The Journal of Engineering

The 2nd 2018 Asian Conference on Artificial Intelligence Technology (ACAIT 2018)

IET Journals
The Institution of Engineering and Technology

Text clustering algorithm based on deep representation learning

eISSN 2051-3305
Received on 18th July 2018
Accepted on 26th July 2018
E-First on 22nd October 2018
doi: 10.1049/joe.2018.8282
www.ietdl.org

Binyu Wang[1], Wenfen Liu[2] ✉, Zijie Lin[1], Xuexian Hu[1], Jianghong Wei[1], Chun Liu[1]

[1]*State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, People's Republic of China*

[2]*Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, 541000 Guilin, People's Republic of China*

✉ *E-mail: liuwenfen@guet.edu.cn*

Abstract: Text clustering is an important method for effectively organising, summarising, and navigating text information. However, in the absence of labels, the text data to be clustered cannot be used to train the text representation model based on deep learning. To address the problem, an algorithm of text clustering based on deep representation learning is proposed using the transfer learning domain adaptation and the parameters update during cluster iteration. First, source domain data is used to perform the pre-training of the deep learning classification model. This procedure acts as an initialisation of the model parameters. Then, the domain discriminator is added to the model, to domain-divide the input sample. If the discriminator cannot distinguish which domain the data belongs to, the common feature space of two domains is obtained, so the domain adaptation problem is solved. Finally, the text feature vectors obtained by the model are clustered with MCSKM++ algorithm. The algorithm not only resolves the model pre-training problem in unsupervised clustering, but also has a good clustering effect on the transfer problem caused by different numbers of domain labels. Experiments suggest that the clustering accuracy of the algorithm is superior to other similar algorithms.

1 Introduction

With the development of the big data era, text is produced substantially and has rich value. How to use text rationally is an opportunity as well a challenge that humans are faced with. Text clustering, as an important method of text data mining, classifies a large number of disordered and disorganised articles by similarity judgment, so that articles with similar themes and contents are classified into the same category and dissimilar articles into different categories [1]. As an unsupervised machine learning method, text clustering, which does not require a training and manual labelling of texts in advance, has certain flexibility and a high level of automation. It has become an important technique for effectively organising, abstracting, and navigating text information, and has attracted the attention of more and more researchers [2].

The representation of unstructured text data is a core problem of text clustering. The text representation based on statistical measurement [3] has disadvantages, such as high-dimensional sparseness, unexpressed semantic information, and so on. So deep learning has become an important method of text representation in recent years due to its powerful automatic feature extraction capabilities. However, text clustering is an unsupervised technique. Pre-training a model without label data is an essential issue, affecting the text clustering result.

In order to solve the model training problem caused by unlabelled data, many scholars have employed the unsupervised learning methods, such as self-encoder, transfer learning, and so on to extract feature vectors of texts. Xu *et al.* [4] proposed an automatic feature extraction method using convolution neural network on the North American Chapter of the Association for Computational Linguistics, in which text is first transformed into a low-dimensional vector by locality-preserving constraint, and then the vector can be transformed into compact binary code for model training. In 2017, Xu *et al.* [5] generalised the proposed model on the basis of his research outcome, using various unsupervised dimension reduction methods to reduce the dimensionality of high-dimensional sparse vectors to get the labels, and then training the modal with those labels. Meanwhile, in 2017, researchers from the IBM Watson Institute [6] proposed a method of semi-supervised text clustering by deep learning, which combines the text deep representation learning with the *K*-means. A small amount of labelled data is employed to supervise the clustering and text training. Then, the final text representation model and clustering results are obtained. Moreover, Andrew Ng gave another solution for the unsupervised learning on neural information processing systems in 2016. He believes that transfer learning is the next hotspot for deep learning and the key to solving masses of unlabelled data in practice.

The methods mentioned above have learnt the feature vector of the text to a certain extent. However, there are still three problems as follows:

1. In the unsupervised dimensionality reduction method, the initial parameters of the model are randomised. As a result, the model training is prone to overfitting, thus affecting the effects of model training.
2. In the transfer learning, the source domain model cannot fully meet the model training of the target domain model, leading to poor clustering results.
3. Without directivity constraints in the unsupervised training, the text feature vectors obtained by the model training inadequately describe the clustering-related features of the dataset, therefore, fails to mine the dominant influence factors that characterise the dataset.

To solve the above problems, we propose a deep-learning text clustering algorithm, which combines the best aspects of transfer learning and unsupervised constraints. It utilises the transfer learning domain adaptation and the continuous optimisation of model parameters during cluster iterations. For Problem 1, to prevent the overfitting by random parameters, we use labeled dataset related to the target dataset, to get the initialised parameters and hyperparameters of the modal. For Problem 2, the domain classifier is imported so that the model can work out the common feature vectors of the target dataset and the source dataset. For Problem 3, we combine the deep representation training with MCSKM++ and optimise the parameters during the clustering so that the text feature vector obtained by the training corresponds to the characteristics of the text dataset clustering. The algorithm solves the model pre-training problem of unsupervised clustering,

