Scientific Research Publishing

Journal of Computer and Communications, 2023, 11, 1-14
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A Novel Efficient and Effective Preprocessing Algorithm for Text Classification

Lijie Zhu[1], Difan Luo[2]

[1]College of Information Science and Technology, Jinan University, Guangzhou, China
[2]University of Birmingham Joint Institute, Jinan University, Guangzhou, China
Email: zhulijiestudy@126.com, ldf20@stu2020.jnu.edu.cn

How to cite this paper: Zhu, L.J. and Luo, D.F. (2023) A Novel Efficient and Effective Preprocessing Algorithm for Text Classification. *Journal of Computer and Communications*, **11**, 1-14.
https://doi.org/10.4236/jcc.2023.113001

Received: February 8, 2023
Accepted: March 10, 2023
Published: March 13, 2023

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. It proposed a novel efficient and effective preprocessing algorithm with three methods for text classification combining the Orthogonal Matching Pursuit algorithm to perform the classification. The main idea of the novel preprocessing strategy is that it combined stopword removal and/or regular filtering with tokenization and lowercase conversion, which can effectively reduce the feature dimension and improve the text feature matrix quality. Simulation tests on the 20 newsgroups dataset show that compared with the existing state-of-the-art method, the new method reduces the number of features by 19.85%, 34.35%, 26.25% and 38.67%, improves accuracy by 7.36%, 8.8%, 5.71% and 7.73%, and increases the speed of text classification by 17.38%, 25.64%, 23.76% and 33.38% on the four data, respectively.

Keywords

Text Classification, Preprocessing, Feature Dimension, Orthogonal Matching Pursuit

1. Introduction

Internet technology and information technology in the new era promote the explosive growth of the global data scale. The era of big data produces a great deal of data, but big data does not represent big information [1]. Text data mostly come from news media, social platforms, emails, and comments on e-commerce platforms. However, while the Internet brings convenience to people's enjoyment of massive text information services, it also inevitably creates some new

problems, such as chaotic text, flooding of spam/information, false comments, etc. How to effectively deal with the complex and diverse text data generated by the Internet, extract the information people need from the huge amount of complex data, and mine the hidden useful information is a research hotspot in academia and industry at present [2].

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) [3]. It has been wonderfully applied in spam filtering [4], sentiment analysis [5]. Text classifications are greatly valuable for information discovery and opinion mining. Nevertheless, plenty of challenges are faced by text classification as a surging number of electronic documents generate and the organization and classification of large numbers of documents result in high-dimensional data. While text categorization is essentially high-dimensional where medium-sized datasets may involve tens of thousands of unique words [6]. Additionally, text high-dimensional data enormously affect the learning/training time and classification accuracy of the classifier [7] that may influence the effectiveness and efficiency of classification. Because there are a large number of text features in text data, the possibility of overfitting can easily increase [8] [9]. This work pays attention to the issue of efficiency in the text classification task, an issue that is often forgotten or little addressed. Today, due to the amount of information available and the type of approaches used (based on Deep Learning), it is of crucial importance. While many researchers apply various machine learning algorithms to improve the effectiveness of text classifiers, few people systematically compare and analyze the impact of different preprocessing strategies on the accuracy of text classification systems. In addition, though some models and algorithms achieve excellent performance, the problem of the high dimension of the text remains. Text categorization remains a prominent research area that requires the use of various techniques and their combinations to address these problems [10]. What's more, preprocessing, a fundamental text processing technology of text classification, can availably alleviate text data explosion problems and put forward a sufficient guarantee.

Inspired by preprocessing to improve text classification and combined with some commonly used preprocessing strategies [9] [10], a new strategy of three methods is proposed by us to reduce unnecessary feature items and the general text feature dimension. Simulation tests indicate that this novel preprocessing strategy has a primary impact on the efficiency of text classification. It simplifies the process of classification and also improves efficiency. The major contributions of this study can be summarized as follows:

- It combined stopword removal and/or regular filtering with tokenization and lowercase conversion to effectively and significantly reduce the text feature dimension and improve the quality of the text feature matrix.
- It proposed a new word frequency representation matrix through preprocessing strategies to ensure the matrix quality and reduce the matrix dimen-

sion and classify with the Orthogonal matching pursuit algorithm.

- It studied the impact of different preprocessing strategies on text classification and analyze the classification performance of the novel algorithm compared with supervised learning algorithms.

