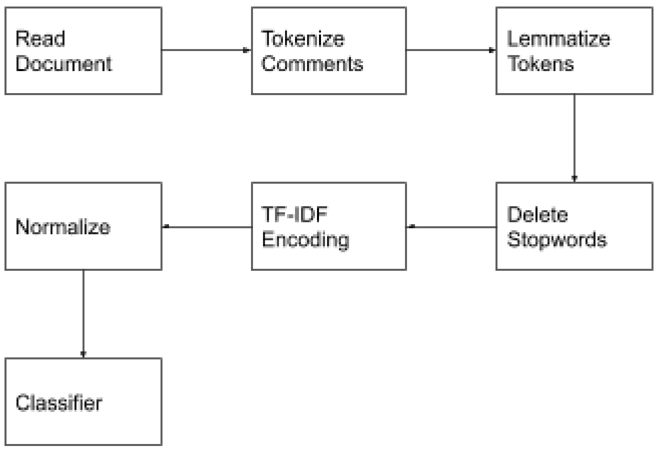


Fig. 2 Text Preprocessing Pipeline

IV. Proposed Approach

Bernoulli Naive Bayes model implementation

a. Technical Approach:

Our first approach was to find appropriate data structures in order to fit all datasets correctly. Meanwhile, the data structure applied needs to be convenient for retrieving and manipulating data as well. After reading documents of different data structures in python, we decided to use Numpy array and Lists as our data structures for this project.

Data preprocess

First of all, we need to preprocess raw data from the Reddit comment dataset into an organized form so it can be used to compute conditional probabilities, likelihoods, and other criterias for the learning of data. In order to accomplish that goal, we preprocessed the reddit_train.csv using functions in text_preprocess.py file, which reads from a CSV file, lemmatize and tokenize the initial file into a element-comparable format. Meanwhile, it removes stopwords, single characters, digits, etc. which are regarded as unrelated to the topics of the comments. We initialized a variable called keywords, which stores the preprocessed raw data. We extracted the comments from the keywords variable and deleted the useless part of the data. Then we retrieved the corresponding labels of the preprocessed training data.

Since comments and labels are both Data Frames at this moment, we imported the Pandas Package, which includes a unique method(iloc[start : end].tolist) that retrieves rows from Data Frames and turns them into lists. For the same purpose, we performed the tokenization function on the keywords variable. As a result, the keywords variable turned into a two dimensional list where each of its elements is a list of a comment, whereas labels turn into a list of labels.

Finally, all the preparation work is completed, and both the comments and labels are transferred from a CSV file into two python lists. The datasets are now ready to be analyzed and learned by the program.

Define the Vocabulary and Feature Vectors

After all the preparation work has been settled, the second step was to define the vocabulary vector, the element of which will be analyzed as features in the future steps. By concatenating all the lists in the keywords variable together, we obtained a one dimensional matrix as the vocabulary vector. It contains all the essential vocabulary in the document.

After that, we defined the feature vectors of the document. The number of words in the vocabulary vector defines the dimension of the feature vectors. By comparing each comment(lists in keywords variable) with the vocabulary vector, we constructed a new two dimensional matrix (size=number of classes * length of the vocabulary vector), which is used to mark the existence of each essential vocabulary in each comment. If the vocabulary exists in a certain comment, we marked 1 in its corresponding position in the new matrix and 0 otherwise. We returned the new binary matrix for the prediction method.

Fit Training Data

We fitted the training data into the Bernoulli Naive Bayes model by computing the individual word likelihoods

P^(wt | Ck) = nk(wt)/ Nk

And then we multiplied all individual words likelihoods together in each comments to obtain the comment likelihoods

P(D |Ck) = P(b|Ck) = Y |V| t=1 [bt P(wt |Ck) + (1−bt)(1−P(wt |Ck))] .

After computing and multiplying P^(wt | Ck) , we computed the relative frequency of comments of class K based on the formula

P^(Ck) = Nk/ N (aka. prior probability).

Finally, we multiplied each of the comment likelihoods with each of the prior probabilities to obtain the posterior probabilities which is a two dimensional matrix. We then picked the class, which result in the greatest posterior probability for each comment as the result of the prediction.

b: Motivation

The Naive Bayes' model uses the information in the sample data to estimate the posterior probability of each class, given the word vector. We may use this information for our task of classifying the subject of the comments.

c: Pros and Cons

By using Naive Bayes', the interactions between each feature are ignored. Instead of computing the actual probabilities, relative probabilities are used. However, it reduces the time of computation.