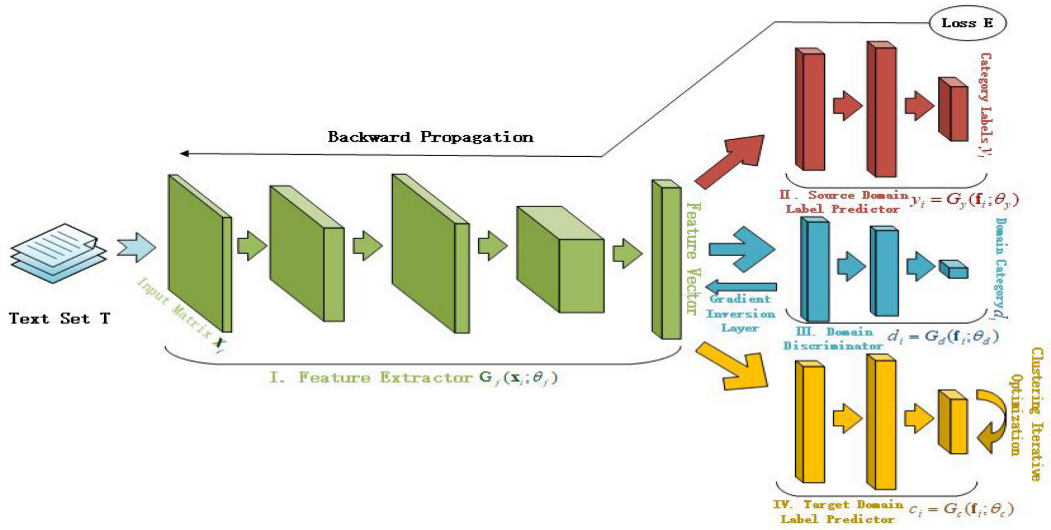

Fig. 1 *Overall model framework of the algorithm*

and has a good clustering effect on the different domain labels transferring problem. It has been shown that our algorithm is obviously better than other algorithms in nearly all aspects.

2 Methodology

The purpose of our algorithm: For a given set of textual datasets $T = \{t_1, t_2, \ldots t_N\}$, we use a deep representation learning model $G_f(T;\theta_f)$ to obtain a set $F = \{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots \boldsymbol{f}_N\}$ of dense feature vectors and classify the vectors into different categories using clustering algorithm. The algorithm focuses on how to train and optimise the parameters θ_f of the deep representation learning model $G_f(T;\theta_f)$ in the absence of labels in the text dataset T, so that the resulting feature vector set can express the data features of the text set to the maximum extent, and then obtain accurate clustering results. To solve the above problems, this paper proposes the following algorithm and the algorithm framework of the algorithm is shown in Fig. 1.

The model is made up of four parts. The green part of the figure is I feature extractor. The feature extractor consists of deep neural networks and can use recurrent neural network (RNN) [7], convolutional neural network (CNN) [8], long-term short-term memory network (LSTM) [9], and other deep learning models to extract features from the data; the red part, II the source domain classifier, which classifies feature vectors of the source domain text, is generally composed of a fully connected layer and a logistic regression classifier; the blue part is a domain discriminator that discriminates the domain of the input data, and the domain discriminator completely consists of the connection layer and the cross-entropy classifier; the yellow part is the target domain classifier, which classifies the feature vectors of the target domain text. The following describes the detailed process and principle of this algorithm.

The overall framework of the model is as follows: first, the most basic deep classification model training is performed using labelled datasets in the source domain, and the obtained model parameters are initialised as the next parameter, making the model suitable for text classification or clustering tasks. In the second step, the domain discriminator is added to the model, and the domain discriminator is used to discriminate the input samples. When the domain discriminator is powerful but unable to distinguish the data domain, a common learning model can be learned to solve the domain adaptation problem. In the third step, the domain-adapted parameters are used as the model initialisation, and the output feature vector $\boldsymbol{F}$ is clustered. In the clustering process, the model parameters are continuously updated according to the distribution characteristics of the target dataset, making it more suitable for the target domain. Finally, the text clustering results are obtained at the same time when the text representation suitable for the task is obtained. The following describes the principle and mechanism of the model in detail.

2.1 Deep learning model training with labelled data in the source domain

For the problem that it is difficult to pre-train the deep model parameters without labels for text set, we introduce text datasets $T' = \{(t'_1, y'_1), (t'_2, y'_2), \ldots (t'_N, y'_N)\}$ that are publicly labelled datasets as similar as possible to the distribution of dataset T. This dataset T' is called the source domain. The processed unlabelled dataset T is called the target domain. For the source domain, the model training of the deep learning classification task is performed using the label information. This step consists of two parts: the I text feature extractor and the II label classifier. Through training, the parameters of the model are optimised, and a deep learning text classification task model is obtained.

For text set T', data preprocessing is first performed. Word2Vec [10] and Cascading convert text set T' into a matrix $X' = \{\boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{N_2}\}$, which is the input space for the classification model. Vector set X' entering the feature extractor, the extractor can be composed of text representation models such as RNN, CNN, and LSTM. In this paper, we use LSTM as an example. For text $\boldsymbol{x}'_i$, the working principle of the moment is

$$i_t = \sigma(W_i \boldsymbol{w}'_{it} + W_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma\left(W_f \boldsymbol{w}'_{it} + W_f h_{t-1} + b_f\right) \tag{2}$$

$$o_t = \sigma(W_o \boldsymbol{w}'_{it} + W_o h_{t-1} + b_o) \tag{3}$$

$$\tilde{C}_t = \tanh(W_c \boldsymbol{w}'_{it} + W_c h_{t-1} + b_c) \tag{4}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \tilde{C}_t \tag{5}$$

