COMP4107 Neural Networks Final Project

Effects of Text Pre-processing Methods on Text-Based Sentiment Analysis Problems

Zijian Zhen (Kirk)
101087006
zijianzhen@cmail.carleton.ca

I. Abstract

Text-based Sentiment Analysis techniques are widely used in many aspects of people's lives. This project aims to explore the influence of text pre-processing techniques when dealing with the text-based sentiment analysis problem, basing on an IMDb movie review text dataset. The performance of 3 classification models trained with and without several Text pre-processing methods has been explored via a 5-fold cross-validation method.

The result of this study shows that an appropriate choice of text pre-processing strategy is helpful to improve the performance of different models.

II. Introduction

In most of the machine learning problems, data representation plays an important role in pursuing a good performance. Especially in text-based classification problems, how to transform the text into numerical representation always has a huge impact on predicting results. This paper focuses on binary sentiment analysis based on a movie review dataset(i.e. to determine whether a movie review document is positive or negative) and aims to compare the performance of 3 different classifiers with several pre-processing methods when they are applied to solve a sentiment analysis problem. The influence of text-cleaning, TF-IDF, and N-gram on the performance of different models would be investigated. NLTK Lemmatizer, text Vectorizer with TF-IDF and N-Gram, and a feature selection strategy based on ANOVA F-value would be applied in the experiments. A 5-Fold cross-validation method would be introduced to compare the performance of the 3 models.

The result of my experiment shows that text pre-processing is necessary for ease and acceleration of the learning process and is capable to optimize the predicting performance of different classifiers.

III. Background

A. Raw Data

The Large Movie Review Dataset [1] is composed of a set of movie reviews, with the text of 50000 reviews as input data, and the corresponding sentiment of the review (positive or negative) as output label. The distribution of label are fairly equal for positive and negative (25000 positive samples and 25000 negative samples).

In this study, in each fold of 5-Fold cross-validation, 80% of the samples would be randomly chosen as the training set, and the rest 20% of the samples would be chosen as the testing set.

B. Evaluation of Performances

The following evaluation metrics would be applied to measure the performance of sentiment analysis models in our study:

$$\text{Recall} = \frac{\text{TN}}{\text{TN+FP}} \times 100\% \qquad \text{F1 Score} = \frac{2}{\text{Recall}^{-1} + \text{Precision}^{-1}} \times 100\%$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \times 100\% \qquad \text{Accuracy} = \frac{\text{TP+TN}}{\text{TP + FP + FN + TN}} \times 100\%$$

Where TP, FP, TN, FN are defined as:

- **TP**: Sample of class 1 predicted as class 1.
- **FP**: Sample of class 0 predicted as class 1.
- **TN**: Sample of class 0 predicted as class 0.
- **FN**: Sample of class 1 predicted as class 0.

Besides all of the above evaluation metrics, the time cost is also an important indicator to evaluate the efficiency. So, the time cost would be also investigated in this study.

IV. Related Work

A. N-gram

According to Chapter 3 in Speech and Language Processing (Jurafsky Martin, 2014) [3], N-gram uses sequences of words instead of single words as features when dealing with raw documents. In language processing, when some words appear next to each other, they may have different meanings in comparison with they appear individually. And the N-gram is capable to capture the relationship of adjacent words. Table I shows a simple example of how n-gram works.

TABLE I
An example of N-gram

Methods	**Features**
Raw document	Today is a sunny day
Uni-gram	“Today”, “is”, “a”, “sunny”, “day”
Bi-gram	“Today_is”, “is_a”, “a_sunny”, “sunny_day”
Tri-gram	“Today_is_a”, “is_a_sunny”, “a_sunny_day”
4-gram	“Today_is_a_sunny”, “is_a_sunny_day”

B. TF-IDF

TF-IDF, proposed by Salton & Buckley(1988) [4], is commonly applied as a term weighting method in text processing. Each term (words or n-gram) in the raw document would be encoded into its corresponding TF-IDF value. TF means Term Frequency, which measures the frequency of a term in the document; IDF means Inverse Document Frequency, which measures the volume of information provided by the term. And TF-IDF is the product of TF and IDF. Here is the formula for TF-IDF used in this project:

$$
\begin{aligned}
\text{TF}(t,d) &= \frac{\text{total number of term } t \text{ in } d}{\text{total number of words in } d} \\
\text{IDF}(t,C) &= \log \frac{|C|}{|\{d \in C : t \in d\}|} \\
\text{TF-IDF}(t,d,C) &= \text{TF}(t,d) \times \text{IDF}(t,C)
\end{aligned}
$$

where C is the corpus, d is the documents in the corpus, and t is the terms in the documents.