The rest of the paper is structured as follows. Section 2 succinctly sketches the process of text classification. Section 3 presents the proposed efficient and effective strategy of text classification in detail. Section 4 describes and discusses the experiments and results analysis. A few conclusions and future proposals are given in Section 5.

2. Text Classification Theory and Technology

This part succinctly introduces the process and related work for text classification in this section. The text classification architecture introduces the specific text classification process in the shallow classification model, and the related work is an introduction to some research foundations of these search objects of this work.

2.1. Text Classification Process

The text classification architecture consists of text preprocessing, feature extraction/feature selection, text representation and classifiers (see Figure 1).

In the text classification architecture, preprocessing is the first step of text classification. Preprocessing can filter out words that have no effect on text data and other useless symbols. More precisely, it removes noise/useless data and mitigates the adverse effects of excessive text data dimension. There are quite a few preprocessing methods, including tokenization, lowercase conversion, lemmatisation,

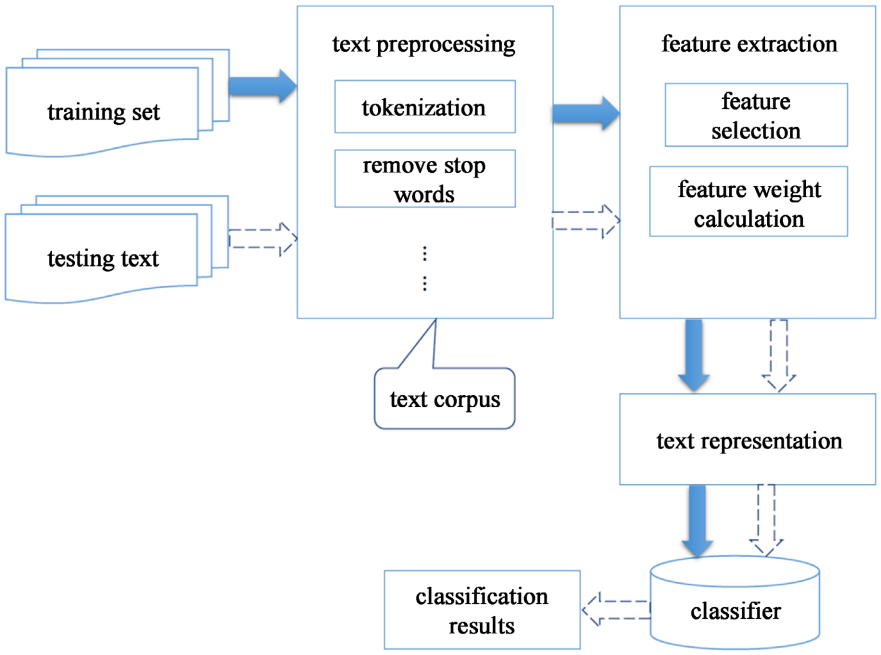

Figure 1. Text classification architecture.

and stopword removal, etc.

Feature extraction is the second step, which is a process of creating new features that depend on the original input feature set to decrease the high dimensionality of the feature vector [11]. The transformation method is done by algebraic transformation, and according to some optimization criteria. Based on the text feature set obtained by feature processing, the task of feature extraction is the process of converting text data from unstructured to structured. It quantifies the feature words extracted from the text feature set to represent text information. Currently commonly used feature extraction methods are: Bag-Of-Words (BOW) [12], Term Frequency-Inverse Document Frequency (TF-IDF) [13], word2vec [14], etc. The feature selection of the word frequency method is to delete the words whose word frequency is less than a certain threshold, thereby reducing the dimensionality of the feature space. This approach selects relevant and essential text features and deletes irrelevant and redundant text features [15].

Text representation as structured data that algorithms can process is undoubtedly an essential part of text classification, which is the third step of text classification. It can be divided into the following categories: 1) based on set theory models; Boolean models, models based on fuzzy sets and extended Boolean models; 2) based on algebraic models: vector space models (VSM) and semantic-based text representation; 3) regression model and binary independent probability model based on the probability statistics model.

The final step in text classification is to train the classifier employing the previously created features and determine the final category for each input text. Frequently used text classifiers include Support Vector Machine (SVM) algorithm [16], Naive Bayes (NB) [17] and neural network classification algorithm [18].