$$h_t = o_t \otimes \tanh(C_t) \tag{6}$$

At the $|\boldsymbol{x}'_i|$ moment, we get the text feature vector $\boldsymbol{f}'_i$ output for the first time. We use the function $G_f(\boldsymbol{x}'_i;\theta_f)$ to represent the whole process, where θ_f is the set of parameters in the LSTM. The process can be simplified to $\boldsymbol{f}'_i = G_f(\boldsymbol{x}'_i;\theta_f)$. Afterwards, the feature vector $\boldsymbol{f}_i$ enters the classifier. This part is composed of a full-connected layer and a logistic regression classifier. The linear and non-linear mapping of the feature vector is performed by many neurons in each layer to obtain the predictive label information $\tilde{y}'_i$ of the text. This process can be expressed as $\tilde{y}'_i = G_y(\boldsymbol{f}'_i;\theta_y)$, θ_y is a collection of parameters in the fully connected layer. The training loss of a single sample from predictive labels $\tilde{y}'_i$ and real labels y'_i is

$$E_y(x_i') = L_y(\tilde{y}_i', y_i') = L_y(G_y(G_f(x_i'; \theta_f); \theta_y), y_i') \quad (7)$$

In batch update mode, the total loss function is

$$E_y = \frac{1}{N_2}\sum_{i=1}^{N_2} E_y(x'_i) = \frac{1}{N_2}\sum_{i=1}^{N_2} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y'_i) \quad (8)$$

The backpropagation is performed using a stochastic gradient descent (SGD) whereby the model parameters (weights and offsets) can be updated as follows:

$$\theta_f \leftarrow \theta_f - \mu\frac{\partial E_y}{\partial \theta_f} \quad (9)$$

$$\theta_y \leftarrow \theta_y - \mu\frac{\partial E_y}{\partial \theta_y} \quad (10)$$

Among them, for the learning rate μ, its range is (0,1).

Therefore, after the first round of parameter adjustment of the model, the above steps are repeated again until the predicted label and the real label are completely the same, and the model stops training, and a classification model based on the deep learning text representation is obtained. The role of the training process is to provide initialisation of the model parameters for the second part of the domain adaptation model to prevent the phenomenon of gradient disappearance and overfitting during the training process caused by random initialisation of parameters.

2.2 Join domain discriminator to solve domain adaptation

As a result, we have obtained a classification model based on deep learning text representation. However, due to the distributional deviation between the source domain data and the target domain data, the model cannot be directly applied to the target data. The domain adaptation problem is based on the problem that the distribution of the target domain is solved under the condition that the distribution of the source domain is known under the condition that the two distributions are similar but not identical. Solving this problem is often based on the shared-classifier assumption: if a common feature representation space, i.e. domain-invariant feature, can be obtained on the source and target domains, the feature learning model of the source domain can be applied to feature learning of the target domain. Referring to the idea of the adaptation of the domain adaptive neural network to the network [11] to the image domain, this step consists of the I, II, and III parts of Fig. 1, combined with the anti-network and deep-learning text representation, adding a domain discriminator to the model, when the domain discriminator cannot distinguish the source of the data domain, the learned eigenvector has domain invariant characteristics and solves the domain adaptation problem.

For text set $T = \{t_1, t_2, \ldots t_{N_1}\}$, data pre-processing is performed in the same manner as in Section 2.1 to obtain vector sets $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_1}\}$. In this section, the input of the model is all the data in the source and target domains, i.e. $X \cup X' = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_1}, \boldsymbol{x}'_1, \boldsymbol{x}'_2, \ldots, \boldsymbol{x}'_{N_2}\}$. The work in the previous section led us to a deep learning classification model for the source domain. Although the domain distribution is different, it cannot be used directly. However, the model can be used as the initialisation of the section to prevent the gradient from disappearing and being simulated in the training process caused by random initialisation of parameters. This initialises the I and II parts of the model, and part III uses random initialisation. For the text matrix $\boldsymbol{x}_i$, the eigenvector $\boldsymbol{f}_i$ is obtained by the mapping transformation of the LSTM model. At this time, however, the eigenvector enters the domain discriminator at the same time as the label predictor enters the label predictor, and the domain of the data is discriminated. The label predictor divides the category label of the data, which promotes that the feature vector can fully express the characteristics of the text. At the same time, the presence of the domain discriminator makes it possible to discriminate the domain label of the feature vector, and the discriminator cannot distinguish the label of $\boldsymbol{f}_i$. When the domain label of the domain belongs to and the class label predictor can still be divided normally, we can obtain the common feature representation space of the source domain and the target domain, that is, learn the domain invariant features. The design of the domain discriminator is performed below.

First, the domain discriminator consists of a fully connected layer and a cross-entropy classifier. The activation function used by the fully connected layer is the rectified linear unit. The function of this discriminator is to determine whether the vector is from the source domain or the target domain for the input feature vector. The process is expressed as $d_i = G_d(\boldsymbol{f}_i; \theta_d)$, where the set of model parameters θ_d in the discriminator is the domain category d_i, when the data belongs to the target domain, $d_i = 1$; otherwise, $d_i = 0$. The discriminator proposes two opposite requirements during training. On the one hand, the model needs a strong domain discriminating function, which can identify the domain to which the data belong to. In the forward propagation process, the discriminator training is the same as the normal classifier, requiring the final classification loss to be minimal. However, on the other hand, in order to learn out the domain invariant features, the discriminator cannot correctly distinguish the feature vectors obtained by the feature extractor, i.e. the domain discriminator is required to have the largest classification loss. The requirements for this discriminator against each other can be expressed as follows:

$$(\hat{\theta}_f, \hat{\theta}_y) = \arg\min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \quad (11)$$

$$\hat{\theta}_d = \arg\max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d) \quad (12)$$

Since the two equations are, respectively, required to be maximised and minimised, in order to make the above formula comply with the expression of the standard gradient propagation direction, a gradient reversal layer (GRL) is added at the I and III junctions at this time, and the gradient inversion is performed. The layer has only the meta-parameter λ and does not require backpropagation updates. It has two unequal manifestations in the forward and backward processes: in the forward propagation process, the GRL remains unchanged $R_\lambda(x) = x$; during the backward propagation, the GRL takes the gradient from the subsequent layer, multiplies $-\lambda$ and will be passed to the previous layer $R_\lambda(x) = -\lambda x$. With this gradient inversion layer, the model can be backpropagated using the standard SGD method.