V. Methodology

A. Pre-Processing

The performance of a machine learning model strongly depends on data representation. The raw data in this study is not a good representation for machine learning models since most of the models do not accept texts as input. In this situation, how to transform the raw text into a machine-readable form can significantly influence the performance of a given classifier in text sentiment analysis problems.

1) ***Text Cleaning***
Digits and punctuation would be removed from the original text since they carry unnecessary information about sentiment. Because the upper/lower case of words is not important for the purpose of sentiment analysis, all text of reviews would be transformed into lower case. After that, NLTK Lemmatizer would be applied to lemmatize the texts, for example, from ”cars” to ”car”, ”did” to ”do”. All the above process would be done by a regular expression and the NLTK library [2].

2) ***Feature Extraction (TF-IDF and N-gram)***
Tokenizer in Keras would be applied as a text tokenization method. In this project, some models are not able to accept features without frequency vectorization. For example, the CNN models are capable to capture the relation between words next to each other, so frequency or tifdf vectorization is not suitable in these models. In this way, simply tokenization of text is sufficient to optimize the performance of these models.
TfidfVectorizer in Scikit-Learn was applied in both tokenization and vectorization parts. TfidfVectorizer scan words and n-gram words in the raw text and transform them into a numerical vector according to TF-IDF statistic. In this project, the set of “n” would be [1,2,3] or [1], which means that uni-gram, bi-gram, and tri-gram, or only uni-gram would be counted when vectorization.

3) ***Feature Selection***
Text classification problem always brings a large number of features, to avoid overfitting, and reduce training time cost, feature selection plays a very important role. SelectKBest in Scikit-Learn was applied for Feature selection. SelectKBest selects k most relevant features according to the given evaluation metrics. In this experiment, ANOVA F-value was used to

evaluate the importance of the feature [5]. The formula of F-value for each feature x in our experiment is as below [6]:

$$F_x = \frac{\sum_{i=1}^{K} \frac{n_i(\hat{x_i}-\hat{x})^2}{K-1}}{\sum_{i=1}^{K}\sum_{j=1}^{n_i} \frac{n_i(x_{ij}-\hat{x_i})^2}{N-K}}$$

with K the total number of distinct value of feature x, N the total number of samples, and x_{ij} the j^{th} value and number of sample in the i^{th} group among K groups. The numerator in the formula is between-group variability, and the denominator in the formula is within-group variability. The feature with a higher F-value, more confident we reject H_0 in F-test, more significant this feature would be in predicting the label in this data. In our experiment, the top 5000 features ranked by F-value would be chosen as the input feature of the models.

B. *Classification Model*

1) ***Multinomial Naive Bayes:***
Naive Bayes methods are supervised learning algorithms based on applying Bayes' theorem with assuming that features are pairwise conditional independent [7]. As a basic and easy machine learning algorithm, Naive Bayes is commonly used in so many classification problems. In this project, the performance of Naive Bayes and other neural network models would be compared. And Multinomial Naive Bayes from Sci-kit Learn with default smoothing parameter $\alpha = 1$ is used in the experiments. When $\alpha = 1$, Laplace smoothing is applied in the model. The smoothing technique enables the avoidance of zero-weighted features by reallocating weight from features with non-zero weight to those with zero weight.

2) ***Multilayer Perceptron:***
The MLP in this experiment is implemented by TensorFlow, with one input layer, one hidden layer with 64 hidden neurons, and an output layer of size 2(positive and negative). And bias neurons are included in the network. The structure of the MLP network is shown in Figure 1.

