A Novel Efficient and Effective Preprocessing Strategy for Text Classification

Anonymous ACL submission

Abstract

Text classification is an essential task of natural language processing. Preprocessing, which determines the representation of text features, is one of the key steps of text classification architecture. This paper proposes a novel efficient and effective preprocessing strategy with three methods for text classification using OMP algorithm to complete the classification. The main idea of our new preprocessing strategy is that we combine regular filtering and/or stopwords removal with tokenization and lowcase convertion, which can effectively reduce the feature dimension and improve the quality of text feature matrix to some extent. Simulation tests on 20Newsgroups dataset show compared with the existing state-of-the-art method, our new best method reduces the number of features by 19.85%, 34.35%, 26.25%, and 38.67%, and increase the speed of text classification by 17.38%, 25.64%, 23.76%, and 33.38% with similar classification accuracy on religion, computer, science and sport data, respectively.

1 Introduction

Text classification (TC), which is the task of assigning one or more categories from a set of known categories to a centralized document, is one of the most fundamental tasks in Natural Language Processing (NLP) (Joachims, 1998). It has been successfully applied in spam filtering (Guzella and Caminhas, 2009), sentiment analysis (Medhat et al., 2014) and other NLP tasks.

Text classifications are highly useful for information discovery and opinion mining. However, with more and more electronic documents generated, many challenges are faced by TC due to organize and classify a large number of documents which cases high-dimensional data. While, text categorization is essentially high-dimensional where medium-sized datasets can contain tens of thousands of unique words (Joachims, 2002). Additionally, text high-dimensional data significantly affect the training time and classification accuracy of the classifier (Wang et al., 2016), which may influence the effectiveness and efficiency of classification. It is easy to increase the likelihood of overfitting since there are a great many text features in text data (Skianis et al., 2016; Abbasi et al., 2010).

Although some models achieve good performance, the problem of hight dimension of text still remains. However, text classification remains an outstanding research area using various techniques and their combinations to solve these problems (Mirończuk and Protasiewicz, 2018). In addition, preprocessing, which is a fundamental text processing technology of TC, can effectively alleviate text data explosion problem and provide a sufficient guarantee. It has been claimed in (Uysal and Gunal, 2014) that preprocessing methods in TC are as important as the feature extraction, feature selection and classification steps, and a proper combination of preprocessing tasks can significantly improve the classification performance. Furthermore, (HaCohen-Kerner et al., 2020) explored the impacts of different preprocessing combinations on classification, and proved that the combination of multiple preprocessing strategies can improve TC's accuracy.

Inspired by (Uysal and Gunal, 2014; HaCohen-Kerner et al., 2020; Haddi et al., 2013), in this paper, we propose a novel strategy with three methods by combing some popular preprocessing strategies to reduce the general text feature dimension and unnecessary feature items. Simulation tests show that our new preprocessing strategy has an essential impact on classification efficiency in TC.

The rest of the paper is structured as follows. Section 2 briefly describes the process of text classification. Section 3 presents the work of this paper in detail, including the proposed efficient and effective strategy. The experiments and results analysis are discussed in Section 4. Section 5 gives a few conclusions and future proposals.

2 Text classification process

In this section, we briefly introduce the architecture and related work of TC.

2.1 Text classification architecture

TC architecture is generally consisted of preprocessing, feature extraction, feature selection and classification in shallow learning model.

There are many text document datasets, such as long and short text. Based on text document, the first step of TC is preprocessing, which filters out words that have no effect on text data and other useless symbols. Preprocessing is essential as it not only removes noise/useless data, but alleviates the adverse effects caused by the excessive dimension of text data. There are numerous preprocessing methods, such as tokenization, stopwords removal, lowercase conversion and lemmatisation, etc.

Feature extraction is the second step of TC. Its task is the process of transforming text data from unstructured to structured. While, the quality of this transformation process will directly affect the final classification result. Currently commonly used feature extraction methods are: Bag-of-words (BOW) (Joulin et al., 2016), word frequency-inverse document frequency (TF-IDF) (Joachims, 1996), word2vec (Goldberg et al., 2014).

Feature selection, which selects the relevant and essential features, and removes irrelevant and redundant features (Zebari et al., 2020; Rehman et al., 2017), is the third step of TC. Its main goal is to construct a feature subset as small as possible but represents the whole input data (Velliangiri et al., 2019). A wide range of feature selection methods include mutual information, information gain and Chi-square (Uysal and Gunal, 2012).

