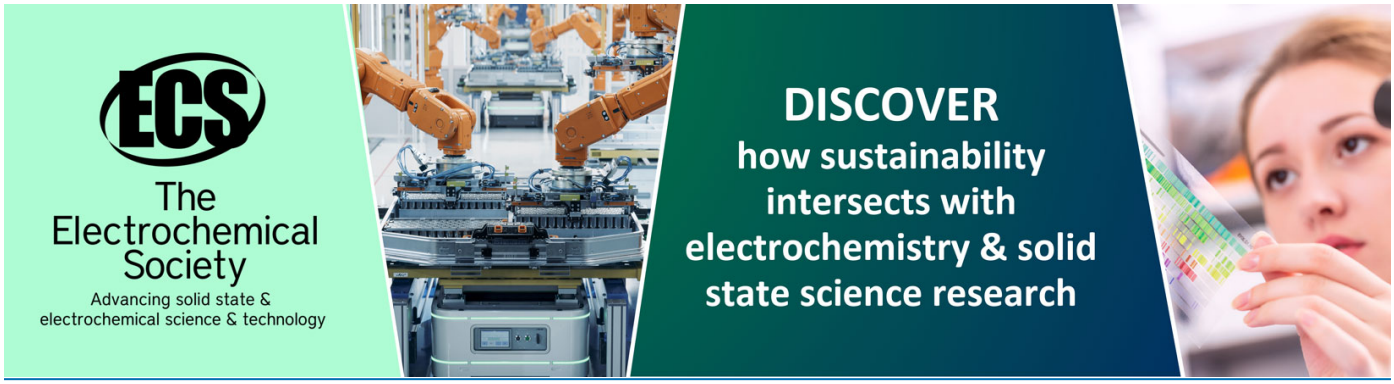
Comparisons and Selections of Features and Classifiers for Short Text Classification



IOP Conf. Series: Materials Science and Engineering **261** (2017) 012018 doi:10.1088/1757-899X/261/1/012018

Comparisons and Selections of Features and Classifiers for Short Text Classification

Ye Wang[1, a], Zhi Zhou[2, b], Shan Jin[1,c], Debin Liu[2, d] and Mi Lu[1, e]

[1] Department of Electrical and Computer Engineering, Texas A&M University
[2]SocialCredits, Ltd

E-mail: [a]wangye0523@tamu.edu, [b]zhi.zhou@socialcredits.cn, [c]jinshan@tamu.edu, [d]debin.liu@socialcredits.cn, [e]mlu@ece.tamu.edu

Abstract. Short text is considerably different from traditional long text documents due to its shortness and conciseness, which somehow hinders the applications of conventional machine learning and data mining algorithms in short text classification. According to traditional artificial intelligence methods, we divide short text classification into three steps, namely preprocessing, feature selection and classifier comparison. In this paper, we have illustrated step-by-step how we approach our goals. Specifically, in feature selection, we compared the performance and robustness of the four methods of one-hot encoding, tf-idf weighting, word2vec and paragraph2vec, and in the classification part, we deliberately chose and compared Naive Bayes, Logistic Regression, Support Vector Machine, K-nearest Neighbor and Decision Tree as our classifiers. Then, we compared and analysed the classifiers horizontally with each other and vertically with feature selections. Regarding the datasets, we crawled more than 400,000 short text files from Shanghai and Shenzhen Stock Exchanges and manually labeled them into two classes, the big and the small. There are eight labels in the big class, and 59 labels in the small class.

1. Task and problem

Short text classification is unlike traditional long text documents, due to its own characteristics in terms of shortness and conciseness. The objective of this paper is to compare existing methods for better short-text classification.

