Compressing the Embedding Matrix by a Dictionary Screening Approach in Text Classification

Jing Zhou[1(✉)], Xinru Jing[1], Muyu Liu[1], and Hansheng Wang[2]

[1] Center for Applied Statistics and School of Statistics, Renmin University of China, Beijing, China
jing.zhou@ruc.edu.cn, jingxinru0207@ruc.edu.cn, Muyu.Liu@ruc.edu.cn
[2] Guanghua School of Management, Peking University, Beijing, China
hansheng@pku.edu.cn

Abstract. In this paper, we propose a dictionary screening method for embedding compression in text classification. The key point is to evaluate the importance of each keyword in the dictionary. To this end, we first train a pre-specified recurrent neural network-based model using a full dictionary. This leads to a benchmark model, which we use to obtain the predicted class probabilities for each sample in a dataset. Next, to evaluate the impact of each keyword in affecting the predicted class probabilities, we develop a novel method for assessing the importance of each keyword in a dictionary. Consequently, each keyword can be screened, and only the most important keywords are reserved. With these screened keywords, a new dictionary with a considerably reduced size can be constructed. Accordingly, the original text sequence can be substantially compressed. The proposed method leads to significant reductions in terms of parameters, average text sequence, and dictionary size. Meanwhile, the prediction power remains very competitive compared to the benchmark model. Extensive numerical studies are presented to demonstrate the empirical performance of the proposed method.

Keywords: Embedding Compression · Dictionary Screening · Text Classification

1 Introduction

Over the past few decades, natural language processing (NLP) has become a popular research field. Among the applications of this filed, text classification is considered to be a problem of great importance. Many successful applications exist, such as news classification [14], topic labeling [2], sentiment analysis [8], and many others. Note that, for most text classification tasks, the inputs are documents constructed from word sequences. Therefore, a standard RNN-based model can be readily applied. Remarkably, the model complexity of an RNN-based model is mainly determined by two factors. They are, respectively, the

H. Kashima et al. (Eds.): PAKDD 2023, LNAI 13935, pp. 457–468, 2023.
https://doi.org/10.1007/978-3-031-33374-3_36

model structure and dictionary size. Obviously, large dictionaries lead to more complicated RNN models with a large number of parameters. To illustrate this idea, consider, for example, a standard RNN model with one embedding layer and one recurrent hidden layer. The total number of parameters needed is $dk + 2k^2$, where d is the dictionary size and k is the dimension for both the hidden layer and embedding space. In the AG's News dataset [18], for instance, the total number of keywords contained in the dictionary could be as large as $d = 93,994$. If, following [3], we set the dimensions of both the hidden layer and embedding space to $k = 128$, then the total number of parameters is more than 12 million. As to be demonstrated later, we find that the dictionary size can be effectively reduced to be $d = 3,000$, if the method developed in this work is used. As a result, the total number of parameters can be reduced to about 0.58 million. This accounts for only about 4.76% of the original model complexity with limited sacrifice of prediction accuracy.

For text classification, most of the prior work focuses on compressing the embedding matrix. For example, a number of researchers have adopted hashing or quantization-based approaches to compress the embedding matrix [9,13,15]. [1] proposed a low rank matrix factorization for the embedding layer. In addition to compressing the embedding matrix, there is another branch of research that shows training with character-level inputs can achieve several benefits over word-level approaches, and it does so with fewer parameters [17]. For a more detailed review paper, we refer to [5]. Despite the excellent research that has been done on model compressing, it seems that most studies focus on simplifying the model structure in one way or another. Little research has been done on dictionary screening. As discussed in previous paragraph, we can see that the size of the dictionary can have an important impact on model size. As a consequence, we are motivated to fill this gap by proposing a dictionary screening method for text classification applications.

The proposed dictionary screening method aims to exclude less useful keywords from a dictionary. It can be used to effectively reduce dictionary size, leading to a significant reduction in model complexity. Specifically, we develop here a novel dictionary screening method as follows. First, we train a pre-specified RNN-based model using the full dictionary on a training dataset. This leads to a benchmark model. With the help of the benchmark model, we obtain the predicted class probabilities for every sample in the validation set. Next, for a given keyword in the dictionary, we consider whether it should be excluded from the dictionary. Obviously, keywords that significantly impact the predicted class probabilities should be kept, while those that do not should be excluded. Thus, the key here is determining how to evaluate the impact of the target keyword in affecting the estimated class probabilities.