In machine learning, the embedded feature selection method is a method that fully combines feature selection with the classifier, that is, the process of integrating feature selection into the classifier. Skianis et.al combined a ℓ_2-regularization method to handle the overfitting problem caused by high-dimensional text and developed a strategy called logistic-Orthogonal Matching Pursuit for text classification [19]. It is particularly important to note that the Orthogonal Matching Pursuit algorithm (OMP) is a classic greedy method, which is the basis of many commonly used efficient algorithms at present and has the characteristics of simple and efficient [20]. In the process of iterative selection of atoms by the OMP algorithm, the residuals are always orthogonal to the linear representation vector formed by the selected atoms. This means that an atom will not be selected twice, that is, the result will converge in a limited number of steps. More importantly, OMP algorithm is a classic strategy used to find the "best matches" of multidimensional data with sparse approximations from large dictionaries.

2.2. Related Work

This section primarily reviews previous work on preprocessing strategies in this

subsection. Preprocessing is referred to as data cleaning, data reduction, and discretion [21]. It can eliminate irrelevant and redundant features and guarantee certain classification accuracy. It has been claimed in [22] that preprocessing strategies in text classification are as important as classification, feature selection and feature extraction. Importantly, proper combination of preprocessing tasks can markedly enhance classification performance. This article investigated the impact of preprocessing tasks on text to compare the effects of these preprocessing tasks in different languages. In this way, the contribution of preprocessing tasks to classification success across different feature dimensions, the possible interactions between these tasks, and the dependencies of these tasks on the language and domain under study are all extensively assessed. The following is the specific method of preprocessing [21] [22].

1) Tokenization is the process of breaking down the text into tokens. A token is a nonempty sequence of characters, excluding spaces and punctuation.

2) Special character removal removes digits and all the special characters (+, −, *, /, \, =, !,., ?, ;,:, ′ , <, >, |, {, }, [,], $, #, &, %).

3) Stemming is an extensively used technique in text analysis. Stemming is the process of removing inflectional affixes of words, and reducing the words to their stems.

4) Lowercase conversion refers to converting all terms in a text string to lowercase.

5) Stop word removal removes meaningless words in the text (e.g., is, at, the, etc.). These words are meaningless for the evaluation of the document content.

Carlos *et al.* have assessed the effect of combining different preprocessing techniques with several classification algorithms available in the WEKA tool [23]. The experiments show that the application of pruning, stemming and WordNet reduces significantly the number of attributes and improves the accuracy of the results. Furthermore, Yaakov *et al.* [24] discussed the impacts of different preprocessing combinations on classification. They demonstrated that the combination of multiple preprocessing strategies can enhance text classification accuracy. This paper analyzes twelve different preprocessing techniques on the Twitter dataset and observes their impact on the classification tasks they support. It also proposes a systematic approach to text preprocessing to apply different preprocessing techniques to preserve features without losing information [25]. Their results show that some preprocessing techniques negatively impact performance and have identified these techniques, along with the best performing combination of preprocessing techniques. Preprocessing makes the feature vector of the text carry as much category information as possible while reducing the dimensionality, in preparation for subsequent feature selection.

The current state-of-the-art text classification strategies either focuses on feature selection algorithms or on how to further "optimize" the classification method, but these attempts ignore the relevance and high-dimensionality of text data in the primary cleaning process. Therefore, this work explores a new combination of preprocessing methods to optimize the high-dimensional and high-sparse

feature matrix of the Bag of Words text representation model based on the above issues to provide an effective classification strategy.

3. Proposed Method

The existing methods to improve the performance of text classification mainly consider how to use better classification algorithms, but rarely study the impact of preprocessing strategies on text classification. Given the sparsity of high-dimensional data and the unstructured characteristics of text classification data, this summary discusses a classification method specifically for high-dimensional data. This article proposes a novel strategy to explore combining three different basic methods of preprocessing by addressing the characteristics of different preprocessing methods. In this paper, this work reduces unnecessary feature items of text and carries as much category information as possible to solve the problem of high-dimensional data.

3.1. Proposed Preprocessing Strategy

Adopting an appropriate preprocessing strategy can improve the overall performance and efficiency of text classification for a specific text dataset. The preprocessing process can clean out outliers, reduce dataset "noise" and address issues such as data imbalance. In addition, preprocessing strategies can also effectively enhance and improve the quality of datasets or feature matrices in text classification tasks.

