Chinese Journal of Electronics
Vol.32, No.6, Nov. 2023

Rating Text Classification with Weighted Negative Supervision on Classifier Layer

ZHANG Jun[1,2], QIU Longlong[1,2], SHEN Fanfan[3], HE Yueshun[1,2], TAN Hai[3], and HE Yanxiang[4]

(1. *Radiological Geosciences Big Data Engineering Laboratory of Jiangxi Province, East China University of Technology, Nanchang 330013, China*)

(2. *School of Information Engineering, East China University of Technology, Nanchang 330013, China*)

(3. *School of Computer Science, Nanjing Audit University, Nanjing 211815, China*)

(4. *Computer School, Wuhan University, Wuhan 430072, China*)

Abstract — Bidirectional encoder representations from transformers (BERT) gives full play to the advantages of the attention mechanism, improves the performance of sentence representation, and provides a better choice for various natural language understanding (NLU) tasks. Many methods using BERT as the pre-trained model achieve state-of-the-art performance in almost various text classification scenarios. Among them, the multi-task learning framework combining the negative supervision and the pre-trained model solves the issue of the model performance degradation that occurs as the semantic similarity of texts conflicts with the classification standards. The current model does not consider the degree of difference between labels, which leads to insufficient difference information learned by the model, and affects classification performance, especially in the rating classification tasks. On the basis of the multi-task learning model, this paper fully considers the degree of difference between labels, which is expressed by using weights to solve the above problems. We supervise negative samples on the classifier layer instead of the encoder layer, so that the classifier layer can also learn the difference information between the labels. Experimental results show that our model can not only performs well in 2-class and multi-class rating text classification tasks, but also performs well in different languages.

Key words — Natural language understanding, Rating text classification, Weighted negative supervision, Degree of difference.

I. Introduction

Text Classification is a classical problem in natural language processing (NLP), which is widely applied in sentiment analysis, medical diagnosis, semantic annotation, public opinion control and other fields. Many researchers study how to improve text classification performance via taking advantage of traditional machine learning methods [1]–[5]. At present, deep learning-based methods [6]–[8] have replaced traditional statistics-based methods in the field of NLP. Compared with convolutional neural networks (CNN) [9]–[11], recurrent neural networks (RNN) [12]–[14] is better in case of dealing with sequential tasks including various NLP tasks. However, RNN is prone to incur the problem of gradient disappearance for it only considers or less considers the influence of words that is closer to the dealt word in a sentence, and even ignores the influence of words that is farther from the dealt word. The long short-term memory (LSTM) [15]–[19] neural networks proposed by Hochreiter *et al.* is a model based on RNN and gating unit, which solves the problem of gradient disappearance in BPTT (back propagation through time) [20] algorithm to a certain extent. However, the effect of LSTM is not ideal as the dealt sentence is too long, this is due to the existence of its structure design limitations inherited from RNN.

Sequence to sequence [21], [22] is also a framework specially used to deal with sequential problems including NLP tasks, which consists of an encoder network and a decoder network. The encoder network encodes each word of the dealt sentence, and extracts the se-

Manuscript Received Sept. 14, 2021; Accepted Mar. 14, 2023. This work was supported by the National Natural Science Foundation of China (62162002, 61662002, 61972293, 61902189), the Natural Science Foundation of Jiangxi Province (20212BAB202002), the Project of Jiangxi Engineering Laboratory on Radioactive Geoscience and Big Data Technology, China (JELRGBDT201905), and the Basic Science (Natural Science) Research Project of Colleges and Universities in Jiangsu Province (22KJA520004).

 DOI:10.23919/cje.2021.00.339

mantic information between these words into a vector. The decoder network generates and outputs the predicted value word by word according to the hidden state of the encoder and the previous output of the decoder network. The sequence to sequence model also has the problem of gradient disappearance for its encoder network and decoder network also adopt RNN model, so its performance could decline when the dealt sequence (or sentence) is too long. In 2014, bahdanau *et al.* proposed the attention mechanism [23] which solve gradient disappearance occurred in the sequence to sequence model effectively.