To this end, for each document, we replace the target keyword with a meaningless substitute, which is often an empty space. By doing so, the input of the target keyword is excluded. We then apply the pretrained benchmark model to this new document. This leads to a new set of estimated class probabilities. Thereafter, for each document, we obtain two sets of estimated class probabil-

ities. One is computed based on the full dictionary, and the other is computed based on the dictionary with the target keyword excluded. Next, the difference between the two sets of probability estimators is evaluated and summarized. Keywords with large differences in class probability estimators should be kept. By selecting an appropriate threshold value, a new dictionary with a substantially compressed size can be obtained. Finally, with the compressed dictionary, each document can be re-constructed. The associated RNN-based model can be re-trained on the re-constructed documents, and its prediction accuracy can be evaluated. Our extensive numerical experiments suggest that the proposed method can compress the parameter quantity by more than 90%, on average, with little accuracy sacrificed.

The main contribution of our work is the development of a compression method for text classification using dictionary screening. There has been relatively little work on compressing dictionary size in the previous literature. A second contribution is that we provide a novel method for evaluating the importance of the keywords in a dictionary. We empirically show that our method outperforms popular baselines like term frequency-inverse document frequency (TF-IDF) and the t-test for keyword importance analysis.

2 Methodology

2.1 Problem Set up

Let $\mathcal{D} = \{w_d : 0 \leq d \leq D\}$ be a dictionary containing a total of D keywords with w_d representing the dth keyword. We define w_0 to be an empty space. Then, assume a total of N documents indexed by $1 \leq i \leq N$. Let $X_i = \{X_{it} \in \mathcal{D} : 1 \leq t \leq T\}$ be the ith document. The document is constructed by a sequence of keywords, X_{it}, which is indexed by t and is generated from $\mathcal{D}$. If the actual document length, T^*, is less than T, we then define $X_{it} = w_0$ for $T^* < t \leq T$. Next, let $Y_i \in \{1, 2, \cdots, K\}$ be the class label associated with the ith document. The goal is then to train a classifier so that we can accurately predict Y_i. To this end, various deep learning models can be used. For illustration purposes, we consider here a simple RNN model with four layers: one input layer of a word sequence with dimension T, one embedding layer with $d_1 = 128$ hidden nodes, one simple RNN layer with $d_2 = 64$ hidden nodes, and one fully connected layer with K nodes (i.e., the number of class labels). Suppose the dictionary size is $D = 10,000$ and the number of class labels is $K = 10$, then for the above simple RNN model, the total number of parameters is given by $df_a = 10,000 \times 128 + 64 \times 128 + 64 \times 64 + 64 \times 10 = 1,292,928$ with the bias term ignored. However, this number will be much reduced if the dictionary size can be significantly decreased. For example, the total number of parameters will be reduced to $df_b = 1,000 \times 128 + 64 \times 128 + 64 \times 64 + 64 \times 10 = 140,928$ if $D = 1,000$. This represents a model complexity reduction as large as $(1 - df_b/df_a) \times 100\% = 89.1\%$. We are then motivated to develop a method for dictionary screening.

2.2 Dictionary Screening

As discussed in the introduction, one of the key tasks for dictionary screening is to evaluate the importance of each keyword in the dictionary. We can formulate the problem as follows. First, we train a pre-specified RNN-based model using the full dictionary, $\mathcal{D}$, on the training dataset. Mathematically, we can write this model as $f(X_i, \theta) = \{f_k(X_i, \theta)\} \in \mathbb{R}^K$, where X_i is the input document, with θ as the unknown parameters that need to be estimated. Note that $f(X_i, \theta)$ is a K-dimensional vector, its kth element, $f_k(X_i, \theta) \in [0, 1]$, is a theoretical assumed function to approximate the class probability. That is, $P(Y_i = k|X_i) \approx f_k(X_i, \theta)$. By the universal approximation theorem [4,7], we know that this approximation can be arbitrarily accurate as long as the approximation function, $f(\cdot, \theta)$, can be sufficiently flexible. To estimate θ, an approximately defined loss function (e.g., the categorical cross entropy) is usually used. Denote the loss function as $\mathcal{L}_N(\theta) = N^{-1} \sum_{i=1}^{N} \ell(X_i, \theta)$, where $\ell(X_i, \theta)$ is the loss function evaluated on the ith document. The parameter estimators can then be obtained as $\hat{\theta} = \text{argmax}\mathcal{L}_N(\theta)$. This leads to the pretrained model as $f(X_i, \hat{\theta})$, which serves as the benchmark model.