The last step of TC is to train a classifier using the previously created features, and define the class for each input text. Widely used classifiers include Support Vector Machine (SVM) algorithm (Cortes and Vapnik, 1995), Naive Bayes (NB) (Heckerman, 2004), Linear Regression (LR) (Christensen, 2006) and neural network classification algorithm (Kim, 2014; Lai et al., 2015; Devlin et al., 2018).

2.2 Related work

Preprocessing is referred to as data cleaning, data reduction and discretion (Chandrasekar and Qian, 2016). It is as important as the feature extraction, feature selection and classification steps in TC (Uysal and Gunal, 2014). Many studies tend to use tokenization and lowercase conversion as the main and usual preprocessing methods.

(Skianis et al., 2018) used a ℓ_2-regularization method to handle the overfitting problem caused by high dimensional text and devloped a method called logistic-Orthogonal Matching Pursuit (OMP) for TC. Note that OMP is a classic method for finding the "best matching" of multi-dimensional data with sparse approximation from a large dictionary. Simulation test results in this paper show that logistic-OMP can improve the accuracy of TC.

3 Proposed preprocessing strategy

The conventional methods for improving the accuracy of TC mainly consider how to use a better classification algorithm, while it is rare to improve it by utilizing a better preprocessing method. Hence, in this section we propose three novel efficient and effective preprocessing strategies of TC.

Tokenization, lowercase conversion, stopwords removal and regular filtering are four widely used preprocessing strategies. More exactly, tokenization is the process of dividing a sentence into words or phrases using a word segmentation algorithm. Lowercase conversion refers to the process of converting all uppercase letters to lowercase letters. They are fundamental methods, and we use them as our basic preprocessing method. Regular filtering primarily filters many useless non-alphanumeric characters. It removes punctuation marks, special characters, and blank characters before and after words. Stopwords removal is to remove these words that do not contribute to text features and reduce the features that contribute more to the retention of useless features.

Although tokenization, lowercase conversion, stopwords removal and regular filtering are four popular preprocessing strategies, as far as we know, there is rare preprocessing method which is a combination of some or all of them. Hence, we propose three preprocessing methods which are combinations of some or all of them. More exactly, Table 1 summarizes our three preprocessing methods (NP_1, NP_2, NP_3) which are the combinations of the existing four methods, where T, L, R and S denote tokenization, lowercase conversion, regular filtering, and stopwords removal, respectively. 1 means execution and 0 is no execution, "EP" means existing preprocessing methods and "NP" denotes proposed preprocessing methods. Note that NP_0, which was

proposed in (Skianis et al., 2018), is a baseline method for comparison. For brevity, these four preprocessing methods are represented by NP_0=1100, NP_1=1101, NP_2=1110, and NP_3=1111. From Table 1, we can see that NP_0, NP_1, NP_2 and NP_3 are the combination of T and L, the combination of T, L and S, the combination of T, L and R, and the combination of T, L, R and S, respectively.

NP \ EP	T	L	R	S
0	1	1	0	0
1	1	1	0	1
2	1	1	1	0
3	1	1	1	1

Table 1: Three proposed preprocessing methods (NP_1, NP_2, NP_3)

In the following, we utilize a text from 20newsgroups[1] dataset, which is an international standard dataset for TC, to illustrate the three proposed preprocessing methods. More exactly, Table 2 displays the texts after performing the four preprocessing methods. The sentence before preprocessing is "Answer: an especial witness-one who is suppose to be apersonal witness. That means to be a true apostle, one must have Christ appear to them.". It is a religion review data from 20newsgroups.

NP	After
NP_0 (14)	answer,especial,who,suppose, personal,that,means,true,apostle, one,must,have,christ,appear
NP_1 (10)	answer,especial,witness,one, suppose,personal,witness,true, christ,them
NP_2 (15)	answer,especial,witness,one,who, suppose,personal,that,means,true, must,have,christ,appear,them
NP_3 (7)	answer,especial,witness, suppose,personal,true,christ

Table 2: Text after performing the four preprocessing methods

The original sentence has 31 features, however, from Table 2, we can see that, after performing the NP_0, NP_1, NP_2 and NP_3 methods, there are 14, 10, 15 and 7 features, respectively. Hence, it shows that compared with NP_0-NP_2, NP_3 reduces more number of features. Furtherfore, the three proposed preprocessing methods are significant in reducing the number of features, which helps to improve the accuracy and efficiency of TC (more details on this will be presented in the next section).