Support Vector Machines

a. ***Technical Approach***
The Support Vector Machines is a classifier that is proposed by Vladimir N. Vapnik and Alexey Ya. It finds a maximum margin that separates between two classes of data. In our task, since we have 20 data classes, we will have to train multiple SVM's, and they will act as one-vs-all classifiers.

b. ***Motivation***
We want to have 20 binary SVM's that are trained independently. Each of the SVM's acts as a one-vs-all classifier, and is responsible for every subreddit class. SVM is a good choice for high dimensional data. In addition, there is a regularization parameter in SVM available for adjusting.

c. ***Pros and cons***
With the regularization parameter, we can effectively avoid the over-fitting issue. We can also avoid the need for importing domain-specific knowledge to our model.
However, it's hard to select the best kernel for the problem.

Decision Tree

a. ***Technical Approach***
The Decision Tree is a model that uses a tree like structure of decisions and their consequences. It works in a sequence and tests a decision against a given threshold value. Different subsets of the data are taken in the process. A terminal node is a final subset. To predict the outcome of each terminal node, the average outcome of the training data in this node is used.

b. ***Motivation***
Models like logistic regression fail in situations where the features are correlated. The Decision Tree model splits the data multiple times, and different subsets of data are tested to prevent this problem.

c. ***Pros and cons***
The decision tree model is very easy to understand and interpret thanks to its tree-like structure. On the contrary, it's very slow and is extremely vulnerable to the variance of our input data.

Logistic Regression

a. ***Technical Approach***
The Logistic Regression model is a supervised learning model. Logistic regression requires a training dataset to learn the parameters and applies the gradient descent algorithm to optimize the learning parameters. By using the sigmoid function, a decision boundary will be set in order to achieve classification.

b. ***Motivation***
Logistic regression is a linear model that is good for categorical predictions. There are 70, 000 comments in our data, and linear models perform well on large and high-dimensional data. Thus, logistic regression is a good choice considering the size of data.

c. ***Pros and cons***
Logistic regression does not only solve classification problem, but also provides probabilities. It is easy to be implemented and not time-consuming. However, logistic regression may sometimes lead to overfitting or underfitting. The accuracy of classification is usually not so high. When key features are not complete, its performance will degrade.

Recurrent Neural Network

a. ***Technical Approach***
We built an RNN that has three layers. The first layer is an embedded layer that takes vectors that represent each word. The second layer is an LSTM layer with 100 memory units. The final output layer creates 20 output values that correspond to each subreddit.

b. ***Motivation***
We would like to simulate a human brain for analyzing and classifying each sentence. The Recurrent neural network remembers its past work. It is a good technique to improve future decisions of labels for given comments.

c. ***Pros and cons***
RNN is easy to use and simulates a human brain. Nevertheless, it requires a large amount of training data to predict a class correctly. The disadvantage can be fixed by introducing a pre-trained word embedding, which wasn't allowed for this task.

V. Results

With the help of parameter tuning, we achieved an accuracy of 0.57811 on the Kaggle test set using Support Vector Machines.

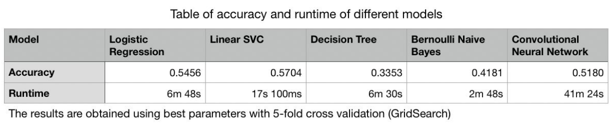


Table of accuracy and runtime of different models

Model	Logistic Regression	Linear SVC	Decision Tree	Bernoulli Naive Bayes	Convolutional Neural Network
Accuracy	0.5456	0.5704	0.3353	0.4181	0.5180
Runtime	6m 48s	17s 100ms	6m 30s	2m 48s	41m 24s

The results are obtained using best parameters with 5-fold cross validation (GridSearch)