The loss function of the label predictor E_y and the loss function of the domain discriminator E_d in the model are as follows:

$$E_y = \frac{1}{N_2}\sum_{i=1}^{N_2} E_y(x_i) = \frac{1}{N_2}\sum_{i=1}^{N_2} L_y(G_y(G_f(\boldsymbol{x}_i; \theta_f); \theta_y), y_i) \quad (13)$$

$$\begin{aligned} E_d &= \frac{1}{N_1}\sum_{i=1}^{N_1} E_d(x_i) + \frac{1}{N_2}\sum_{i=1}^{N_2} E_d(x_i) \\ &= \frac{1}{N_1}\sum_{i=1}^{N_1} L_d(G_d(G_f(\boldsymbol{x}_i; \theta_f); \theta_d), d_i) \\ &\quad + \frac{1}{N_2}\sum_{i=1}^{N_2} L_d(G_d(G_f(\boldsymbol{x}_i; \theta_f); \theta_d), d_i) \end{aligned} \quad (14)$$

where N_1, N_2 are the number of texts in the target domain and source domain, respectively. Through the gradient inversion layer, the overall loss function of the model is

$$E(\theta_f, \theta_y, \theta_d) = E_y - \lambda E_d$$
$$= \frac{1}{N_2}\sum_{i=1}^{N_2} L_y(G_y(G_f(\boldsymbol{x}_i;\theta_f);\theta_y), y_i)$$
$$-\lambda\left(\frac{1}{N_1}\sum_{i=1}^{N_1} L_d(G_d(G_f(\boldsymbol{x}_i;\theta_f);\theta_d), d_i)\right) \quad (15)$$
$$+\frac{1}{N_2}\sum_{i=1}^{N_2} L_d(G_d(G_f(\boldsymbol{x}_i;\theta_f);\theta_d), d_i))$$

This optimises this function in the standard gradient descent method to get the following parameter update method:

$$\theta_f \leftarrow \theta_f - \mu\left(\frac{\partial E_y(x_i)}{\partial \theta_f} - \lambda\frac{\partial E_d(x_i)}{\partial \theta_f}\right) \quad (16)$$

$$\theta_y \leftarrow \theta_y - \mu\frac{\partial E_y(x_i)}{\partial \theta_y} \quad (17)$$

$$\theta_d \leftarrow \theta_d - \mu\frac{\partial E_d(x_i)}{\partial \theta_d} \quad (18)$$

When the iteration is completed, through the above-mentioned update manner, while obtaining a text representation model suitable for the target domain, the model can learn a feature vector having domain-invariant characteristics.

2.3 Continuous optimisation of model parameters during clustering iteration

Through the previous two sections, we used the deep learning text representation and transfer learning domain to adapt to the common feature space of the target domain and the source domain, and a text deep representation model. At this point, the output eigenvectors can be clustered directly on the existing basis to obtain the final clustering result. However, the eigenvectors obtained by the model still carry the characteristic genes of the target domain, which affects the clustering effect in the target domain. The clustering effect is not ideal for the case of different types of domain transfer learning. In this section, we combine the presentation learning process with the *K*-means clustering process to cluster the previously obtained eigenvectors and update the model parameters in the clustering iterative optimisation process until the objective function reaches convergence. The final text deep represents the model and clustering results.

The model in this section is composed of two parts I and IV in Fig. 1. Part IV is a classifier for the target domain. The role of this classifier is to update the parameters of the model continuously during the cluster iteration process. The initialisation of the classifier can be initiated using random initialisation or using the model parameters of part II. We define the presentation function as $c_i = G_c(\boldsymbol{f}_i;\theta_c)$. For each eigenvector, define a set of binary variables $r_{ik} \in \{0, 1\}$ that represent whether the vector $\boldsymbol{f}_i$ is assigned to the j cluster, where $j \in \{1, 2, \ldots, K\}$. Therefore, if assigned to a clustering category, the variables are satisfied

$$r_{ij} = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases} \quad (19)$$

When using a fuzzy clustering algorithm, $r_{ik} \in [0, 1]$ $\boldsymbol{\mu}_j$ is defined as the centroid of the cluster. After that, get the objective function that the cluster should satisfy

$$Z = \sum_{i=1}^{N}\sum_{j=1}^{K} r_{ij} \| G_f(\boldsymbol{x}_i;\theta_f) - \boldsymbol{\mu}_j \| \quad (20)$$

There are a total of three sets of parameters in the objective function: the cluster assignment set for each text $\{r_{ik}\}$, the cluster centroid set $\{\boldsymbol{\mu}_j\}$, and the parameter set θ_f within the deep representation model. When the objective function reaches its minimum, the values of the sets are the clustering results.