Each short text is less than 20 words, and we manually labeled it into two class of tags, there are eight types of labels in the big class and fifty-nine types of labels in the small class. The total number of short text files is about 400,000, and each tag's short text files are equal. In other words, we can feed about 50,000 short text files to train each label in the big class, and around 6,700 files to train each label in the small class. There exist some challenges specific to short text classification. Shortness entails a lack of information and conciseness is a synonym for simplicity, which can cause confusion when we try to classify 59 labels with files of no more than 20 words. Besides, we have more than 400,000 short text files, which amount to approximately 5,349,348 words and would certainly create a lot of sparsity in the

vectorization process. Sparsity would in turn lead to several problems such as data redundancy, sparsity matrix, etc. We will introduce them in next few sections.

Company Announcement	Big Class	Small Class
KPC Pharmaceuticals, Inc.: Announcement on the Progress of the Restricted Stock Grant in the 2016 Equity Incentive Plan	Major Events	Equity Incentive Compensation
ENN Ecological Holdings Co. Ltd.: 2016 Semi annual report	Financial Report	Semi Annual Report
BOETECHNOLOGYGROUPCO.,LTD: Announcement on repurchasing part of company shares and canceling the creditor	Equity and Capital Stock	Repurchasing Equity
Total: 409,871 short text announcement	8 labels	59 labels

Figure 1. Announcement examples in Dataset

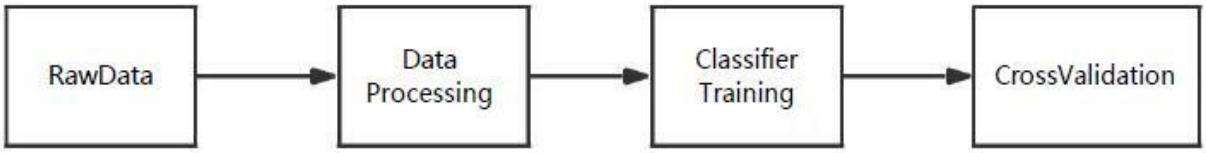


Figure 2. Basic Flow Chart

The rest of this paper is organized as follows: Section II presents a high-level description of the approach in terms of data processing and classifier training. Section III introduces the system implementing our task and how we handle the challenges. Section IV discusses the experiment results, and the paper concludes with Section V.

2. High-level description of approach

As can be seen, Figure 1 lists some examples of our data samples which will be used later. Figure 2 shows the basic flow chart, which can be regarded as top level description of our approach. In this section, we will discuss in details about the entire process.

2.1. Data processing

There are many different data processing techniques. We should be aware of their characteristics and choose the appropriate method.

2.1.1. SEGMENTATION. According to [1], segmentation is becoming increasingly more important in Chinese, Japanese and many other Asian language processing tasks. Unlike English, Chinese words are not delimited by whitespace characters, so word segmentation is a fundamental first step in processing these languages. Several algorithms have been proposed for Chinese word segmentation [2], and the study in automatic Chinese word segmentation has made significant progress in recent years. For the purpose of our study, we just chose the currently most popular segmentation method which is based on prefix trie and the Viterbi algorithm.

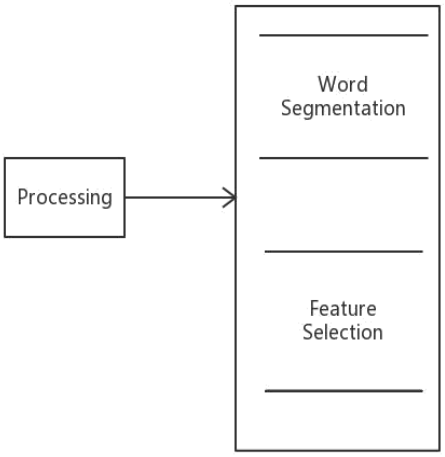


Figure 3. Processing the Data

2.1.2. FEATURE SELECTION. Figure 3 shows the data processing flow chart. Once we get the segmented words, we can convert them into a vector matrix for later training. This process is called word embedding, or distributional models. The reason for constructing such a vector matrix is that we can utilize the term-context matrix to represent the short text, which is much simpler for training purposes. There are many ways to construct the vectors, such as sparse vectors and dense vectors. Sparse vectors have most elements equal to zero and lengths of about 20,000 to 50,000, which will be very time-consuming computationally, while dense vectors are constructed in 100-500 dimensions, so are much faster than sparse vectors when used in training and classifications. Dense vectors may also better capture synonymies than sparse vectors use [3]. Moreover, we employed two methods for the sparse vector construction, i.e. counter vectorizer and term frequency-inverse document frequency (tf-idf). Counter vectorizer is also called one-hot coding, which is applied to categorical features.