Next, with the help of the pretrained model, we consider how to evaluate the importance of each keyword in $\mathcal{D}$. Specifically, consider the dth keyword, w_d, in $\mathcal{D}$ with $1 \le d \le D$. Define $\mathcal{S}_{\mathcal{F}} = \{1, 2, \cdots, N\}$ as the indices for the full training document. Then, for every $i \in \mathcal{S}_{\mathcal{F}}$, we compute its estimated class probability vector as $\widehat{p}_i = f(X_i, \hat{\theta})$. Next, for the document $X_i = \{X_{it} \in \mathcal{D} : 1 \le t \le T\}$, we generate a document copy as $X_i^{(d)} = \{X_{it}^{(d)} \in \mathcal{D} : 1 \le t \le T\}$, where $X_{it}^{(d)} = X_{it}$ if $X_{it} \ne w_d$, and $X_{it}^{(d)} = w_0$ if $X_{it} = w_d$. In other words, $X_i^{(d)}$ is a document that is almost the same as X_i. The only difference is that keyword w_d is replaced by an empty space, w_0. We next apply the benchmark model to $X_i^{(d)}$, so an update class probability vector, $\widehat{p}_i^{(d)} = f(X_i^{(d)}, \hat{\theta})$, can be obtained. The difference between $\widehat{p}_i$ and $\widehat{p}_i^{(d)}$ is evaluated by their ℓ_2-distance as $||\widehat{p}_i - \widehat{p}_i^{(d)}||^2$. We then summarize the difference for every $w_d \in \mathcal{D}$ as $\widehat{\lambda}(d) = |\mathcal{S}_{\mathcal{F}}|^{-1} \sum_{i \in \mathcal{S}_{\mathcal{F}}} ||\widehat{p}_i - \widehat{p}_i^{(d)}||^2$, where $|\mathcal{S}_{\mathcal{F}}|$ is the size of $\mathcal{S}_{\mathcal{F}}$. This is further treated as the important score for each keyword, $w_d \in \mathcal{D}$. Because this important score is obtained by evaluating the differences in class probability estimators, we name it as the CPE method for simplicity.

Finally, with a carefully selected threshold value, λ, a new dictionary can be constructed as $\mathcal{D}_\lambda = \{w_d \in \mathcal{D} : \widehat{\lambda}(d) \ge \lambda\} \cup \{w_0\}$. To this end, each document X_i can be reconstructed as $X_{\lambda_i} = \{X_{\lambda_i t} \in \mathcal{D}_\lambda : 1 \le t \le T\}$, where $X_{\lambda_i t} = X_{it}$ if $X_{it} \in \mathcal{D}_\lambda$, and $X_{\lambda_i t} = w_0$ otherwise. Then, by replacing X_is in the loss function with X_{λ_i}, a new set of parameter estimators can be obtained as $\widehat{\theta}_\lambda = \text{argmax}\mathcal{L}_\lambda(\theta)$, where $\mathcal{L}_\lambda(\theta) = N^{-1} \sum_{i=1}^{N} \ell(X_{\lambda_i}, \theta)$. Once $\widehat{\theta}_\lambda$ is obtained, the prediction accuracy of the resulting model, $f(X_{\lambda_i}, \widehat{\theta}_\lambda)$, can be evaluated on the testing dataset. Thereafter, the resulting model, $f(X_{\lambda_i}, \widehat{\theta}_\lambda)$, serves as the reduced model after applying dictionary screening. The algorithm details are presented as follows.