Input: Text matrix, LSTM model, model parameter set, objective function;
Output: Clustering results;
1. Initiate the parameters of the model by using the domain against transfer learning
2. Use MCSKM++ algorithm to select the initial cluster centre point
3. Allocation cluster: Assign each point to the nearest cluster
4. Estimating centre: Recalculating cluster centres
5. Update parameters: All points with cluster information can update the parameters in the neural network
6. Repeat 3, 4, 5 until the objective function converges

Fig. 2 *Algorithm 1 Optimisation objective function*

The *K*-means algorithm uses the gradient descent method to optimise the objective function and obtain an iterative process [12]. In fact, this is the same as the deep-learning text representation model. Therefore, we can combine the *K*-means clustering with the deep learning model during the optimisation process, and continuously update the parameters of the text representation model during the clustering iteration optimisation process. To optimise the objective function, we borrow the idea of expectation maximisation. Each iteration process involves two steps: E-step and M-step. Inspired by the iterative optimisation of the *K*-means algorithm, we design Algorithm 1 (see Fig. 2) so that the parameters are successively minimised. The algorithm flow is shown below. As the adaptive training is more sensitive to the initial centre point of the cluster, the MCSKM++ [13] algorithm is used to cluster the text.

First, the algorithm uses the 2.2 model to initialise all the parameters in the neural network, and uses the MCSKM++ strategy to initialise the cluster centroid. Then, the algorithm iterates through the following three steps until the objective function converges.

(1) Allocating clusters: Keeping the set $\{\boldsymbol{\mu}_j\}$ and θ_f fixed, the target function Z is minimized by adjusting the value of the set $\{r_{ik}\}$. The essence of this step is to assign each data point to the class cluster in the case of the model's parameters and the centroid remain unchanged. Keeping the set is constant, and the objective function is minimized by adjusting the value of the set. The essence of this is that each data point is assigned to a cluster with the model parameters and the center of mass unchanged. Because $\{\boldsymbol{\mu}_j\}$ and θ_f are fixed, through the objective function Z, we know that when $G_f(\boldsymbol{x}_i;\theta_f) - \boldsymbol{\mu}_j$ value is minimum and $r_{ik} = 1$, Z reaches the minimum. It means that it is necessary to assign each text to the nearest clustering centroid, which is the same as the E step in the K-means algorithm.

(2) Estimating the centroid: Keep the set invariant, by adjusting the value of the set to minimise the objective function. This step requires estimating the cluster centroid based on the cluster allocation in the clustering step. To optimise the objective function, we have

$$\frac{\partial Z}{\partial \boldsymbol{\mu}_j} = 2\sum_{i=1}^{N} r_{ij}(G_f(\boldsymbol{x}_i;\theta_f) - \boldsymbol{\mu}_j) \quad (21)$$

$$\boldsymbol{\mu}_j = \sum_{i=1}^{N} r_{ij} G_f(\boldsymbol{x}_i;\theta_f) / \sum_{i=1}^{N} r_{ij} \quad (22)$$

That is, when the cluster centre is the centre of each distribution point, the objective function reaches the minimum value. This is equivalent to the M step in the K-means algorithm.

(3) Update parameters: By keeping the set fixed, the parameters are updated to minimise, this step is that each text has its own category label temporarily after a round of clustering iterations, and the text representation model is trained using the label. We use the loss function as a function and use the Adam algorithm to train the

Table 1 Cross-domain text clustering dataset 20NG-A constructed by 20 newsgroups

Dataset	Source domain D^s	Target domain D^t
Comp vs Rec	comp.graphics	comp.os.ms-windows.misc
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware
	rec.motorcycles	rec.autos
	rec.sport.baseball	rec.sport.hockey
Comp vs Sci	comp.os.ms-windows.misc	comp.graphics
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware
	sci.electronics	sci.crypt
	sci.space	sci.med
Comp vs Talk	comp.os.ms-windows.misc	comp.graphics
	comp.sys.ibm.pc.hardware	comp.sys.mac.hardware
	talk.politics.mideast	talk.politics.guns
	talk.politics.misc	talk.religion.misc
Rec vs Sci	rec.autos	rec.motorcycles
	rec.sport.baseball	rec.sport.hockey
	sci.crypt	sci.electronics
	sci.med	sci.space
Rec vs Talk	rec.autos	rec.motorcycles
	rec.sport.baseball	rec.sport.hockey
	talk.politics.mideast talk.politics.misc	talk.politics.guns
		talk.religion.misc
Sci vs Talk	sci.crypt	sci.electronics
	sci.med	sci.space
	talk.politics.misc	talk.politics.guns
	talk.religion.misc	talk.politics.mideast

neural network [14], that is, to update the model parameters of the text representation model by means of backpropagation.

The algorithm continuously optimises the model parameters during the clustering iteration process. The advantages are: (i) Although the domain-adapted deep learning model can learn the domain invariant features, the eigenvectors is still affected by the source data domain, which restricts the feature expression of the target domain. However, the addition of this part is an unsupervised self-adaptive training completely on the target domain. The literature [15] has proved that this adaptive algorithm has a significant improvement, making the entire model more adapted to the text representation and feature extraction of the target domain, making the clustering result more accurate. (ii) This clustering process can solve the problem of different numbers of source domains and target domain categories, and expand the use of transfer learning. As the unsupervised cannot optimise fine-tune for the transfer learning model, the original transfer learning requires that the source domain and the target domain have the same number of categories, and the algorithm proposed in this paper can be clustered according to the characteristics and categories of its own data. The number K results in the category division of the dataset. Therefore, some source domains with different numbers of target domain categories can be selected, which enlarges the scope of use of transfer learning and improves the clustering accuracy.