Categorical features are "attribute-value" pairs where the value is restricted to a list of discrete possibilities without ordering. Counter vectorizer is like a raw vectorizer, while tf-idf is more refined, since it is a numerical statistic that is intended to reflect how important a word is to the short text in our collections. It is often used as a weighted factor in application. The key characteristic of tf-idf is that it increases proportionally with the frequency of a word appearing in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general [4].

Compared with sparse vectors, dense vectors are more popular because they are shorter and yet more meaningful. There are three popular methods for constructing dense vectors, i.e. singular value decomposition, neural language models and Brown clustering. Here we would like to focus on the neural language model, which is the state-of-the-art method for dense vector construction [3] [5]. There are two type of neural language models, Skip-gram and continuous bag of words (CBOW), which are also collectively called word2vec models, as shown in Figure 4. [6] provides a detailed explanation of the two models. In short, the CBOW architecture predicts the current word based on the context while skip-gram predicts surrounding words given the current word. One advantage of dense vectors is that we can get a short and yet meaningful vector to represent each word. Also, it has the superior characteristic that the matrix of similar words also has a closer distance, which is helpful to constructing the thesaurus. Moreover, [7] enhanced his work of word embedding and proposed a novel model called paragraph2vec (or doc2vec) as shown in Figure 5. This method trains the entire document as a vector matrix, while for word2vec, the basic idea is to predict the word. Similar to word2vec, in the training of the document

vector we need to go through the whole text. We have also implemented doc2vec in this paper and will compare it with other feature selection methods in the results and analyses part.