Algorithm 1: Dictionary screening method

Procedure1 `Train a benchmark model:`

Input : Document $X_i = \{X_{it} \in \mathcal{D} : 1 \leq t \leq T\}$; $Y_i \in \{1, 2, \cdots, K\}$ for $1 \leq i \leq N$

Output: Benchmark model $f(X_i, \hat{\theta})$ with estimated parameters $\hat{\theta}$ for $1 \leq i \leq N$

$\hat{\theta} = argmax\mathcal{L}_N(\theta)$ with $\mathcal{L}_N(\theta) = N^{-1}\sum_{i\in 1}^{N} \ell(X_i, \theta)$

where$\ell(X_i, \theta)$ is the loss evaluated on the ith document

Procedure2 `Evaluate the importance of each keyword in` $\mathcal{D}$**:**

Input : $\mathcal{D} = \{w_d : 0 \leq d \leq D\}$, $f(X_i, \hat{\theta})$ for $1 \leq i \leq N$

Output: $\mathcal{D}_\lambda = \{w_d \in \mathcal{D} : \widehat{\lambda}(d) \geq \lambda\} \cup \{w_0\}$

for $d = 1$ *to* D **do**

  for $i \in \mathcal{S}_\mathcal{F}$,$\mathcal{S}_\mathcal{F} = \{1, 2, \cdots, N\}$ *is the indices for the full training document.* **do**

    $X_i = \{X_{it} \in \mathcal{D} : 1 \leq t \leq T\}$

    Generate a copy as $X_i^{(d)} = \{X_{it}^{(d)} \in \mathcal{D} : 1 \leq t \leq T\}$ following:

    If $X_{it} \neq w_d$ then $X_{it}^{(d)} = X_{it}$ else $X_{it}^{(d)} = w_0$

    Uncompressed class probability : $\widehat{p}_i = f(X_i, \hat{\theta})$

    Compressed class probability : $\widehat{p}_i^{(d)} = f(X_i^{(d)}, \hat{\theta})$

    Difference computed by ℓ_2-distance as $||\widehat{p}_i - \widehat{p}_i^{(d)}||^2$

  end

  Summarize the difference:

  $\widehat{\lambda}(d) = |\mathcal{S}_\mathcal{F}|^{-1}\sum_{i\in\mathcal{S}_\mathcal{F}} ||\widehat{p}_i - \widehat{p}_i^{(d)}||^2$ where $|\mathcal{S}_\mathcal{F}|$ is the size of $\mathcal{S}_\mathcal{F}$

end

Select a threshold value λ to finally obtain

$\mathcal{D}_\lambda = \{w_d \in \mathcal{D} : \widehat{\lambda}(d) \geq \lambda\} \cup \{w_0\}$

3 Experiments

3.1 Task Description and Datasets

To demonstrate its empirical performance, the proposed dictionary screening method is evaluated on four large-scale datasets covering various text classification tasks. These are, respectively, news classification (AG's News and Sougou News), sentiment analysis (Amazon Review Polarity, ARP), and entity classification (DBPedia). These datasets are popularly studied in previous literature [17,18]. Summary statistics of the four large-scale datasets are presented in Table 1. For all the datasets (except for Sougou News), the sample size of each category is equal in both the training and testing sets. Take AG's News for example, it has 30,000 samples and 1,900 samples per class in the training set and testing set, respectively. For more detailed information about the four datasets, see [18]. It should be noted that, to make the experiments more diverse,

the Sougou News data used in this paper are different from those in [18]. Particularly, we used the original Chinese characters of Sougou News to test the proposed method, not the Pinyin style used in [18]'s work. Moreover, unlike the other three datasets, the sample size of each category (e.g., sports, entertainment, business, and the Internet) in Sougou News is not equal. The proportions of the four categories in the training and testing sets are 48%, 15%, 25%, and 12%, respectively.

Table 1. Summary of four large-scale datasets.

Dataset	Classes	Task	TrainingSize	TestingSize
AG's News	4	news classification	120,000	7,600
Sougou News	4	news classification	63,146	15,787
DBPedia	14	entity classification	560,000	70,000
Amazon Review Polarity	2	sentiment analysis	3,600,000	400,000

3.2 Model Settings

We consider here two different types of deep learning models for text classification. They are, respectively, TextCNN [10] and TextBiLSTM [11]. We follow their network structures but with some modifications to adapt to our experiments. In the task of text classification, the input is text sequence X_i with length T. It should be noted that T is different for different datasets. In the current experiment, to train the benchmark models, T is set to 60, 300, 50, and 100 for AG's News, Sougou News, DBPedia, and Amazon Review Polarity, respectively. Practically, each keyword $w_d \in \mathcal{D}$ in the text sequence will be converted to a high dimensional vector of d_1 via an embedding layer [12]. For all three models, the embedding size, d_1, is set to 128. Next, we briefly describe the construction details for the two models.