3 Results

3.1 Experimental dataset

In the experimental part of this paper, we choose to be applied to the 20Newsgroups dataset that has been applied to evaluate cross-domain text classification algorithms several times for experimental analysis. The 20Newsgroups dataset contains nearly 20,000 newsgroup articles and is distributed evenly across 20 newsgroups. Two different processing methods are used to verify the validity of this algorithm.

A. The same number of domain categories: Referring to the treatment of datasets by previous people [16], this paper still generates six cross-domain text classification datasets based on the data hierarchy on the 20Newsgroups dataset, focusing on the two-category problem based on top-level newsgroups, mainly using Sci, Talk, Comp, and Rec four top newsgroups experimented. Its logic is that for each cross-domain dataset, two top-level newsgroups are selected as two categories of newsgroups, and several sub-newsgroups are selected to form a new domain. Table 1 lists the six cross-domain text datasets built on 20Newsgroups.

B. The number of domain categories is different: To test the performance of our algorithm when the number of target domains and source domain categories is different, we design a 20 NG-B dataset that uses the same selection method as above, but the source domain D^s consists of Comp, Sci, and Talk. Each two groups of news sets, and the target domain consists of Comp, Sci, Talk of the remaining sub-news and some of the Rec sub news set, the dataset distribution is shown in Table 2.

3.2 Experimental environment and hyperparameter settings

This paper selected Tensorflow as the experimental basic framework, and other environment configurations include Intel core i7 processor, 16 GB memory, 64-bit Ubuntu 16.04 LTS operating system. Due to the large amount of computation in the model training process, an additional GTX 1070 GPU accelerator card is added to improve the training efficiency.

First, the validity of the dimensions of the output dimension vector of the text representation model is evaluated. We switch the dimension size between the two. We use adjusted mutual information to reflect the performance of the CNN model and the LSTM model. The experimental results are shown in Fig. 3. We find that 100 is the best output size for the CNN and LSTM models. Therefore, we set the output size to 100 in the following experiment.

For the LSTM model, the number of hidden layer units is set to 400. According to Greff *et al.*'s research experience on parameter setting, combined with experimental data volume, the batch size was set to 32, the learning rate was initialised to 0.001, and the decay rate was set to 0.9 for every 1000 training steps. To increase the model's generalisation ability, set the dropout rate [17] (dropout) to 0.7. Using the early-stop [18] strategy, when the F1 value of the verification set does not increase by 5 rounds, the

Table 2 Cross-domain text clustering dataset 20NG-B constructed by 20 newsgroups

Source domain D^s		Target domain D^t	
Comp	comp.graphics	Comp	comp.os.ms-windows.misc
	comp.sys.ibm.pc.hardware		comp.sys.mac.hardware
Sci	sci.electronics	Sci	sci.crypt
	sci.space		sci.med
Talk	talk.politics.mideast	Talk	talk.politics.guns
	talk.politics.misc		talk.religion.misc
		Rec	rec.motorcycles
			rec.sport.hockey

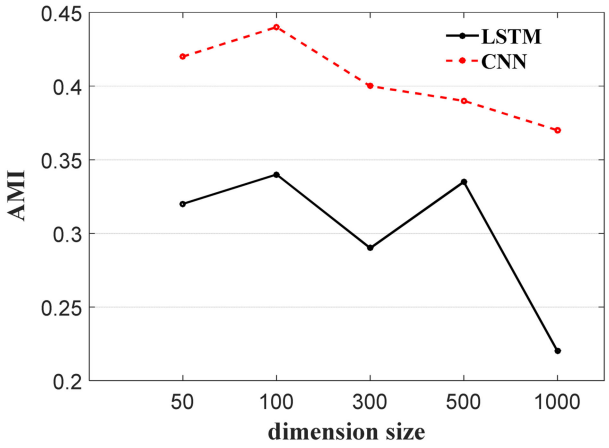

Fig. 3 *Effect of the dimensions of the output feature vector on the performance of the algorithm*

Table 3 Parameter settings of the LSTM model

Parameter	Setting value
word vector dimension	300
number of hidden layer units	400
max epoch	50
batch size	32
dropout rate (dropout)	0.7
early stop	5
learning rate	0.001
decline rate	0.9

system ends training to further prevent overfitting. After a lot of experimental comparisons, determine the model parameter settings, as shown in Table 3.

3.3 Evaluation indicators

We evaluated the clustering performance by comparing the clustering results of the text with the labels provided by the text corpora. Two metrics are selected, namely accuracy (ACC) and standardised mutual information metric (NMI) are used to measure clustering performance. Given a text x_i, let c_i and t_i are the obtained cluster labels and the labels provided by the corpus. Accuracy is defined as

$$\mathrm{ACC} = \frac{\sum_{i=1}^{n} \delta(y_i, \mathrm{map}(c_i))}{N} \tag{23}$$

where N is the total number of texts, $\delta(x, y)$ is an indicator function. If $x = y$, $\delta(x, y) = 1$; otherwise, $\delta(x, y) = 0$. $\mathrm{map}(c_i)$ is a permutation mapping function that maps each cluster label c_i to an equivalent label in text data by a Hungarian algorithm.

3.4 Experimental design and result analysis

In order to verify the validity of this model, the algorithm proposed in this paper is programmed. The feature extractor adopts two methods: CNN and LSTM. The model hyperparameters and partial configurations have been used. In addition, we further implement some representative and widely used algorithms and compare them with the performance of this paper, including the textual representation of Bag of Words (BoW) and vector space model (VSM) (TF-IDF)(term frequency-inverse document frequency) based on statistical measurement methods, and source only (SO) based on transfer learning. The adaptive CNN STCC [5], and the comparison algorithms are given in Table 4.