Tokenization, lowercase conversion, regular filtering, and stopword removal are four closely used preprocessing strategies [21] [22]. More precisely, tokenization refers to the process of dividing a sentence into words or phrases applying word segmentation algorithm. Lowercase conversion is the process of converting all uppercase letters to lowercase letters. Both are this basic preprocessing method since they are fundamental preprocessing strategies. Stopword removal refers to remove these words that do not contribute to text features and diminish the impact of useless features on the correct classification of main features. Regular filtering principally filters numerous useless non-alphanumeric characters. Its function removes punctuation marks, special characters, and blank characters before and afterward. Although tokenization, lowercase conversion, regular filtering and stopword removal are four popular preprocessing strategies, a rare preprocessing method is a combination of some or all of the basic preprocessing strategies. Consequently, this work proposes three preprocessing methods that are combinations of some or all of them. **Table 1** summarizes the three preprocessing methods (NP_1, NP_2, NP_3) that are the combinations of the basic preprocessing methods.

Where TK, LC, RF, and SR denote tokenization, lowercase conversion, regular filtering, and stopword removal. Specifically, 1 refers to execution and 0 means no execution, and it utilizes ✓ for 1 and × for 0. TK and LC in the underline are basic preprocessing methods. These four preprocessing methods are represented

Table 1. Three proposed methods (NP_1, NP_2, NP_3).

NP	TK	LC	RF	SR
NP_0	✓	✓	×	×
NP_1	✓	✓	×	✓
NP_2	✓	✓	✓	×
NP_3	✓	✓	✓	✓

by NP_0 = 1100, NP_1 = 1101, NP_2 = 1110, and NP_3 = 1111 for brevity. Note that NP_0 is a baseline method for comparison proposed in [19], and NP_{1-3} are the preprocessing strategies of different combinations proposed by us. More precisely, NP_0, NP_1, NP_2 and NP_3 are an amalgam of TK and LC, a combination of TK, LC and SR, an amalgam of TK, LC and RF, and a combination of TK, LC, RF and SR, respectively.

3.2. A Novel Preprocessing and OMP Strategy for Text Classification

The appropriate text classification solution also depends to some extent on the classification strategy. Considering this situation, this paper puts forward a new embedded feature selection method for classification unlike existing machine learning methods. It can quickly find the best feature set with higher efficiency. Preprocessing prepares for the subsequent feature selection, and it makes the text feature vector carry as much category information as possible while reducing the dimensionality. In particular, the preprocessing strategy proposed in this paper can reduce the dimensionality of the feature matrix, that is, reduce data redundancy. For example, it can filter out some terms in the word frequency matrix to reduce the redundancy and sparsity of the text feature matrix.

In other words, the preprocessing method can filter out some useless feature items in the word frequency matrix to reduce the interference of the high dimensionality and sparsity of the text feature matrix on the efficiency and performance of text classification. Bag-Of-Words (BOW) is the most commonly used statistical feature in literature to extract high-dimensional sparse features, and unigram is used as the representation of its feature matrix [12]. Although the BOW is simple and efficient to implement, which gives a way to represent a one-dimensional vector of a sentence, it cannot show the importance of different words and does not consider the contextual relationship between words. Therefore, combined with the characteristics of the BOW, the representation model obtained by the preprocessing combination method was called Preprocessing Bag of Words (PBOW), and it is described as below.

$$\boldsymbol{D} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & \cdots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & d_{m3} & \cdots & d_{mn} \end{bmatrix} \tag{1}$$

In formula (1), m represents the total number of samples, and n represents the total number of features. Matrix $\boldsymbol{D}$ is a document-word matrix, *i.e.* a feature matrix, also known as a word frequency matrix. Specifically, each row of the matrix $\boldsymbol{D}$ represents a text, and each column represents a word that has appeared in the text, called a sample feature. d_{mn} represents the number of times the nth word appears in the mth document. The newly proposed preprocessing strategy filters text data, reduces redundant features and retains main features, thereby reducing feature matrix dimension.