TextCNN. After the embedding layer, we use three convolutional layers to extract text information. Each convolutional layer has $d_1 = 128$ filters with kernel size $k \in \{3, 4, 5\}$, followed by a max pooling with receptive field size $r = 1$. Rectified linear units (ReLUs) [6] are used as activation functions in the convolutional layers. Then, we concatenate the max pooling results of the three layers and pass it to the final dense layer through a softmax function for classification.

TextBiLSTM. The bidirectional LSTM (Bi-LSTM) can be seen as an improved version of the LSTM. This model structure can consider not only forward encoded information, but also reverse encoded information [11]. We apply a Bi-LSTM layer with the hidden states dimension, d_2, set to 128. We then use the representation obtained from the final timestep (e.g., X_{iT}) of the Bi-LSTM layer and pass it through a softmax function for text classification.

3.3 Tuning Parameter Specification

The implementation of the proposed dictionary screening method involves a tuning parameter, which is the threshold value λ. For a given classification model, this tuning parameter should be carefully selected to achieve the best empirical performance. Here, the best empirical performance means that the proposed dictionary screening method can reduce the number of model parameters as much as possible under the condition of ensuring little or no loss of accuracy. Generally, the larger the λ value is, the smaller the size of the screened dictionary, and thus the higher the reduction rate that can be achieved. However, considerable prediction accuracy might be lost. In our experiments, different λ values indicate that different numbers of keywords can be reserved for subsequent text compression. To this end, for analysis simplicity, we investigate the number of important keywords reserved, denoted as K. Specifically, we rearrange the keywords in descending order according to the importance score (e.g., $\widehat{\lambda}(d)$), and we select the top 1000, 3000, and 5000 keywords, respectively. That is, $K = \{1000, 3000, 5000\}$. Therefore, we can evaluate the impact of different tuning parameters on the performance of the proposed dictionary screening method.

3.4 Competing Methods

For comparison purposes, two other methods for evaluating the importance of the keywords in a dictionary are studied. The first one is to calculate the TF-IDF [16] value of each keyword $w_d \in \mathcal{D}$. For the kth keyword, we use the word counts as the term-frequency (TF). The inverse document frequency (IDF) is the logarithm of the division between the total number of documents and the number of documents with the kth word in the whole dataset. To this end, the TF-IDF value for each $w_d \in \mathcal{D}$ can be obtained by multiplying the values of TF and IDF. It is remarkable that the larger the TF-IDF value is, the more important the keyword is. The second method is to compute a t-test type statistic. Recall that, for the dth keyword, $w_d \in \mathcal{D}$, we have two sets of class probabilities, $\widehat{p}_i$ and $\widehat{p}_i^{(d)}$. Both are K-dimensional vectors. Then, for each dimension $k \in K$, a standard paired two sample t-test can be constructed to test for the statistical significance. The resulting p values obtained from different ks (e.g., different categories) are then summarized, and the smallest one is selected as the final t-test type measure for the target keyword, denoted as $P_{i,d}$. In this case, the smaller the $P_{i,d}$ value is, the more important the keyword is.

In summary, we have three methods to evaluate the importance of each keyword in $\mathcal{D}$. These are the proposed method for evaluating the differences in class probability estimators (CPE), the method for evaluating the TF-IDF values (TF-IDF), and the method for evaluating the t-test type statistics (t-statistic). To make a fair comparison, the new dictionaries constructed by the three methods are of equal size (e.g., with same tuning parameter K). Then, following the procedure described in Sect. 3.2, we can obtain three different prediction accuracies based on the screened dictionary.

3.5 Performance Measures and Implementation

Following the existing literature [1,17,18], and our own concerns, we adopt six measures to gauge the empirical performances of the different compression methods: the parameter reduction ratio (Prr), dictionary reduction ratio (Drr), storage reduction ratio (Srr), FLOP reduction ratio (Frr), and reduction ratio for averaged text sequence (Trr). Meanwhile, the out-of-sample prediction accuracy (Acc) is also monitored.

Both text classification models (e.g., TextCNN and TextBiLSTM) are trained on the four large-scale datasets. This leads to a total of eight working models. All the working models are trained using the AdaDelta (Zeiler, 2012) with $\rho = 0.95, \epsilon = 10^{-5}$, and a batch size of 128. The weight decay is set to 5×10^{-4} with an ℓ_2-norm regularizer. To prevent overfitting, the dropout and early stopping strategies are used for different working models. Finally, a total of 200 epochs are conducted for each working model. For each working model, we choose the epoch with the maximum prediction accuracy on the validation set as the baseline model. All the experiments were run on a Tesla P100 GPU with 64 GB memory.