In order to control the variables to keep the experimental process fair, the text clustering algorithms used in this paper are all the MCSKM++ algorithms proposed in the third chapter. The experimental results obtained by the text clustering algorithm on the 20NG-A and 20NG-B datasets are shown in Tables 5 and 6, respectively.

The experimental results on the 20NG-A dataset are shown in the above figure. The bold part in Table 6 and the yellow and orange columns in Fig. 4 are the algorithms proposed in this paper. Obviously, it can be seen that the proposed algorithm is based on CNN network or LSTM network is superior to several other text clustering algorithms, and the LSTM-based text representation model is slightly superior to CNN in this model. Compared with the performance of the text clustering algorithm based on BoW and VSM, the performance of the proposed algorithm is improved by more than 20%. This shows that the text representation using the deep learning model is obviously better than the traditional text representation based on the statistical measurement learning

Table 4 Introduction to algorithms for experimental implementation

Abbreviation	Algorithm	Algorithm introduction
BoW	BoW	Bag of Words is the simplest and straightforward method of text representation. He treats documents as a collection of words, ignoring the elements of word order, grammar, and syntax. Each word appears independently and has no other words. rely
VSM	VSM (TF-TDF)	text clustering Algorithm for text representation using vector space model
SO	source only	based on the migration learning, the model trained with the source domain tagged data is directly applied to the target domain, or for the target domain with tagged data, the fine-tune operation is used to re-optimise the model
STCC [5]	self-taught CNNs for short text clustering	the short text clustering algorithm based on adaptive CNN proposed by Xu Jiaming in 2017 is an unsupervised representation of text clustering algorithm based on depth representation learning
PA (CNN)	proposed approach (CNN)	this paper proposes a text clustering algorithm based on depth representation learning, in which the feature extractor model uses CNN model
PA (LSTM)	proposed approach (LSTM)	this paper proposes a text clustering algorithm based on depth representation learning, in which the feature extractor model adopts a LSTM model

Table 5 Clustering accuracy on 20NG-A dataset

Dataset	BoW	VSM	SO	STCC	PA (CNN)	PA (LSTM)
Comp vs Rec	0.690	0.782	0.869	0.862	0.923	0.942
Comp vs Sci	0.511	0.619	0.808	0.663	0.845	0.831
Comp vs Talk	0.717	0.785	0.873	0.851	0.911	0.931
Rec vs Sci	0.574	0.593	0.746	0.674	0.782	0.793
Rec vs Talk	0.632	0.745	0.827	0.802	0.865	0.887
Sci vs Talk	0.625	0.696	0.754	0.766	0.817	0.842
average value	0.632	0.703	0.812	0.770	0.857	0.871

Table 6 Accuracy of clustering algorithms on 20NG-B dataset

Dataset	BoW	VSM	SO	STCC	PA (CNN)	PA (LSTM)
20NG-B	0.507	0.592	0.573	0.735	**0.772**	**0.784**

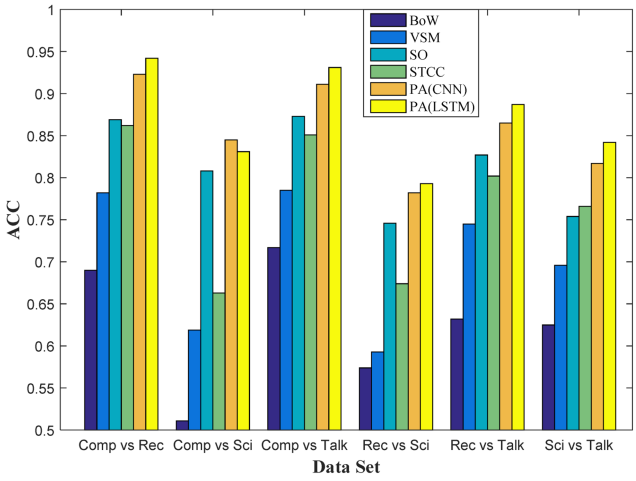

Fig. 4 *Clustering accuracy of six datasets in 20NG-A*

method, and embodies the feature extraction ability of deep learning. By comparing this algorithm with the SO algorithm based on transfer learning, it can be seen that the second-step domain adaptation and the third-step clustering iterative correction model parameters of the model improve the performance of the algorithm, making the text representation model more suitable for the target dataset and verifying the rationality of the model. For existing text clustering algorithms based on adaptive CNN, the proposed algorithm still has performance advantages.

Fig. 5 is the average of the clustering performance of the algorithm on each dataset. Since these datasets are all derived from the sum of 20NG, the mean is significant, and the superiority of this algorithm can be clearly seen from the histogram. For the 20NG-B dataset, results are shown in Table 6.

Table 3 and Fig. 6 show the accuracy of the various clustering algorithms for this algorithm when the number of domain categories is different. The number of source and target domain categories in the 20NG-B dataset is different, which results in poor learning of the SO algorithm based on transfer learning because the classifier formed by the source text domain depth model can still only distinguish three categories, articles of the Rec category cannot be distinguished separately, and the algorithm of this paper first performs domain adaptation, and then the cluster correction model parameters have a certain recognition effect on the fourth type of Rec, although compared with Table 5, the performance has a certain degree of decline, but for others there are major advantages of the algorithm. This shows that the algorithm proposed in this paper can identify domain categories that do not exist in the source domain. From another perspective, the algorithm can still be applied to the source domain datasets with different domain class numbers, which expands the scope of the migrated source domains.