In 2017, Vaswanai *et al.* of Google put forward the transformer model based on their proposed self-attention mechanism and multi-headed-attention mechanism [24], which are based on attention mechanism. The transformer model can extract more and exact semantic information. In 2018, Devlin *et al.* proposed BERT (bidirectional encoder representations from transformers) model [25] for the first time, which is a very successful case using the transformer model. Later, a lot of related researches employed BERT model to pre-train various language models, and then trained the specific downstream tasks (such as text classification and machine translation) using the pre-trained BERT model [26]–[29].

At present, almost all text classification methods are based on BERT model in all kinds of classification datasets, which can obtain best effect. For example, Ohashi *et al.* proposed AM (the auxiliary task with the margin-based loss) model and AAN (the auxiliary task using all negative examples) model [30] both which are based on BERT model. Moreover, their proposed models are applied combined with negative supervision mechanism, which can perform better on MR, TREC and other datasets. The negative supervision mechanism proposed in this paper is to improve the accuracy of rating classification by learning distinct representations of texts with different labels. However, the method proposed in this paper does not distinguish the degree of difference between different labels, resulting in insufficient difference information extracted by learning, especially in the rating classification task. In addition, AM and AAN models only learn the difference of sentence representation (that is, the output difference of encoder layer), which can not reflect the learning of label difference in classifier layer, so that classification effect is not good.

To obtain better rating classification effect, this paper proposes rating classification with weighted negative supervision on classifier layer (WNSCL). WNSCL is also based on the idea of a negative supervision mechanism. Different from AM and AAN, WNSCL fully considers the degree of differences between different labels, learns the differences between the final output states of the texts with different labels. And WNSCL calculates the weight representing the degree of category differences and the loss function using the generated weight and the category differences to improve the model's classification performance. In classifier layer, WNSCL calculates the difference information of texts with different labels for the output of the classifier layer, so that the classifier layer can also learn the differences between labels. In addition, in case of training the model, WNSCL uses the dropout mechanism [31] to randomly inactivate some nodes (the inactivation probability is 0.4) to eliminate the over-fitting phenomenon to some extent. Compared with the existing technology, the main advantages of our method are given as follows.

- The proposed model can learn the difference degree between different labels, and better learn to extract the difference information of text categories in rating classification tasks.
- The classifier layer can also learn the above-mentioned category differences, which can improve the classification effect better.
- The accuracy of 2-class and multi-class text rating classification tasks has been significantly improved.
- The effect of rating classification in different language environments has been improved.

The rest of this paper is as follows. Section II introduces the work related to this paper. Section III in this paper expounds the idea of WNSCL in detail. Section IV describes the experimental process and analyses the results in detail. Section V concludes the main work of this paper and introduces our future work briefly.

II. Related Work

Many researches have used pre-trained BERT models as encoders for text classification tasks. This section introduces related work including BERT based fine-tuning, multi-task learning with deep neural networks and negative supervision.

1. BERT based fine-tuning

More and more NLP tasks including text classification tasks are implemented by using pre-trained BERT model. Sun *et al.* propose to select the output of which layer of BERT should be as the representation of the sentence based on the specific task, and use the target domain data to further pre-train the BERT in the text classification task [27]. Xu *et al.* introduce self-ensemble and self-distillation mechanism to improve the fine-tuning strategy while fine tuning BERT [28]. Self-ensemble mechanism combines several base models with parameter averaging rather than keeping several base models. And self-distillation mechanism uses knowledge

distillation to improve fine-tuning efficiency. Kuratov and Arkhipov propose to train the Russian monolingual model using multilingual initialization [29].

For text classification tasks, these original methods only add a fully connected layer directly to the BERT model as a classifier, which is not ideal for fine-grained emotional polarity classification for there is the conflict between classification standard and semantic similarity.

2. Multi-task learning

Multi-task learning (MTL) is an inductive transfer mechanism whose principle goal is to improve generalization ability. MTL can improve generalization by learning data related to task. Some studies combine MTL with algorithms such as k-nearest neighbor and kernel regression to obtain better performance [32]. Luong *et al.* applied MTL to the sequence to sequence model, which uses three MTL settings: one-to-many, many-to-one and many-to-many [33]. Chen *et al.* used deep wavelet decomposition networks and residual networks to provide an MTL framework for time series classification and retrieval tasks [34].