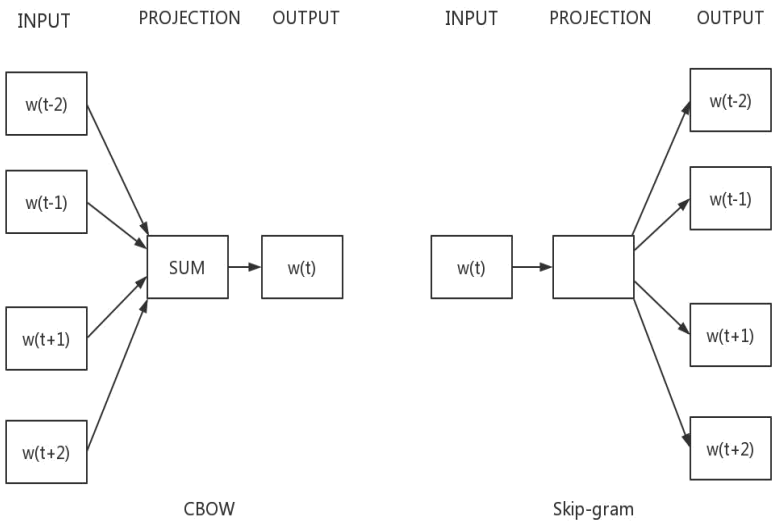


Figure 4. Two Models of word2vec

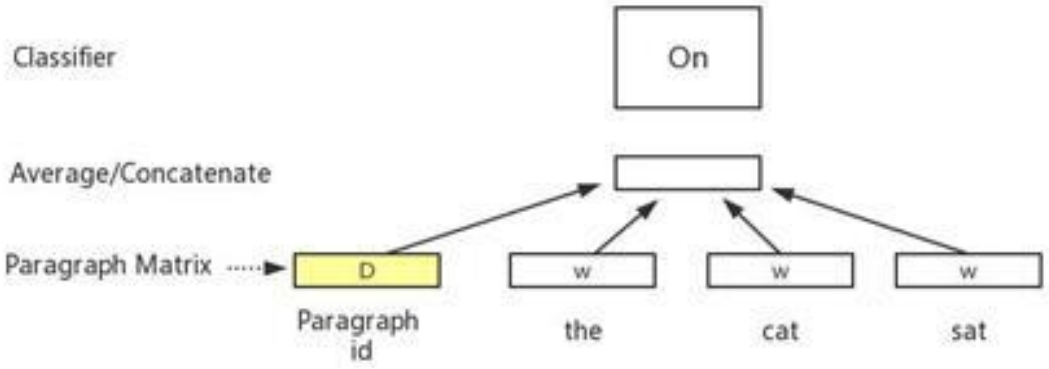


Figure 5. Paragraph2vec Model

2.2. CLASSIFICATION

Once the feature is selected, its time to train the classifier. Classification is one of the most important steps in all machine learning's tasks. Classification is the problem of identifying to which set or category a new observation belongs, on the basis of a training set of data containing observations whose class is known. Since we already have labeled all the instances, we only need to choose supervised learning classifiers. Among the various learning algorithms, we cannot simply decide the best one before experimenting and comparing some of them. Hence we selected several popular classifiers, including naive Bayes (NB), decision tree (DT), k-nearest neighbor (KNN), logistic regression (LR) and support vector classifier (SVC), and applied them with both the big and small classes of labels. Each classifier has its own advantages and disadvantages.

2.2.1. NAIVE BAYES. Naive Bayes methods build upon the famous Bayes' theorem with the (not so) "naive" assumption of independence between each pair of features. The NB method is very low time complexity, and its assumption usually works quite well in some real-world situations such as spam filtering and document classification. As a consequence of the decoupling of the conditional probability distributions of different features, the probability distribution of each feature can be independently estimated as one-dimensional distribution, which in turn helps alleviate problems stemming from the curse of dimensionality. In this study we compared both Gaussian (GNB) and multinomial (MNB) naive Bayes classifiers. When dealing with dense vectors, we treat those data as continuous data, and when dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

2.2.2. LOGISTIC REGRESSION. Multinomial logistic regression is known by a variety of other names, including polytomous LR, multi-class LR, softmax regression, multinomial logit, maximum entropy (MaxEnt) classifier, and conditional maximum entropy model. In fact, multinomial logistic regression is a classification method that generalizes logistic regression to multiple-class problems. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

2.2.3. SUPPORT VECTOR MACHINE. Support Vector Machine (SVM) is widely used among classification, regression and even outlier detection. The advantage of SVM is obvious: First, it is very effective not only in high dimensional spaces, but also in cases whether the number of dimensions is greater than the number of samples. Second, it is considerably memory efficient due to its own advantage of kernel mapping to high-dimensional feature spaces [8]. Just as every coin has two sides, the disadvantage of SVM is that if the number of feature is much greater than the number of samples, the method is likely to give poor performance. A linear support vector classifier (SVC) is used in this paper.

3. SPECIFIC SYSTEM IMPLEMENTATIONS

The whole structure of our system is divided into four parts as we illustrated before: Getting raw data, processing data, training classifier and cross validation. We implemented it by Python2.7 with some open source APIs like Scikit-learn [9] and Gensim [10].

4. EXPERIMENT

Several classifiers have been trained to classify Chinese short text les, including GNB, MNB, SVC, LR, KNN and DT [11]. However, we will only present the experiment results of MNB, SVC and LR in this section, since they are much better than those of the other classifiers.

Figure 6 to Figure 10 show the 5-fold cross-validation results for each of the three classifiers. In the cross-validation tests we have used four features, i.e. word2vec, doc2vec, tf-idf and counter vectorizers. The performances of the last two are similar so we treat them indifferently as a single tf-idf/counter feature. In addition, we have filtered stop words in each experiment, which also slightly improves the accuracy.