4 Results Analysis

4.1 Tuning Parameter Effects

In this subsection, we study the impact of the tuning parameter, K, which determines the number of keywords reserved. Three measures are used to gauge the finite sample performance: Acc, Prr, and Trr. For illustration purposes, we use the AG's News dataset as an example. For this experiment, three different K values are studied: $K = \{1000, 3000, 5000\}$. The detailed results are given in Fig. 1. The top panel of Fig. 1 displays the performance of the TextCNN model. The red line in the first barplot is the prediction accuracy for the benchmark model. We find the resulting prediction accuracy (Acc) of the reduced model increases as K becomes larger, while the parameter reduction ratio (Prr) and the reduction ratio for averaged text sequence (Trr) decrease. In the case of $K = 3000$, we can see the parameter reduction ratio (Prr) is more than 95%, but there is almost no accuracy loss. This suggests that the benchmark model can be substantially compressed with little sacrifice in predictive power. Additionally, the averaged text sequence is substantially reduced based on the dictionary screening. This indicates that there might be some redundant information in the original text that contributes less to the text classification. The bottom panel of Fig. 1 presents the results for the TextBiLSTM model, which are very similar to the findings of TextCNN.

4.2 Performance of Compression Results

On the one hand, the proposed dictionary screening method aims to compress a text classification model as much as possible. On the other hand, an over compressed model might suffer from a significant loss of prediction accuracy. Thus, it

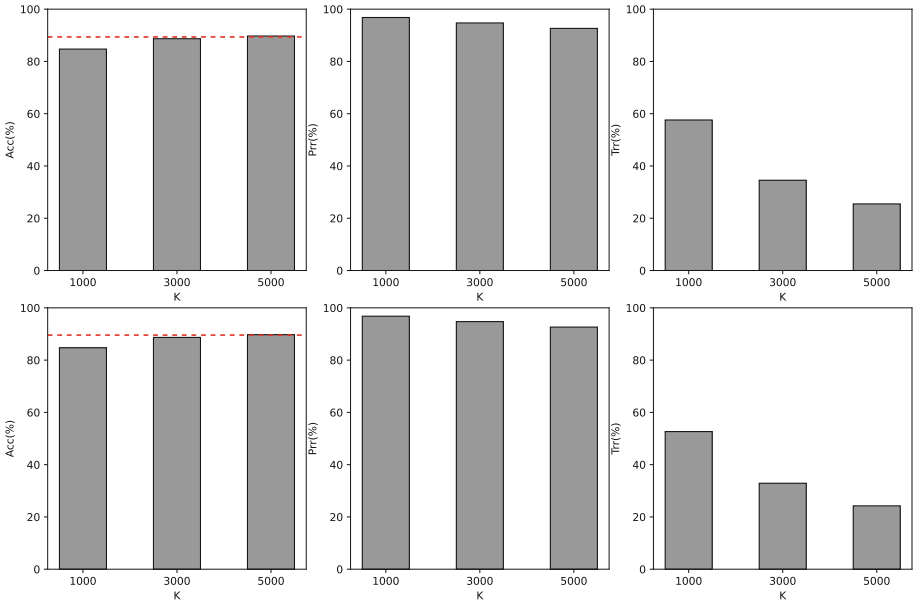

Fig. 1. Detailed experimental results for TextCNN and TextBiLSTM on the AG's News dataset. Three different K values are considered (K = 1000,3000,5000). Three performance measures are summarized: prediction accuracy (Acc), parameter reduction ratio (Prr), and reduction ratio for averaged text sequence (Trr). The red dashed line in the left panel represents the accuracy of the benchmark model. (Color figure online)