[1]http://qwone.com/ jason/20Newsgroups/

4 Experiments and analysis

In this section, we conduct numerical tests on the 20newsgroups which is an international standard dataset, to illustrate the effects of the three proposed preprocessing methods on reducing the number of features and improving the efficiency and accuracy of TC. We perform data preprocessing in Pycharm2021 and implement TC's model on matlab based on extensive experiments.

4.1 The effect of the new preprocessing strategies on reducing the number of features

In the subsection, we use 20newgroups to illustrate the three proposed preprocessing methods.

Table 3 shows the numbers of features after performing the four preprocessing methods on religion, computer, science and sport data from the 20newsgroups dataset, where the numbers of features before the preprocessing are 31727, 26568, 39186 and 35199 respectively. From Table 3, we can see that the proposed three methods have significant effects on reducing the number of features. In particular, NP_3 method reduces the numbers of features to 14994, 11552, 17994 and 12837 for the four data, respectively, and compared with the baseline method NP_0 (Skianis et al., 2018), the improvements are respectively 19.85%, 34.35%, 26.25% and 38.67%.

Note that reducing the number of features can improve the efficiency of TC, for more details, please see the next subsection.

Data \ NP	NP_0	NP_1	NP_2	NP_3
religion	18707	18403	15667	14994
computer	17128	16761	12179	11552
science	24398	24075	18660	17994
sport	20932	20608	13477	12837

Table 3: Number of features after performing the four preprocessing methods

4.2 Time analysis

In this subsection, we compare the training time by applying OMP algorithm (Mirończuk and Protasiewicz, 2018) of four preprocessing strategies.

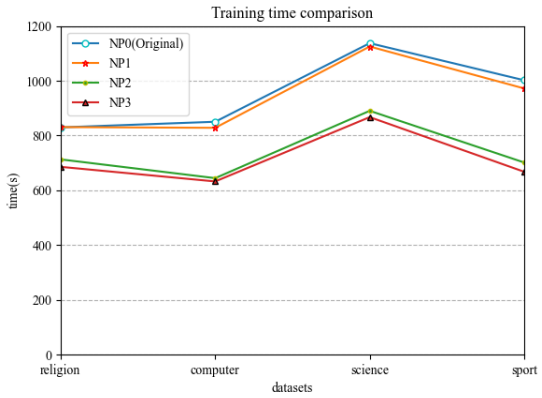

Figure 1: The comparison of training time

Figure 1 shows that the training time, which is almost the running time of text classification algorithm, of the four preprocessing strategies for preprocessing religion, computer, science and sport data from 20newsgroups dataset over 50 independent running experiments, and Table 4 shows the relative training time of NP_1 -NP_3 to NP_0.

From Figure 1, we can see that the training time of all the four preprocessing strategies depends on the quantity and quality of data. More importantly, Figure 1 and Table 4 show that the training time of NP_1 -NP_3 are much shorter than that of NP_0 except that the training time of NP_1 is 0.13% longer than that of NP_0 for preprocessing the religion data. This is because, the new preprocessing strategies can significantly reduce the number of features, thereby reducing the computational cost of OMP algorithm as our classification algorithm.

Table 4 also shows that NP_3 has the significant improvement in reducing the training time, which are 17.38%, 25.64%, 23.76% and 33.38% for preprocessing religion, computer, science and sport data, respectively.

datasets	NP_1	NP_2	NP_3
religion	-0.13%	14.09%	17.38%
computer	2.60%	24.23%	25.64%
science	1.08%	21.73%	23.76%
sport	0.30%	30.007	33.38%

Table 4: The relative training time of NP_1 -NP_3 to NP_0 (Blue represents the best)

4.3 Accuracy analysis

In this subsection, we compare the classification accuracy by applying OMP algorithm to text classification (Mirończuk and Protasiewicz, 2018) based on the four preprocessing methods. Note that accuracy is an important evaluation metric in NLP and is defined as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN},$$

where "TP" represents that the actual is a positive case and the prediction is positive; "FP" represents that the actual is a negative case and the prediction is positive; "FN" is opposite with "FP" as well as "TN" with "TP". Accuracy is the ratio of the number of correct classifications to the total number of predictions. The higher the accuracy, the better the classification model.