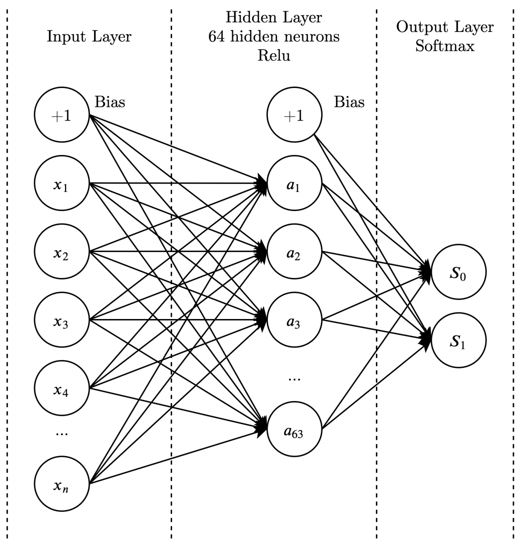

Fig. 1. MLP Structure

Relu is applied as the activation function in the hidden layer. For the output layer, Softmax is applied as activation function, to scale the output close to 0 or 1.
In order to reduce overfitting, dropout with drop rate of 0.3 would be introduced in our model. This means that we randomly block 30% of neurons in the hidden layer every iteration. This provides a regularizing effect, to prevents units from co-adapting too much because each neuron in the network can't rely on any single input feature.

3) ***Multi-Channel CNN:***
The inspiration of the Multi-Channel CNN model comes from Deep Learning for Natural Language Processing by Brownlee [8] and a paper by Kim [9]. According to Brownlee and Kim, using standard CNN models with different kernel sizes enables the model to process the raw text in different n-gram at a time. Based on Brownlee's model, I design a model that is capable of process the raw text in uni-gram, bi-gram, and tri-gram. Notice that the input layer of the Multi-Channel CNN model accepts data processed by Keras Tokenizer function, and there is no TF-IDF encoding (the reason would be explained in the Discussion section). Same as the MLP model, the Multi-Channel CNN model applies Relu and Softmax as the activation function for the hidden layers and output layer with drop rate of 0.3. The architecture of the Multi-Channel CNN model is shown in Fig 2.