This novel approach to deep analysis of text data is simpler and more efficient to implement. Therefore, this paper mainly utilizes the feature matrix obtained by preprocessing optimization as the dictionary matrix of Orthogonal Matching Pursuit (OMP) algorithm to realize feature selection and classification in an efficient and greedy iterative way [20]. Specifically, the main expression of the model is $\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}$, where $\boldsymbol{y} \in \mathbb{R}^m$ is the test sample, $\boldsymbol{D} \in \mathbb{R}^{m \times n}$ represents the dictionary, namely the PBOW feature matrix constructed in this paper, and $\boldsymbol{x} \in \mathbb{R}^n$ is the unknown sparse coefficient. In addition, this work introduces the objective function of ℓ_2-regularization to alleviate the overfitting problem caused by large sample feature size, the high dimensionality of text data and poor sparsity. The main implementation process of the classification algorithm combined with novel preprocessing strategies in this paper is as follows:

Input: $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{r}_0 = \boldsymbol{y}$, $\mathcal{L} = \varnothing$, λ, threshold ϵ.

Step 1: Implement a novel preprocessing strategy to obtain the PBOW matrix $\boldsymbol{D} = \left[\boldsymbol{D}_1, \boldsymbol{D}_2, \cdots, \boldsymbol{D}_c\right]$;

Step 2: Compute sparse subset: $j^i = \operatorname{argmax}_{j \notin \mathcal{L}} \left| \boldsymbol{x}^{\mathrm{T}} \boldsymbol{r}_{i-1} \right| \left(i = 1, 2, \cdots, c\right)$;

Step 3: Sparse index set: $\mathcal{L} = \mathcal{L} \cup j^i$;

Step 4: Solve for the sparse parameter: $\boldsymbol{\alpha}_i = \underset{\boldsymbol{x} \in \mathbb{R}^n, j \in \mathcal{L}}{\arg\min} \left\| \boldsymbol{y} - \boldsymbol{D}_i \boldsymbol{x} \right\|_2^2 + \lambda \left\| \boldsymbol{x} \right\|_2^2$;

Step 5: Calculate residuals: $\boldsymbol{r}_i = \left\| \boldsymbol{y} - \boldsymbol{D}_i \boldsymbol{\alpha}_i \right\|_2^2 \left(i = 1, 2, \cdots, c\right)$, if $\left\| \boldsymbol{r}_i \right\|_2^2 < \epsilon$ break else continue.

Output: $label\left(\boldsymbol{y}\right) = \arg\min_i \boldsymbol{r}_i$.

Where, regularization parameter $\lambda > 0$ is used to set the trade-off between the sparsity of the solution and the reconstruction error, and threshold $\epsilon = 10^{-6}$. Firstly, this algorithm utilizes the proposed preprocessing strategy to reduce the dimension of text data and obtains a new word frequency matrix vector $\boldsymbol{D}$. Secondly, it solves sparse subset j^i and gets sparse index set $\mathcal{L}$. Then, the OMP algorithm employs the training dictionary $\boldsymbol{D}$ to greedily select the dictionary atom $\boldsymbol{y}$ that best matches the test sample, thus solving the objective function to calculate the sparse coefficient $\boldsymbol{\alpha}_i$. Subsequently, it obtains the residual $\boldsymbol{r}_i$ by continuously selecting relevant atoms in the dictionary, and iterate step by step until the termination condition that the residual is less than ϵ is satisfied.

Finally, output the category $label(\boldsymbol{y})$ of the test sample $\boldsymbol{y}$.

4. Performance Evaluation

This section implements numerical tests for the news class dataset to demonstrate the effects of the three proposed preprocessing methods on decreasing the number of features and enhancing the accuracy and efficiency of text classification.

4.1. Datasets and Experimental Setup

This part is devoted to describing datasets and settings. It performed extensive experiments for each pair of a dataset and machine learning method. The four different sets contain the number of a feature of 17,000 - 21,000 word unigrams to determine a reasonable number of word unigrams for PBOW text presentation matrix. This study is generalized as a binary classification problem, where it classifies a document according to two related classes: 1) religion: soc.religion.christian vs. alt.atheism; 2) comp.sys: mac.hardware vs. ibm.pc.hardware; 3) sci: space vs. med and 4) rec.sport: hockey vs. baseball. Table 2 summarizes the dataset settings in detail.

This work selects four categories of religion, computer, science and sport from the 20 newsgroups and set the ratio of training to testing sets dataset to 4:1. Furthermore, this paper selects five baseline methods for comparison to illustrate the impact of the newly proposed method. Baseline model is the following.

- Basic preprocessing (NP_0): preprocessing methods are TK and LC, and the classifier is OMP.
- DecisionTree (DT): it summarizes decision rules from a variety of characteristic and label data and classifies them in a dendrogram structure.
- RandomForest (RF): represent the Bagging ensemble algorithm, all of its base evaluators are decision trees.
- AdaBoost (AB): an iterative algorithm whose core idea is to train different classifiers for the same training set, and then combine these weak classifiers to form a stronger final classifier.
- KNeighbors (KNN): implement the K-nearest neighbor algorithm, using Euclidean distance for measurement.