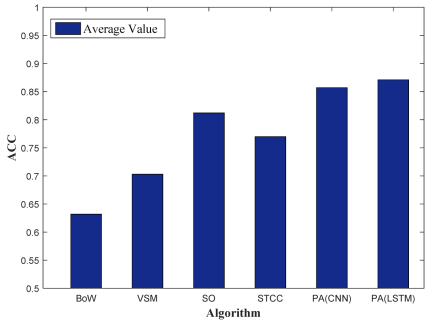

Fig. 5 *Average accuracy of clustering algorithms for 20NG-A algorithms*

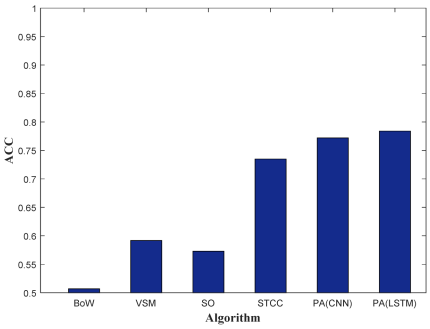

Fig. 6 *Clustering accuracy of algorithms for 20NG-B datasets*

4 Conclusion

This paper focuses on the deep-learning text representation and proposes a text clustering algorithm that uses transitional learning domain adaptation and clustering iterative adjustment parameters for the problem that unsupervised learning clustering cannot pre-train the deep representation model. This algorithm solves the problem of unsupervised and unpredictable model pre-training. At the same time, it also has a good clustering effect on the transfer problem with different number of domain labels. In the next step, we will continue to study the deep-learning text representation for unsupervised problems, and try to update and optimise the hyper-parameters of the target-domain deep model in the process of transfer training on the basis of adaptation of the transfer learning domain, so that the model is closely related to the characteristics of the target domain data, while expanding the scope of the source domain that the model can migrate.

5 Acknowledgments

This work was supported by the National Natural Science Foundation of China (61862011), the Natural Science Foundation of Guangxi (2018GXNSFAA138116), and Guangxi Key Laboratory of Cryptography and Information Security (GCIS201704).

6 References

[1] Aggarwal, C.C., Zhai, C.X.: 'A survey of text clustering algorithms', in Aggarwal, C.C., Zhai, C. (Eds.): '*Mining Text Data*' (Springer, US, 2012), pp. 77–128

[2] Yin, J., Wang, J.: 'A text clustering algorithm using an online clustering scheme for initialization'. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, San Francisco, USA, August 2016, pp. 1995–2004

[3] Dhillon, I.S., Modha, D.S.: 'Concept decompositions for large sparse text data using clustering', *Mach. Learn.*, 2001, **42**, (1), pp. 143–175

[4] Xu, J., Wang, P., Tian, G., *et al.*: 'Short text clustering via convolutional neural networks'. The Workshop on Vector Space Modeling for Natural Language Processing, Denver, USA, June 2015, pp. 62–69

[5] Xu, J., Xu, B., Wang, P., *et al.*: 'Self-taught convolutional neural networks for short text clustering', *Neural Netw.*, 2017, **88**, pp. 22–31

[6] Wang, Z., Mi, H., Ittycheriah, A.: 'Semi-supervised clustering for short text via deep representation learning'. Signll Conf. on Computational Natural Language Learning, Berlin, Germany, August 2016, pp. 31–39

[7] Legrand, J., Collobert, R.: 'Joint RNN-based greedy parsing and word composition', *Comput. Sci.*, 2015, pp. 51–61

[8] Kim, Y.: 'Convolutional neural networks for sentence classification'. Eprint Arxiv, 2014

[9] Hochreiter, S., Schmidhuber, J.: 'Long short-term memory', *Neural Comput.*, 1997, **9**, (8), pp. 1735–1780

[10] Mikolov, T., Chen, K., Corrado, G., *et al.*: 'Efficient estimation of word representations in vector space', *Comput. Sci.*, 2013

[11] Ganin, Y., Ustinova, E., Ajakan, H., *et al.*: 'Domain-adversarial training of neural networks', *J. Mach. Learn. Res.*, 2016, **17**, (1), pp. 2001–2035

[12] Bishop, C.M.: 'Pattern recognition and machine learning', *IEEE Trans. Inf. Theory*, 2012, **9**, (4), pp. 257–261

[13] Binyu, W., Wenfen, L., Xuexian, H., *et al.*: 'Research on text clustering for selecting initial cluster center based on cosine distance', *Comp. Eng. Appl.*, 2018, **54**, (10), pp. 11–18

[14] Kingma, D., Adam, B.J.: 'A method for stochastic optimization', *Comput. Sci.*, 2014

[15] Xue, G.R., Dai, W., Yang, Q., *et al.*: 'Topic-bridged PLSA for cross-domain text classification'. Int. ACM SIGIR Conf. on Research & Development in Information Retrieval (DBLP), Singapore, July 2008, pp. 627–634

[16] Greff, K., Srivastava, R.K., Koutník, J., *et al.*: 'LSTM: a search space odyssey', *IEEE Trans. Neural Netw. Learn. Syst.*, 2017, **28**, (10), pp. 2222–2232

[17] Srivastava, N., Hinton, G., Krizhevsky, A., *et al.*: 'Dropout: a simple way to prevent neural networks from overfitting', *J. Mach. Learn. Res.*, 2014, **15**, (1), pp. 1929–1958

[18] Raskutti, G., Wainwright, M.J., Yu, B.: 'Early stopping and non-parametric regression: an optimal data-dependent stopping rule', *J. Mach. Learn. Res.*, 2014, **15**, (1), pp. 335–366