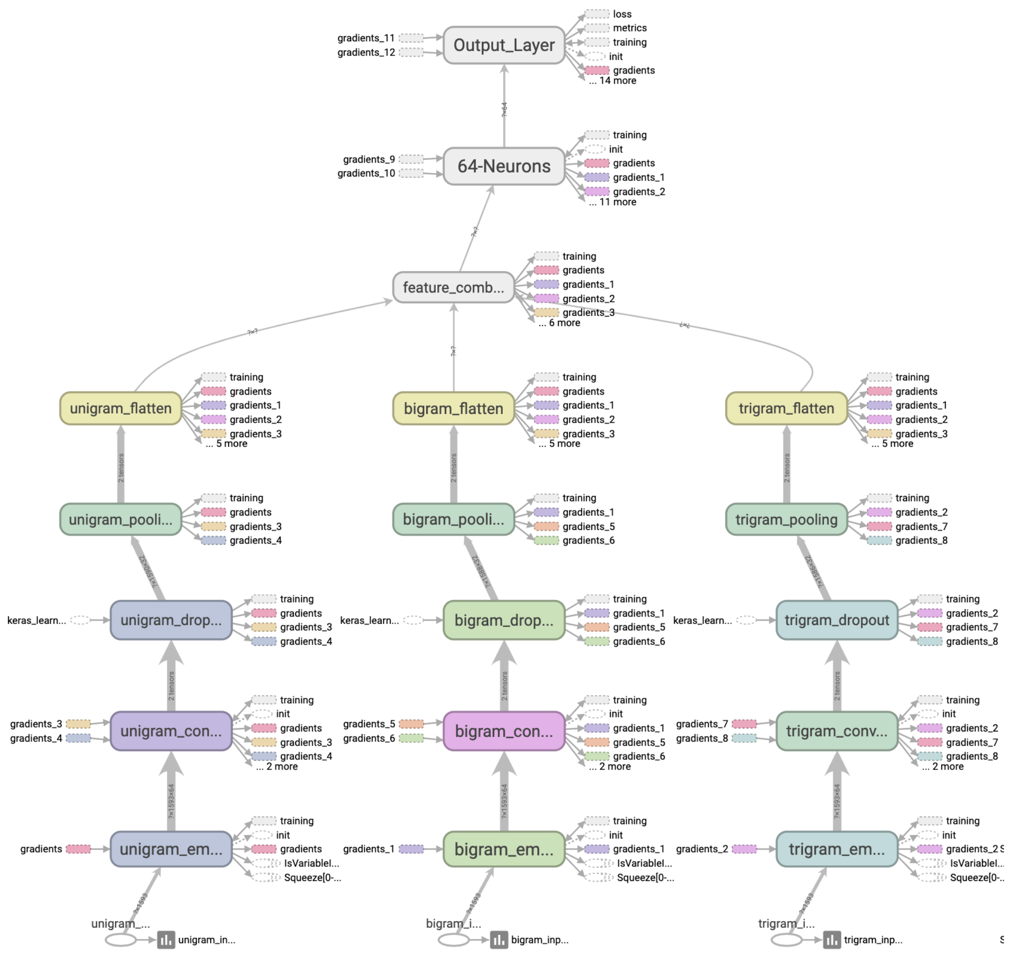

Fig. 2. N-gram CNN Architecture

C. How to Train Model

For all tests taken on MLP and Multi-Channel CNN model, totally of 3 epochs with batch size 64 would be trained. Adam with learning_rate= 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ would be chosen as optimizer. In my investigation, Insofar, RMSprop, Adadelta, Adam have very close performance in validation accuracy in this dataset. However, theoretically, the bias correction property of Adam algorithm ensures a faster and better performance than the other algorithms when the gradient becomes sparse (in some cases, the cost function of the Multi-Channel CNN model doesn't decrease with some other optimizer). In this way, Adam is chosen.

D. Performance Evaluation

After the training process, the evaluation metrics mentioned in the Background part would be used to compare the performance between different models with different pre-processing strategies. In this part, a confusion matrix for each model would be produced.

VI. DISCUSSION

In the Discussion section, only a few representative experiments will be analyzed in detail. All experiments are performed via a 5-Fold cross-validation method. The values in the confusion matrix are average values across 5-Folds.

A. Text-Cleaning

From Table II and Table III, we can see the different performance between Multi-Channel CNN models trained with and without Text Cleaning process. It can be investigated from these two tables, the overall performance of the model increases a lot when the Text Cleaning process was introduced. Especially, there is a 6% increment on validation Recall, which means that the Multi-Channel CNN model with Text-Cleaning is more capable to predict data which is actual positives.

Besides, according to the training performance and validating performance recorded in these two tables, there is an over-fitting on Multi-Channel CNN models.

B. TF-IDF

Table IV and Table V show the confusion metrics of Multinomial Naive Bayes with and without TF-IDF features extraction. It can be witnessed a small improvement of just 0.5% on validation F1 Score and 0.83% on predicting accuracy. However, there are great changes in Recall and Precision.

According to the validation Recall and Precision in the two tables, after applying TF-IDF, an increase of 2.5% can be witnessed on validation Precision, and a decrease of 1.6% can be witnessed on validation Recall. This phenomenon also appears in the MLP model, which indicates that, before using TF-IDF, the models are more likely to give a positive prediction. But after using TF-IDF, positive predictions given by the models decrease, and the accuracy of positive predictions (also known as Precision) was improved. In this way, the model's capability of predicting negative text and positive text are more balanced.

TABLE II
Confusion Matrix for 5-Fold
Multi-Channel CNN
without Text Cleaning

Training Set		**Validation Set**	
TP=18655.0	FP=323.0	TP=4153.2	FP=442.8
FN=1345.0	TN=19677.0	FN=846.8	TN=4557.2
Accuracy = 95.83%		Accuracy = 87.10%	
Recall = 93.27%		Recall = 83.06%	
Precision = 98.30%		Precision = 90.37%	
F1 Score = 95.72%		F1 Score = 86.56%	
Time Cost: 179.88 sec.			