Table 5 illustrates the classification accuracy of the four preprocessing methods for classifying religion, computer, science and sport data from 20newsgroups dataset. From Table 5 we can see that NP_1 has better effect in improving the classification accuracy for classifying computer and sport data than NP_0, NP_2 has better effect in improving the classification accuracy for classifying science data than NP_0 and NP_3 has better effect in improving the classification accuracy for classifying religion data than NP_0.

Although Table 5 shows that both NP_2 and NP_3 may have slightly worse effect in improving the classification accuracy for classifying some data than NP_0, as shown in the above subsection, NP_2 and NP_3 are much more efficiency than NP_0.

M / Data	NP_0	NP_1	NP_2	NP_3
religion	0.919	0.915	0.926	0.933
computer	0.889	0.898	0.888	0.883
science	0.961	0.956	0.962	0.946
sport	0.954	0.962	0.930	0.948

Table 5: Classification accuracy (Blue is the best)

5 Conclusion

In this paper, we designed an effective and efficient preprocessing strategy with three methods for TC. Our experimental results show that the three proposed preprocessing methods can significantly reduce the number of features and improving the training time with more or less the same classification accuracy based on the existing preprocessing methods. We will implement Elmo, BERT and other methods in DL to enhance TC's performance in the future.

References

A. Abbasi, S. France, Z. Zhang, and H. Chen. 2010. Selecting attributes for sentiment classification using feature relation networks. *IEEE Transactions on Knowledge and Data Engineering*, 23(3):447–462.

P. Chandrasekar and K. Qian. 2016. The impact of data preprocessing on the performance of a naive bayes classifier. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 618–619. IEEE.

R. Christensen. 2006. *Log-linear models and logistic regression*. Springer Science & Business Media.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Y. Goldberg et al. 2014. word2vec parameter learning explained continuous bagof-word model. In *Proc. Int. Conf. Learn. Represent.(ICLR 2013)*, volume 15.

T. S. Guzella and W. M. Caminhas. 2009. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, 36(7):10206–10222.

Y. HaCohen-Kerner, D. Miller, and Y. Yigal. 2020. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, 15(5):e0232525.

E. Haddi, X. Liu, and Y. Shi. 2013. The role of text preprocessing in sentiment analysis. *Procedia Computer Science*, 17:26–32.

D. Heckerman. 2004. Bayesian networks for data mining. data mining and knowledge discovery.

T. Joachims. 1996. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science.

T. Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

T. Joachims. 2002. *Learning to classify text using support vector machines*, volume 668. Springer Science & Business Media.

A. Joulin, E. Grave, P. Bojanowski, and et al. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Y. Kim. 2014. Convolutional neural networks for sentence classification. *Eprint Arxiv*.

S. Lai, L. Xu, K. Liu, and J. Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

W. Medhat, A. Hassan, and H. Korashy. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

M. M. Mirończuk and J. Protasiewicz. 2018. A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, 106:36–54.

A. Rehman, K. Javed, and H. A. Babri. 2017. Feature selection based on a normalized difference measure for text classification. *Information Processing & Management*, 53(2):473–489.

K. Skianis, F. Rousseau, and M. Vazirgiannis. 2016. Regularizing text categorization with clusters of words. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1827–1837.

K. Skianis, N. Tziortziotis, and M. Vazirgiannis. 2018. Orthogonal matching pursuit for text classification. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 93–103. Association for Computational Linguistics.

A. K. Uysal and S. Gunal. 2012. A novel probabilistic feature selection method for text classification. *Knowledge-Based Systems*, 36:226–235.

A. K. Uysal and S. Gunal. 2014. The impact of preprocessing on text classification. *Information processing & management*, 50(1):104–112.

S. Velliangiri, S. Alagumuthukrishnan, and et al. 2019. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165:104–111.

D. Wang, H. Zhang, R. Liu, X. Liu, and J. Wang. 2016. Unsupervised feature selection through gram–schmidt orthogonalization—a word co-occurrence perspective. *Neurocomputing*, 173:845–854.

R. Zebari, A. Abdulazeez, D. Zeebaree, and et al. 2020. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *Journal of Applied Science and Technology Trends*, 1(2):56–70.