From the tables we can see that in all cases, the tf-idf/counter feature has the highest accuracy, while word2vec next, and doc2vec the lowest. The feature doc2vec produces the worst result in any circumstance, even much worse than plain guessing, which is different from the long text classification results [5]. We also get different results for the big and small classes of labels. The small class generally results in higher accuracy than the big class, which is counterintuitive and needs further investigation.

big				small			
Data = ½				Data = ½			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	46.5454798	30.979916	67.9974846	MNB	50.5129724	3.9339202	71.8358194
SVC	56.523723	20.7917638	70.6630796	SVC	77.1284876	2.055327	83.5525026
LR	64.3266498	30.979916	70.7194048	LR	81.4567048	2.4392982	84.218733
Data = 1/5				Data = 1/5			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	46.020836	30.9789578	66.8190094	MNB	47.8368036	3.6108782	71.3708568
SVC	58.7445636	22.0106284	69.811158	SVC	70.3710892	2.7923356	83.9617922
LR	64.9365838	30.9789578	70.0025738	LR	81.5665502	2.438548	84.1286016
Data = 1/10				Data = 1/10			
Classifier	Accuracy, %			Classifier	Accuracy, %		
	word2vec	doc2vec	tf-idf/counter		word2vec	doc2vec	tf-idf/counter
MNB	45.997871	30.979801	64.6257898	MNB	47.0733842	3.5133214	71.528666
SVC	56.9952578	25.7708636	69.0387578	SVC	60.1547316	2.1768262	83.9285676
LR	64.455278	30.979801	68.6656358	LR	81.3346074	2.4397412	83.5524258

Figure 6. Result

Figure 7. 1/2 datasets of overall

Figure 8. 1/2 datasets of word2vec

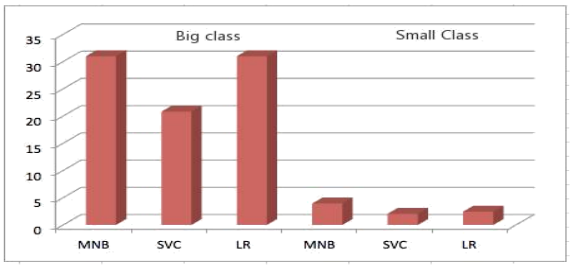


Figure 9. 1/2 datasets of doc2vec

Figure 10. 1/2 datasets of tf-idf/counter

A comparison of different classifiers would show that with the tf-idf/counters feature, LR and SVC are much better than MNB, and the results of the two are comparable with each other. This may not be the case when other features are used. While a high accuracy is expected for the SVC because of the use of kernel function, it is a little surprising that the overall highest accuracy 84.22% is associated with LR.

Additionally we have tried to change the size of the dataset, and it seems increasing the size of the dataset and raises the accuracy, but this impact is not significant.

5. CONCLUSIONS

In this paper, we have demonstrated the classification of Chinese short text, in the context of public financial documents. Different features and classifiers are applied and compared, and the cross-validation results show that logistic regression and support vector classifier with the tf-idf or CounterVectorizer feature attain the highest accuracy and are the most stable in all circumstances. We have also observed some distinct characteristics of the short text classification problem like that the small class produces better results than big class, and that doc2vec always doesn't work well.

References

[1] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of machine Learning research 3(Jan), 993{1022 (2003)
[2] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G.: API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD Workshop: Languages for Data Mining and Machine Learning. pp. 108-122 (2013)
[3] Hand, D.J., Mannila, H., Smyth, P.: Principles of data mining. MIT press (2001)
[4] Huang, C., Zhao, H.: Chinese word segmentation: A decade review. Journal of Chinese Information Processing 21(3), 8{20 (2007)
[5] Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. vol. 14, pp. 1188{1196 (2014)
[6] Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. pp. 1{8. IEEE (2008)
[7] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111-3119 (2013)
[9] Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random elds. In: Proceedings of the 20th international conference on Computational Linguistics. p. 562. Association for Computational Linguistics (2004)
[10] Rehurek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45{50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
[11] Rokach, L., Maimon, O.: Data mining with decision trees: theory and applications. World scientific (2014)