TABLE III
Confusion Matrix for 5-Fold
Multi-Channel CNN
with Text Cleaning

Training Set		**Validation Set**	
TP=19532.4	FP=270.0	TP=4466.8	FP=485.2
FN=467.6	TN=19730.0	FN=533.2	TN=4514.8
Accuracy = 98.16%		Accuracy = 89.82%	
Recall = 97.66%		Recall = 89.34%	
Precision = 98.64%		Precision = 90.20%	
F1 Score = 98.15%		F1 Score = 89.77%	
Time Cost: 163.32 sec.			

TABLE IV
Confusion Matrix for 5-Fold
Multinomial Naive Bayes
with Text Cleaning, N-Gram

Training Set		**Validation Set**	
TP=18636.2	FP=3095.4	TP=4607.6	FP=827.8
FN=1363.8	TN=16904.6	FN=392.4	TN=4172.2
Accuracy = 88.85%		Accuracy =87.80%	
Recall = 93.18%		Recall = 92.15%	
Precision = 85.76%		Precision = 84.77%	
F1 Score = 89.31%		F1 Score = 88.31%	
Time Cost: 19.62 sec.			

TABLE V
Confusion Matrix for 5-Fold
Multinomial Naive Bayes
with Test Cleaning, TF-IDF, N-Gram

Training Set		**Validation Set**	
TP=18239.6	FP=2450.4	TP=4530.0	FP=666.8
FN=1760.4	TN=17549.6	FN=470.0	TN=4333.2
Accuracy = 89.47%		Accuracy = 88.63%	
Recall = 91.20%		Recall = 90.60%	
Precision = 88.16%		Precision = 87.17%	
F1 Score = 90.59%		F1 Score = 88.85%	
Time Cost: 18.22 sec.			

C. N-gram

Table VI and Table VII show the confusion matrices of MLP with and without N-gram features (N-gram here means Uni-gram, Bi-gram and Tri-gram). We can see from these two tables when N-gram was introduced, the overall performance of MLP has been slightly increased. This indicated that feature extraction with N-gram brings more valuable information to the models. However, the time cost of a model with N-grams is almost twice that of those without N-grams, I guess that this is because the input feature under N-gram vectorization is in a higher dimension.

D. Some other Findings

Table VIII shows the validation performance of some of my experiments. And there are some valuable findings in this table:

TABLE VI
Confusion Matrix for 5-Fold
Multi-Layer Perceptron
with Text Cleaning, TF-IDF, N-Gram

Training Set		**Validation Set**	
TP=18636.6	FP=1639.6	TP=4616.8	FP=580.8
FN=1363.4	TN=18360.4	FN=383.2	TN=4419.2
Accuracy = 92.49%		Accuracy = 90.36%	
Recall = 93.18%		Recall = 92.34%	
Precision = 91.91%		Precision = 88.83%	
F1 Score = 92.54%		F1 Score = 90.55%	
Time Cost: 43.32 sec.			

TABLE VII
Confusion Matrix for 5-Fold
Multi-Layer Perceptron
with Test Cleaning, TF-IDF

Training Set		**Validation Set**	
TP=18530.0	FP=1483.2	TP=4443.6	FP=562.8
FN=1470.0	TN=18516.8	FN=556.4	TN=4437.2
Accuracy = 92.62%		Accuracy = 88.81%	
Recall = 92.59%		Recall = 88.87%	
Precision = 92.65%		Precision = 88.76%	
F1 Score = 92.62%		F1 Score = 88.82%	
Time Cost: 20.32 sec.			

Mentioned in previous parts, TF-IDF was witnessed helpful in improving the predicting performance and balanced the capability of positive and negative predicting. But when TF-IDF was applied in the Multi-Channel CNN model, the performance of the model dropped to the random baseline (near 50% accuracy). This is because the Multi-Channel CNN model needs to capture the relationship between words that appear next to each other. In this situation, frequency feature extraction is not able to provide necessary information for training the Multi-Channel CNN model, since it only provides the frequency of terms but not the order. These cases indicate that not all text pre-processing methods can be applied in all models, in most cases, only an appropriate pre-processing method can help the model to reach its optimum performance.