is of great importance to understand the trade-off between prediction accuracy and model compression. Obviously, they should be appropriately balanced. In this subsection, we report the fine-tuned compression results so that their best performance can be demonstrated. For the best empirical performance, we try to extend the search scope for more tuning parameters. Accordingly, every value in $\{1000, 2000, \cdots, 10000\}$(e.g., with an interval of 1000) is tested for K in this subsection. In our case, we expect that the parameter reduction ratio (Prr) will be no less than 50% and the accuracy loss will be no more than 2%. The best results in terms of the above criteria are summarized in Table 2. From Table 2, we can draw the following conclusions. First, for all cases, the benchmark models can be compressed substantially using the dictionary screening method with little sacrifice of accuracy. For example, the value of Prr is more than 95% in the case of TextCNN on Sougou News with only 0.31% sacrifice of accuracy. Second, we report that for half of the cases, the prediction accuracy of the reduced model is higher than that of the benchmark model (e.g., ΔAcc is smaller than zero). For instance, for the TextBiLSTM model on DBPedia, the prediction accuracy of the baseline model is as high as 97.77%, while that of the reduced model is further improved to 97.99%. Finally, we find the reduction in storage and FLOPs are also quite substantial. For instance, the Srr is 99.56% and the Prr is

99.70% for the TextBiLSTM on ARP. To summarize, we find that the proposed dictionary screening method works quite well on all models and datasets under consideration.

Table 2. Fine-tuned dictionary screening results for all model and dataset combinations with the best performance. ARP stands for the Amazon Review Polarity dataset. Acc-1 is the prediction accuracy of the benchmark model. Acc-2 is the prediction accuracy of the reduced model. ΔAcc is the difference between the benchmark Acc and reduced Acc. All computed values are in % units.

	Model	Acc-1	Acc-2	ΔAcc	Prr	Drr	Trr	Srr	Frr
TextCNN	AG's News	89.37	89.43	-0.06	95.24	96.81	34.54	85.61	89.49
	Sougou News	95.56	95.25	0.31	97.28	98.19	48.35	97.26	93.83
	DBPedia	98.41	97.88	0.53	98.97	99.27	27.17	99.40	98.97
	ARP	92.70	91.45	1.25	99.74	99.66	13.75	98.77	99.33
TextBiLSTM	AG's News	89.58	89.72	-0.14	92.65	94.68	24.25	85.19	90.72
	Sougou News	95.77	95.34	0.43	96.98	98.19	47.53	97.96	95.81
	DBPedia	97.77	97.99	-0.22	99.33	99.56	35.14	98.95	98.44
	ARP	92.16	92.26	-0.10	99.78	99.87	24.75	99.56	99.70

4.3 Competing Methods

Table 3. Results of the three competing methods (e.g., CPE, TF-IDF, and t-statistics). ARP stands for the Amazon Review Polarity dataset. Reduced Acc is the prediction accuracy of the reduced model. For each model and dataset combination, the reduced models were trained with dictionaries of equal size. Trr is the reduction ratio for averaged text sequence. The values outside of brackets are the results of the propose CPE method. The first value in brackets is the result obtained by TF-IDF, and the second value is the result by t-statistic. All computed values are in % units.

	Model	Reduced Acc	Trr
TextCNN	AG's News	89.66 (89.33, 87.91)	34.54 (29.19, 46.07)
	Sogou News	94.71 (94.66, 94.68)	57.54 (44.39, 55.56)
	DBPedia	97.89 (97.89, 17.23)	27.17 (25.36, 93.12)
	ARP	90.73 (90.49, 89.77)	25.50 (24.25, 32.50)
TextBiLSTM	AG's News	90.49 (90.09, 89.89)	18.07 (13.54, 23.84)
	Sogou News	95.47 (95.14, 95.31)	47.53 (37.53, 52.54)
	DBPedia	97.99 (98.00, 17.39)	35.14 (30.80, 99.82)
	ARP	92.32 (92.24, 91.55)	24.75 (24.25, 30.00)

Because the key step of the proposed dictionary screening method is to evaluate the importance of each keyword in the dictionary. We compare the performance of the proposed evaluating method, CPE, with two other competing

methods. These are the TF-IDF and t-statistics methods, which are described in Subsect. 4.4. For a fair comparison, the new dictionaries constructed with the three methods are of equal size (e.g., the dictionary reduction ratio, Drr, is the same) for each model and dataset combination. As a result, only the reduced prediction accuracy (Acc) and reduction ratio for averaged text sequence (Trr) are presented in Table 3. From Table 3, we can obtain the following conclusions. First, we can see that the t-statistics method is not stable in evaluating the importance of keywords because its results for the DBPedia dataset were not comparable with the other methods. In the case of DBPedia, the t-statistic method ceases to be effective because its Trr value is nearly 100%. This indicates that it cannot filter the important keywords from the dictionary, leading to a very low reduced Acc value. Second, in all cases, the proposed CPE method achieved a slightly higher reduced accuracy value compared with the TF-IDF method. Moreover, for most cases, by using the proposed CPE method, we can achieve a substantially reduced ratio for averaged text sequence (Trr). This indicates that the proposed CPE method can achieve a better performance in terms of predicted accuracy when keeping a relatively short text sequence.