The combination of MTL and pre-training [35], [36] has been employed for text classification, which is referred to as MT-DNN model [37]. In MT-DNN model, the specific tasks share an encoder and fine tune the parameters of the encoder via training multiple specific tasks at the same time and updating weights, which makes the model have better generalization ability on the specific tasks. However, MT-DNN model requires more labeled datasets for multiple specific tasks need to be trained.

3. Negative supervision

In order to break through the performance limitations of the classification model as texts with similar semantics have different labels, Ohashi *et al.* propose the model AAN which achieves ideal performance in text classification tasks based on the idea of negative supervision [30]. Model AAN calculates the distance between representation vectors of texts with different labels by supervising negative samples, and adds the distance between vectors to the loss function to update the model parameters. In this way, AAN can learn the distinct information of different labeled texts. However, AAN does not consider the degree of difference between different labels, which makes the difference information learned by the model incomplete. In addition, the classifier layer cannot learn the difference information either, which will limit the classification performance.

III. The Proposed Model WNSCL

The idea of negative supervision in rating text classification tasks can make samples with different labels have distinct representations. However, the previous work [30] does not consider the degree of difference between these labels, which makes the difference information learned by the model not accurate enough and affects the performance of text classification task especially in rating text classification.

For example, for a 5-class rating classification task, there are five labels 0, 1, 2, 3, 4 corresponding to negative, somewhat negative, neutral, somewhat positive, and positive emotions. We suppose there are three samples of X_1, X_2, and X_3 with labels 0, 1, and 4 respectively from the training dataset, which are input into a model to obtain the corresponding representations V_1, V_2, and V_3. While conducting negative supervision, the previous approaches directly accumulate the distance between V_1 and V_2 and the distance between V_1 and V_3 into a loss function, ignoring the possible case the difference degree between label 0 and label 4 is greater than that between label 0 and label 1.

In this paper, we propose weighted negative supervision on classifier layer (WNSCL) model which fully considers the difference degree between labels. For representing this difference degree, our proposed method calculates the distance between V_1 and V_2 and the distance between V_1 and V_3, which is multiplied by a weight $W_{i,j}$. $W_{i,j}$ stands for the negative supervision weights of the sample pairs with label i and label j. In the example, the difference degree between V_1 and V_3 can be greater than that betweenV_1 andV_2 via setting the weight $W_{0,1}$ between V_1 and V_2 smaller than the weight $W_{0,4}$ between V_1 and V_3.

1. Architecture of WNSCL

WNSCL is a multi-task model, which consists of a main task and a negative supervision task (NST). The main task is to train a conventional classifier by adding a full connection layer as the classifier layer to BERT model.

The architecture of WNSCL model is shown in Fig.1. For the convenience of description, we define various parameters in WNSCL. The parameter X stands for the input of a sentence or a text, which is encoded into one n-dimensional vector l_1 ($l_1 \in \mathbb{R}^n$) using BERT model. The vector l_1 is calculated by a classifier layer to get the output vector l_2 ($l_2 \in \mathbb{R}^c$), which is the predicted value of WNSCL.

In Fig.1, the left part is NST, and the right part is the main task. For a text classification task, a sentence X is input into the bottom layer. X is converted into a matrix and input into the BERT model. After forward calculation in BERT, a text representation vector l_1 containing context is generated. Dimension of l_1 is the same as a row vector in the input matrix. And then the representation vector l_1 is input into the classifier layer