Besides, according to the experiments on Naive Bayes and MLP, under the same text pre-processing method, MLP always hits a better performance in comparison to Naive Bayes. This phenomenon tells us that, the choice of models is also important in the Sentiment Analysis problem.

TABLE VIII
VALIDATION PERFORMANCE OF MODELS WITH DIFFERENT TEXT PRE-PROCESSING METHODS

Models	Pre-processing Method			Evaluation Metrics			
	Text-Cleaning	*TF-IDF*	*N-gram*	Recall	Precision	F1-Score	Accuracy
Naive Bayes	T	T	T	90.60%	87.17%	88.85%	88.63%
	T	T	F	88.71%	86.18%	87.42%	87.12%
	T	F	T	92.15%	84.77%	88.31%	87.80%
	F	F	F	84.97%	86.29%	85.62%	85.88%
Multi-Layer Perceptron	T	T	T	92.34%	88.83%	**90.55%**	**90.36%**
	T	T	F	88.87%	88.76%	88.82%	88.81%
	T	F	T	**92.83%**	86.71%	89.67%	89.36%
	F	F	F	85.81%	87.80%	86.80%	87.14%
Multi-Channel CNN	T	F	T	89.34%	90.20%	89.77%	89.82%
	F	F	T	83.06%	**90.37%**	86.56%	87.10%
	T	T	T	47.35%	49.18%	48.25%	49.21%
	F	T	T	51.36%	51.02%	51.19%	51.03%

VII. CONCLUSION AND FUTURE WORKS

A. Conclusion

Based on the results of my experiments, a suitable data representation can definitely optimize the performance and training time cost of the models. Text-cleaning, TF-IDF, and N-gram are witnessed helpful for improving the performance of different classifiers. But not all pre-processing methods are acceptable for all models. In some cases, abuse of the pre-processing method can have a negative impact on predicting results. In this way, the text pre-processing strategy should be chosen according to the property of the models.

B. Future Works

A conference paper by Thongtan and Tanasanee in July 2019 [10] shows that accuracy on the IMDB dataset is improved when using cosine similarity instead of dot product during the text vectorization process while using feature combination with Naive Bayes and n-grams achieves an accuracy of 97.42%. I would like to implement this method and compare the performance of different pre-processing strategies for further study.

REFERENCES

[1] Sentiment Analysis, http://ai.stanford.edu/~amaas/data/sentiment/

[2] Natural Language Toolkit, NLTK 3.5b1 documentation, https://www.nltk.org/

[3] Jurafsky, Dan, and James H. Martin. Speech and Language Processing. Upper Saddle River, NJ: Prentice Hall, Pearson Education International, 2014.

[4] Salton, G., & Buckley, C. Weighting approaches in automatic text retrieval. Information Processing and Management, 1988. 24(5), 513–523.

[5] Sklearn.feature_extraction.text.tfidfvectorizer https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text .TfidfVectorizer.html?highlight=tfidf#sklearn.feature_extraction.text.TfidfVectorizer

[6] Nadir E, Othman I, and Ahmed O, "A Novel Feature Selection Based on One-Way ANOVA F-Test for E-Mail Spam Classification", The Research Journal of Applied Sciences, Engineering and Technology, 2014.

[7] 1.9.Naive Bayes, https://scikit-learn.org/stable/modules/naive_bayes.html

[8] Brownlee, Jason. Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems, Machine Learning Mastery, Jason Brownlee, 2017.

[9] Kim, Yoon. "Convolutional Neural Networks for Sentence Classification." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. https://doi.org/10.3115/v1/d14-1181.

[10] Thongtan, Tan, and Tanasanee Phienthrakul. "Sentiment Classification Using Document Embeddings Trained with Cosine Similarity." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, July 2019. https://doi.org/10.18653/v1/p19-2057.