DT, RF, AB and KNN are all supervised classification methods in machine learning that are easy to understand. The NP_0 as a comparison algorithm in terms

Table 2. The number of train and test document in 20 new groups.

Category	Train	Test	Total	Vocabulary
religion	1079	717	1796	18,707
computer	1168	777	1945	17,128
science	1187	790	1977	24,398
sport	1197	796	1993	20,932

of efficiency and all base methods for the comparison in terms of accuracy to better measure results. Additionally，this work makes use of the PBOW as the text feature matrix and the OMP algorithm as the classifier. Finally, it performs data preprocessing in Pycharm and implements classifier in Matlab.

4.2. Preprocessing Results

This work utilizes a text from the 20 news groups dataset, which is an international standard dataset for text classification. It selects a sentence before preprocessing from this dataset, and it is "Answer: an especial witness-one who is supposed to be a personal witness. That means to be a true apostle, one must have Christ appear to them." More precisely, Table 3 exhibits the preprocessing results after performing the four preprocessing methods.

It is obviously that the original sentence has 31 features from Table 3. However, after performing the NP_0, NP_1, NP_2 and NP_3 methods, there are 14, 10, 15 and 7 features, separately. NP_0 is the baseline method for the comparison of preprocessing strategies. Hence, NP_3 reduces more useless features compared to NP_0 - NP_2. In addition, the proposed three preprocessing methods are of great significance in reducing the number of features, which contributes to enhance the efficiency and accuracy of text classification.

Table 4 exhibits the numbers of features after performing the four preprocessing methods on religion, computer, science and sport data from the 20 newsgroups dataset, where the numbers of features before the preprocessing are 31,727, 39,186, 26,568 and 35,199, separately. Obviously, the three methods proposed by us have a significant effect on decreasing the number of features. Particularly, NP_3 reduces the number of features to 14,994, 17,994, 11,552 and 12,837 for the four data, and compared with baseline method NP_0, the improvements are respectively 19.85%, 26.25%, 32.55% and 38.67%.

Table 3. Text data after performing preprocessing methods.

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

Table 4. The number of features after performing the four preprocessing strategies.

Category	NP_0	NP_1	NP_2	NP_3
religion	18,707	18,403	15,667	14,994
science	24,398	24,075	18,660	17,994
computer	17,128	16,761	12,179	11,552
sport	20,932	20,608	13,477	12,837

Table 5. The accuracy results of different text classification algorithms.

Category	DT	RF	AB	KNN	NP_0 + OMP	our method
religion	0.917*	0.904	0.916	0.869	0.919	**0.933**
computer	0.854	0.874*	0.843	0.825	0.889	**0.898**
science	0.917	0.935*	0.910	0.930	0.961	**0.962**
sport	0.893	0.910	0.913*	0.911	0.954	**0.962**

4.3. Model Accuracy

In this section, the accuracy indicator is to measure the effectiveness of text classification. On the basis of four preprocessing methods, this work applies OMP algorithm to text classification, and then compare the accuracy of different algorithms. Note that accuracy is a crucial evaluation metric in NLP.

Table 5 gives the text classification accuracy under different classification methods and preprocessing combination strategies. The underlined number is the minimum value, the value* is the supervised classifier with the highest accuracy, and the bold number is the best value. Obviously, our methods, including NP_1, NP_2 and NP_3, outperform supervised algorithms and NP_0 in terms of accuracy. Specifically, our novel preprocessing method improves the accuracy by 5.7% - 8.8% over the worst, 1.7% - 5.4% over the best supervised classifier, and 0.84% - 1.5% over the NP_0 method. Our preprocessing strategies achieve satisfactory text classification results and the intended experimental purpose compared with five baseline algorithms. This also fully shows that selecting the appropriate preprocessing method can effectively remove noisy data and reduce data redundancy.

4.4. Model Efficiency

In terms of efficiency, supervised learning algorithms only run in the Pycharm environment, the proposed method and NP_0 are implemented in MATLAB. Then this work only chooses the NP_0 method as the baseline method for the efficiency comparison to ensure the rationality of the efficiency comparison. This part analyzes the influence of different preprocessing combination strategies on text learning time.