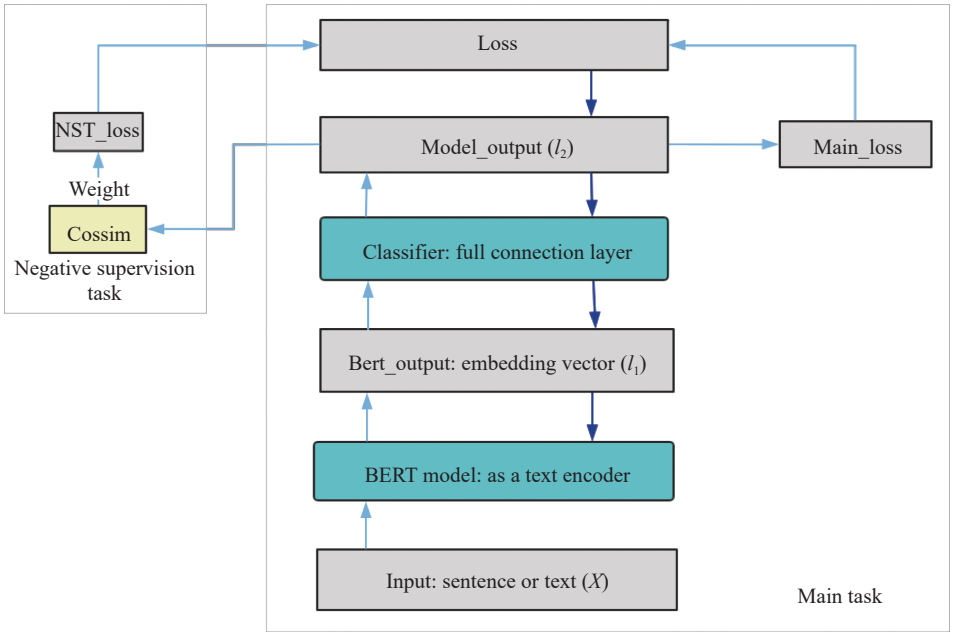


Fig. 1. Architecture of the proposed WNSCL model.

and calculated in WNSCL to generate the predicted value l_2. For the main task, WNSCL uses a specific loss function to calculate the loss value (L_m) between l_2 and the target. Meanwhile, l_2 is also input into NST so that NST can calculate its loss value (L_n) in a mini-batch via traversing all negative samples (the calculation of L_n in details is introduced in Section III.2). Finally, the loss of the main task and the loss of NST are added up and the sum is the total loss. The total loss is back propagated and the weights of WNSCL can be updated. In Fig.1, the up arrows stand for the forward calculation process, while the downward arrows stand for the back propagation process.

2. Negative supervision task

In order to enable the classifier layer to learn the difference information of different labels, different from AAN, WNSCL supervises the negative samples in the classifier layer instead of in the output layer of BERT. Thus, the output vector l_2 of the classifier layer is not only used to calculate the loss of the main task, but also input into NST to participate in the calculation of the loss function in the negative supervision task. For calculating the loss, NST loops through all the vectors l_2 of all samples in a mini-batch. During traversal, if two samples with different labels are found, NST calculates the cosine similarity between the vectors l_2 corresponding to the two samples. These cosine similarities are weighted summed with the weights calculated in the classifier layer to get the loss of NST (L_n), which is showed in equation (1).

$$L_a^b = \sum W_{k,i}\text{cossim}(l_2^k, l_2^i) \tag{1}$$

where L_n^k is the value of the loss function of NST corresponding to the kth sample X_k,$W_{k,j}$ is the negative supervision weight between two different samples with labels k and j, l_2^k and l_2^i stand for the vectors l_2 corresponding to the samples X_k and X_i in a mini-batch, function cossim(,) is used to calculate the cosine similarity between l_2^k and l_2^i.

For weight W, it is related to the distance of the labels. Moreover, W is a monotonically increasing function related to distance between different labels, for vectors l_2 of the samples with bigger distance between labels need be more dissimilar. For that, NST uses a linear function designed as equation (2).

$$W_{k,i} = \alpha \cdot \text{dis}(\text{label}_i, \text{label}_j) \tag{2}$$

where function dis(,) describes the distance between two different labels. It returns the absolute value of the difference between the two labels as these labels are described using scalars. Parameter α is a coefficient, which describes the importance of the loss function in NST. Its value is obtained via selecting the best value corresponding to the best performance from multiple sets of experiments.

Algorithm 1 describes how to calculate the loss function value of samples in a mini-batch of NST. Based on the analysis in Section III.1, it is from the classifier layer that NST obtains the predicted value l_2 of WNSCL. Algorithm 1 uses a two-level loop to find all negative samples in the mini-batch. Each inner loop tries to find a pair of negative samples via traversing each sample in a mini-batch. If a pair of negative samples is found, NST calculates the loss function value

of the pair of negative samples according to equations (1) and (2), and sum the loss value to L_n.