5 Conclusions

In this paper, we propose a dictionary screening method for embedding compression in text classification tasks. The goal of this method is to evaluate the importance of each keyword in the dictionary. To this end, we develop a method called CPE to evaluate the differences in class probability estimators. With the CPE method, each keyword in the original dictionary can be screened, and only the most important keywords can be reserved. The proposed method leads to a significant reduction in terms of parameters, average text sequence, and dictionary size. Meanwhile, the prediction power remains competitive. Extensive numerical studies are presented to demonstrate the empirical performance of the proposed method.

To conclude this article, we present here a number of interesting topics for future study. First, the proposed dictionary screening method involves a tuning parameter (e.g., K), and its optimal value for balancing prediction accuracy and parameter reduction needs to be learned. This is an important topic for future study. Second, the proposed method is only used for a text classification task. However, there are other natural language tasks, such as machine translation, question answering, and so on. The scalability of the proposed method in these tasks is also a very worthy study. Lastly, the proposed method is only conducted on English and Chinese, other language types should be investigated to test its validity.

Acknowledgements. Zhou's research is supported in part by the National Natural Science Foundation of China (Nos. 72171226, 11971504), the Beijing Municipal Social Science Foundation (No. 19GLC052). Wang's research is partially supported by the National Natural Science Foundation of China (No. 12271012, 11831008) and the Open

Research Fund of the Key Laboratory of Advanced Theory and Application in Statistics and Data Science (KLATASDS-MOE-ECNU-KLATASDS2101).

References

1. Acharya, A., Goel, R., Metallinou, A., Dhillon, I.: Online embedding compression for text classification using low rank matrix factorization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 6196–6203 (2019)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
3. Cho, K., Merrienboer, B.V., Gulcehre, C., Schwenk, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. Computer Science (2014)
4. Cybendo, G.: Approximations by superpositions of a sigmoidal function. Math. Control Signals Systems **2**, 183–192 (1989)
5. Deng, L., Li, G., Han, S., Shi, L., Xie, Y.: Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc. IEEE **108**(4), 485–532 (2020)
6. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. J. Mach. Learn. Res. **15**, 315–323 (2011)
7. Hornik, K., Stinchcombe, M.B., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**, 359–366 (1989)
8. Hossain, E., Sharif, O., Hoque, M.M., Sarker, I.H.: SentiLSTM: a deep learning approach for sentiment analysis of restaurant reviews. In: Proceedings of 20th International Conference on Hybrid Intelligent Systems (2020)
9. Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jegou, H., Mikolov, T.: Fasttext.zip: compressing text classification models. arXiv preprint arXiv:1612.03651 (2016)
10. Kim, Y.: Convolutional neural networks for sentence classification. Eprint Arxiv (2014)
11. Li, F., Zhang, M., Fu, G., Qian, T., Ji, D.: A Bi-LSTM-RNN model for relation classification using low-cost sequence features (2016)
12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. Computer Science (2013)
13. Raunak, V.: Effective dimensionality reduction for word embeddings. arXiv preprint arXiv:1708.03629 (2017)
14. Sachan, D.S., Zaheer, M., Salakhutdinov, R.: Revisiting LSTM networks for semi-supervised text classification via mixed objective function. Proceedings of the AAAI Conference on Artificial Intelligence (2019)
15. Shu, R., Nakayama, H.: Compressing word embeddings via deep compositional code learning. arXiv preprint arXiv:1711.01068 (2017)
16. Sparck-Jones, K.: A statistical interpretation of term specificity and its application in retrieval. J. Document. **28**(1), 11–21 (1972)
17. Xiao, Y., Cho, K.: Efficient character-level document classification by combining convolution and recurrent layers (2016)
18. Zhang, X., Zhao, J., Lecun, Y.: Character-level convolutional networks for text classification. MIT Press (2015)