The learning time, which is almost the running (both training and testing) time of the text classification algorithm, conducted over 50 independent running experiments in Figure 2. More importantly, the learning time of NP_1-NP_3 is much shorter than that of NP_0 except that the learning time of NP_1 is 0.13% higher than that of NP_0 for preprocessing the religion data. The main reason is that the new preprocessing strategies can noticeably reduce the number of features, consequently reducing the computational cost of the OMP algorithm. The results also demonstrate that NP_3 has a remarkable improvement in reducing the learning time, with 17.38%, 23.76%, 25.64% and 33.38% of learning time for preprocessing religion, science, computer and sport data, separately.

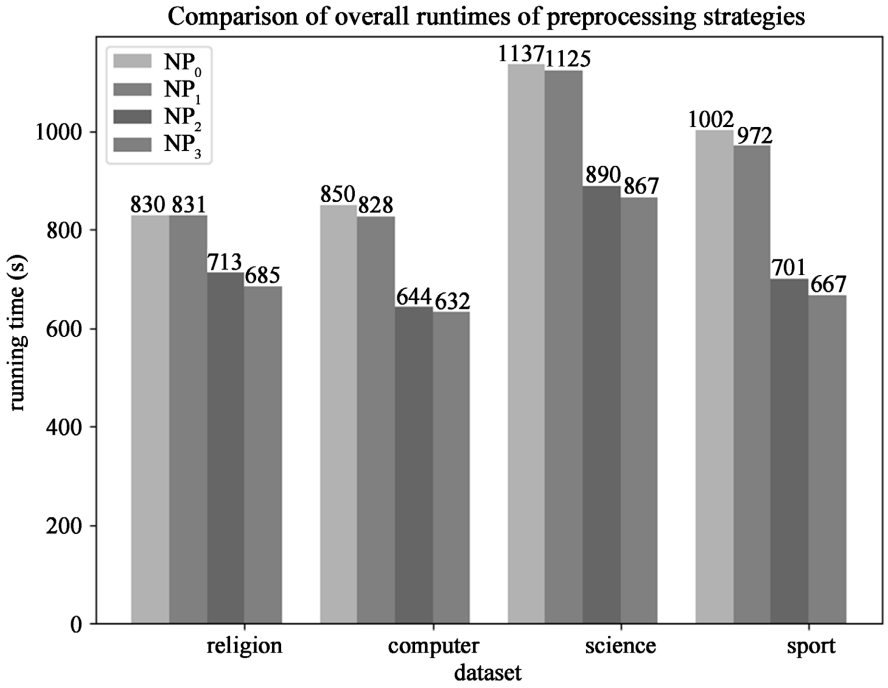

Figure 2. The learning time of proposed strategies.

5. Conclusion

This article conceived an effective and efficient preprocessing strategy with three methods for text classification. Based on the existing preprocessing methods, the novel method solves high-dimensional data problems and significantly reduces the number of text features, improves efficiency and enhances the accuracy. Further, this work will continue to improve the generality and robustness of the model.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] George, G., Haas, M.R. and Pentland, A. (2014) Big Data and Management. *Academy of Management Journal*, **57**, 321-326. https://doi.org/10.5465/amj.2014.4002

[2] Xu, C. (2018) A Novel Recommendation Method Based on Social Network Using Matrix Factorization Technique. *Information Processing & Management*, **54**, 463-474. https://doi.org/10.1016/j.ipm.2018.02.005

[3] Joachims, T. (2005) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Machine Learning*: *ECML*-98: 10*th European Conference on Machine Learning*, Chemnitz, 21-23 April 1998, 137-142. https://doi.org/10.1007/BFb0026683

[4] Guzella, T.S. and Caminhas, W.M. (2009) A Review of Machine Learning Approaches to Spam Filtering. *Expert Systems with Applications*, **36**, 10206-10222. https://doi.org/10.1016/j.eswa.2009.02.037

[5] Medhat, W., Hassan, A. and Korashy, H. (2014) Sentiment Analysis Algorithms and

Applications: A Survey. *Ain Shams Engineering Journal*, **5**, 1093-1113. https://doi.org/10.1016/j.asej.2014.04.011

[6] Joachims, T. (2002) Learning to Classify Text Using Support Vector Machines (Vol. 668). Springer Science & Business Media, Berlin. https://doi.org/10.1007/978-1-4615-0907-3