Algorithm 1 Calculation of loss of NST

```
//Set the mini-batch size to batch_size
//model_output is the output of classifier layer
for i in range(len(batch_size)) do
    l_2^1 = model_output[i]
    label1 = target[i]
    for j in range(len(batch_size)) do
        l_2^2 = model_output[j]
        label2 = target[j]
        //If X_1 and X_2 have different labels
        if label1!= label2
            W = α × dis(label1, label2)
            L_n += W × cossim(l_2^1, l_2^2)
        endif
    endfor
endfor
```

3. The training procedure of WNSCL model

Google provides a few kinds of pre-trained BERT models. As the scale of the downstream task is not too big, WNSCL only uses the basic version of BERT as the encoder, which includes 12 transformer blocks. This subsection mainly introduces the training procedure of WNSCL based on the basic version of BERT.

The input X of model WNSCL is a sentence or a text. First, X must be transformed into a form of vector that the model can understand. Google provides a vocabulary file named vocab.txt included in the pre-trained BERT model. In this vocabulary, each word is numbered using the bag-of-words model. In order to pre-train X, WNSCL needs convert X into a vector *ids* (that is, a vector composed of *id* of each word in X) based on this file. The dimension of the vector *ids* is the length of X. And then the vector *ids* is input into the model directly. The following describes in details the specific process of the vector *ids* in WNSCL.

The *ids* is a one-hot encoding vector in fact, which wastes encoding space if the vocabulary is relatively large. Moreover, it cannot reflect the degree of correlation between words because the distances between all words are the same. Therefore, we need a method to convert *ids* into embedding vector [38], [39]. BERT used in WNSCL provides a dedicated embedding layer to carry out this transformation. The embedding layer is a fully connected layer. Its input is a vector whose dimension is size of the vocabulary, and its output dimension denoted as n is the dimension that you want to encode. After processed in the embedding layer, an n-dimensional distributed encoding vector l_{token} is generated. WNSCL processes a text parallelly, the position information of each word is lost. However, the position information of each word always plays a key role in understanding the sentence, so WNSCL uses the sine and cosine functions to generate unique encoding information for the position of each word in a sentence, which is recorded as l_{position} and also an n-dimensional vector. In addition, BERT used in WNSCL also generates a vector l_{segment} for sentence pair related to the tasks distinguishing which sentence each word belongs to. And then WNSCL adds l_{token}, l_{position} and l_{segment} up to generate an n-dimensional vector as the real input of WNSCL as shown in equation (3), which is denoted as $\text{WNSCL}_{\text{input}}$.

$$\text{WNSCL}_{\text{input}} = l_{\text{token}} + l_{\text{position}} + l_{\text{segment}} \tag{3}$$

Based on the above analysis, it can be seen that one word is represented using an n-dimensional vector. In order to represent a sentence, WNSCL arrays all of these n-dimensional vectors corresponding to each word of the sentence into a matrix. And then WNSCL iteratively encodes this matrix 12 times (corresponding to 12 transformer blocks) into the output matrix (referred to as *bert_output* or l_1 in which each word carries other words' information via making use of self-attention mechanism and multi-headed attention mechanism that BERT takes use of. Matrix *bert_output* contains all words' vectors of a sentence.

Since the classifier layer is a fully connected layer, it cannot accept the matrix as input, so the matrix *bert_output* cannot be directly input to the classifier layer. To solve above problem, WNSCL adds a special symbol [**CLS**] at the beginning of each sentence based on the idea of BERT, which can be regarded as a meaningless symbol. After 12 times encoding, the vector corresponding to [**CLS**] also included in *bert_output* contains the information of all the words in a sentence, which can be used for classification tasks alone without using the whole matrix of *bert_output*.

Next, the generated vector of [**CLS**] is passed into the classifier layer which is a fully connected layer. The classifier layer deals with the vector and outputs a C-dimensional vector l_2 in which the value of C is the number of classes.