Figure 3. Accuracy and runtime table

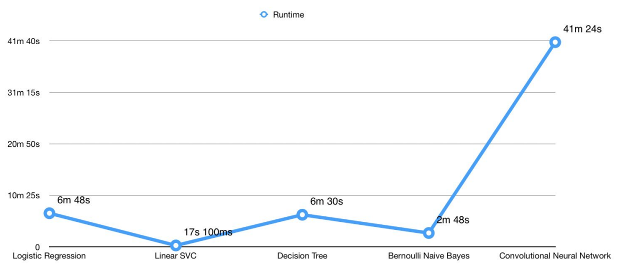


Figure 4. Comparison of runtimes between models

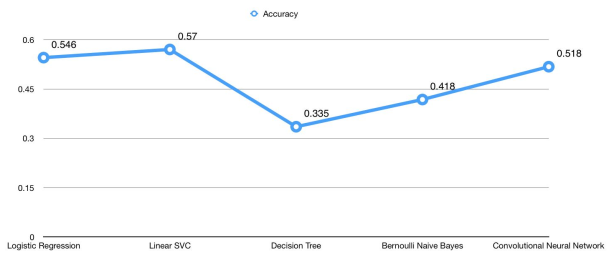
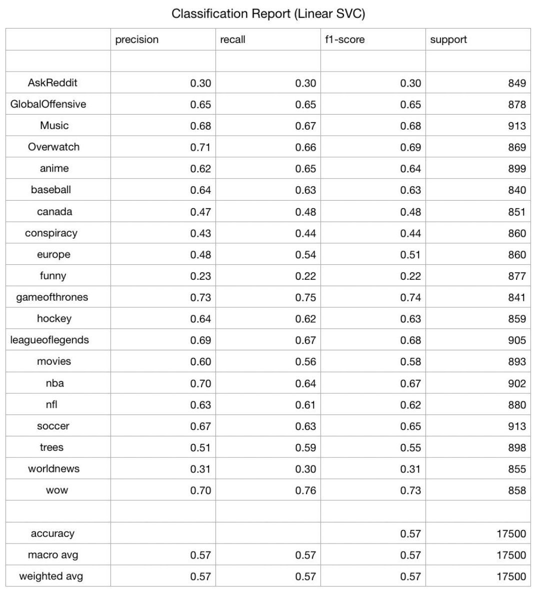


Figure 5. Comparison of accuracies between models

Here we have the classification report for our Support Vector Machine model. We can see that the model performs the best for the Overwatch, gameofthrones, nba, and wow subreddits and performs the worst for the AskReddit, funny, and worldnews subreddits.

The reason behind the success and the failure of the subreddit predictions might be due to correlations between the words and the subreddits. For example, it's almost impossible to tell to which subreddit the sentence "How to cook the perfect spaghetti" belongs.


Classification Report (Linear SVC)

	precision	recall	f1-score	support
AskReddit	0.30	0.30	0.30	849
GlobalOffensive	0.65	0.65	0.65	878
Music	0.68	0.67	0.68	913
Overwatch	0.71	0.66	0.69	869
anime	0.62	0.65	0.64	899
baseball	0.64	0.63	0.63	840
canada	0.47	0.48	0.48	851
conspiracy	0.43	0.44	0.44	860
europe	0.48	0.54	0.51	860
funny	0.23	0.22	0.22	877
gameofthrones	0.73	0.75	0.74	841
hockey	0.64	0.62	0.63	859
leagueoflegends	0.69	0.67	0.68	905
movies	0.60	0.56	0.58	893
nba	0.70	0.64	0.67	902
nfl	0.63	0.61	0.62	880
soccer	0.67	0.63	0.65	913
trees	0.51	0.59	0.55	898
worldnews	0.31	0.30	0.31	855
wow	0.70	0.76	0.73	858
accuracy			0.57	17500
macro avg	0.57	0.57	0.57	17500
weighted avg	0.57	0.57	0.57	17500

Figure 6. Classification report of SVM

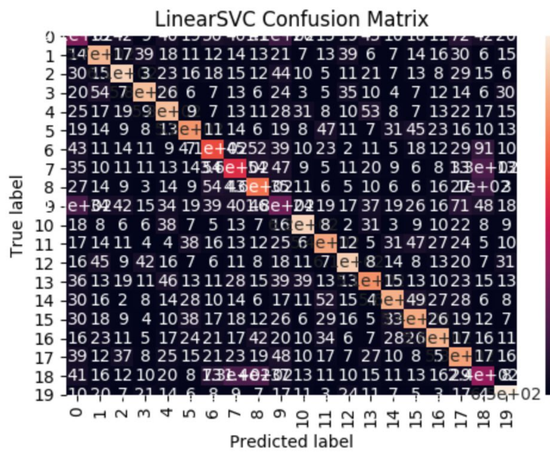


Figure 7. Confusion matrix of SVM

VI. Discussion and Conclusion

A. Key takeaways

1. The Support Vector Machines performs best against the Reddit comment dataset, while the Decision Trees method performs the worst.
2. Preprocessing data gives negative improvements to the SVM, while giving a considerable amount of improvement to the Naive Bayes' model.

B. Future investigation

There are a few improvements that can be added to our experiments but were not implemented due to time constraints. We will briefly discuss them below.

- Tuning Recurrent Neural Network: Due to limited time, we only built a basic model based on a simple online tutorial. This model may not suit our problem very well, and a specific model for the specific problem is required.
- Combine text sentiment classification: As previously mentioned in the report, it's sometimes impossible to classify based only on the words.

With sentiment classification, we can gain extra information from the sentence, and therefore predict sentences like “How to cook the perfect spaghetti”.

VII. Statement of Contributions

References

[1] https://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn07-notes-nup.pdf

[2] https://link.springer.com/article/10.1007%2Fs11633-015-0912-z

[3] https://towardsdatascience.com/multi-class-text-classification-with-lstm-1590bee1bd17