[7] Wang, D., Zhang, H., Liu, R., Liu, X. and Wang, J. (2016) Unsupervised Feature Selection through Gram-Schmidt Orthogonalization—A Word Co-Occurrence Perspective. *Neurocomputing*, **173**, 845-854. https://doi.org/10.1016/j.neucom.2015.08.038

[8] Skianis, K., Rousseau, F. and Vazirgiannis, M. (2016) Regularizing Text Categorization with Clusters of Words. *Proceedings of the* 2016 *Conference on Empirical Methods in Natural Language Processing*, Austin, 1-5 November 2016, 1827-1837. https://doi.org/10.18653/v1/D16-1188

[9] Abbasi, A., France, S., Zhang, Z. and Chen, H. (2010) Selecting Attributes for Sentiment Classification Using Feature Relation Networks. *IEEE Transactions on Knowledge and Data Engineering*, **23**, 447-462. https://doi.org/10.1109/TKDE.2010.110

[10] Mirończuk, M.M. and Protasiewicz, J. (2018) A Recent Overview of the State-of-the-Art Elements of Text Classification. *Expert Systems with Applications*, **106**, 36-54. https://doi.org/10.1016/j.eswa.2018.03.058

[11] Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D. and Saeed, J. (2020) A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction. *Journal of Applied Science and Technology Trends*, **1**, 56-70. https://doi.org/10.38094/jastt1224

[12] Joulin, A., Grave, E., Bojanowski, P. and Mikolov, T. (2016) Bag of Tricks for Efficient Text Classification. https://doi.org/10.18653/v1/E17-2068

[13] Joachims, T. (1996) A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Carnegie Mellon University, Pittsburgh.

[14] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space.

[15] Rehman, A., Javed, K. and Babri, H.A. (2017) Feature Selection Based on a Normalized Difference Measure for Text Classification. *Information Processing & Management*, **53**, 473-489. https://doi.org/10.1016/j.ipm.2016.12.004

[16] Cortes, C. and Vapnik, V. (1995) Support-Vector Networks. *Machine Learning*, **20**, 273-297. https://doi.org/10.1007/BF00994018

[17] Darwiche, A. (2010) Bayesian Networks. *Communications of the ACM*, **53**, 80-90. https://doi.org/10.1145/1859204.1859227

[18] Lai, S., Xu, L., Liu, K. and Zhao, J. (2015) Recurrent Convolutional Neural Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, **29**, 2267-2273. https://doi.org/10.1609/aaai.v29i1.9513

[19] Skianis, K., Tziortziotis, N. and Vazirgiannis, M. (2018) Orthogonal Matching Pursuit for Text Classification. https://doi.org/10.18653/v1/W18-6113

[20] Swirszcz, G., Abe, N. and Lozano, A.C. (2009) Grouped Orthogonal Matching Pursuit for Variable Selection and Prediction. *Proceedings of the* 22*nd International Conference on Neural Information Processing Systems*, Vancouver, 7-10 December 2009, 1150-1158.

[21] Chandrasekar, P. and Qian, K. (2016) The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier. 2016 *IEEE* 40*th Annual Computer Software and Applications Conference* (*COMPSAC*), Vol. 2, 618-619.

https://doi.org/10.1109/COMPSAC.2016.205

[22] Uysal, A.K. and Gunal, S. (2014) The Impact of Preprocessing on Text Classification. *Information Processing & Management*, **50**, 104-112. https://doi.org/10.1016/j.ipm.2013.08.006

[23] Gonçalves, C.A., Gonçalves, C.T., Camacho, R. and Oliveira, E.C. (2010) The Impact of Pre-Processing on the Classification of MEDLINE Documents. *Proceedings of the* 10*th International Workshop on Pattern Recognition in Information Systems*, *PRIS* 2010, Funchal, June 2010, 53-61.

[24] HaCohen-Kerner, Y., Miller, D. and Yigal, Y. (2020) The Influence of Preprocessing on Text Classification Using a Bag-of-Words Representation. *PLOS ONE*, **15**, e0232525. https://doi.org/10.1371/journal.pone.0232525

[25] Naseem, U., Razzak, I. and Eklund, P.W. (2021) A Survey of Pre-Processing Techniques to Improve Short-Text Quality: A Case Study on Hate Speech Detection on Twitter. *Multimedia Tools and Applications*, **80**, 35239-35266. https://doi.org/10.1007/s11042-020-10082-6