So far, a forward propagation process of WNSCL is completed. Next, we need to calculate the loss of the model, and then propagate it back along the model to update the parameters. For the main task, this paper directly uses the cross entropy loss function to calculate the loss (L_m) between l_2 and targets. L_n is calculated using the method mentioned in Section III.2. The total loss of WNSCL (L) is calculated by summing L_m and L_n using equation (4).

$$L = l_m + l_n \tag{4}$$

In order to reach the optimization target, WNSCL back propagates L to each layer, and make use of Adam optimizer to update parameters. The training can be terminated until the model converges after repeating the above steps enough times. Algorithm 2 shows the training procedure.

Algorithm 2 The training procedure of WNSCL

```
//Set the max number of epoch: max_epoch
//Split the mini_batch from train dataset
for i in range(max_epoch) do
    for mini_batch in dataset
        l_1s = BERT(mini_batch)
        l_2s = Classifier_layer(l_1)
        //Calculate the loss of main task using Cross entropy function
        L_m = Cross_entropy(l_2s, targets)
        //Calculate the loss of NST using Algorithm 1
        L_n = Algorithm1(l_2s, targets)
        //Calculate the total loss L
        L+= L_m+L_n
    endfor
    Compute gradient ▽
    Update weights: Adam(▽)
endfor
```

It can be seen that Algorithm 2 uses Algorithm 1 to calculate the loss function value of NST for each batch of dataset. Based on the idea of multi-task learning, the goal of learning the two subtasks at the same time can be achieved via summing the loss of the main task and the loss of NST into the total loss to optimize the model. Algorithm 2 uses a two-level loop training dataset. Each execution of the outer loop represents that the entire training dataset is trained and the parameters of WNSCL are updated once. Each execution of the inner loop represents a mini-batch is trained once.

To better understand the idea and the process procedure of WNSCL, here takes the task SST-5 as an example. In the example, a movie review X in the dataset is input to WNSCL. First, X is encoded as a one-hot vector based on the vocabulary file vocab.txt. Then, the vector is input into WNSCL and calculated forward, and the predicted value l_2 of the model is generated. Since task SST-5 has 5 classes, the predicted value l_2 of WNSCL is set as a 5-dimensional vector, and the subscript corresponding to the largest component of vector l_2 is the label predicted by the model. Then, WNSCL uses vector l_2 to calculate the loss function value of the main task and the loss function value of NST respectively, and back propagates it along the WNSCL to update the parameters.

IV. Experiments

To evaluate the performance of our proposed WNSCL model, we design multiple sets of experiments on 2-class rating classification and multi-class rating classification in different language environments respectively.

1. Experiment environment and settings

The proposed WNSCL is trained by using GTX 1660 Ti GPU. For the pre-trained encoder model, we use BERT implemented via using PyTorch which is a well-known deep learning framework. The used model BERT in WNSCL is the basic version ($n = 768$) provided by Google. In order to test the performance of WNSCL model better, we conducted multiple sets of comparative experiments on the specified English and Chinese datasets. For these two language datasets, WNSCL uses two BERT models called bert_base_uncased and bert_base_chinese both provided by Google as encoders respectively.

This paper compares the performance of WNSCL with AAN mentioned in Section I and BERT with a full connection layer (baseline). We implemented AAN model using PyTorch. The BERT models in AAN and WNSCL both use the same configuration.

2. Datasets

We collect multiple standard datasets for text classification from different sources to verify the performance of WNSCL model. The datasets are as follows shown in Table 1.

Table 1. Statistics on the datasets

Dataset	$\|C\|$	# of train sets	# of validate sets	# of test sets
SST-2	2	8544	1101	2210
SST-3	2	8544	1101	2210
SST-5	5	8544	1101	2210
MR	2	6823	1706	2133
Db-2	2	7000	1000	2000
Db-5	5	15297	3000	7000

In Tabel 1, $|C|$ is number of class. The last three columns are the number of training sets, the number of validation sets, and the number of test sets respectively. The dataset SST and the dataset MR are the English datasets, and the dataset Db is a Chinese dataset.

SST is a fine-grained emotional polarity movie review data collected by Stanford University. SST uses a floating point number between 0 and 1 to express the emotion of a movie review. The larger the corresponding floating point number is, the more positive the emotion is. We divide the interval $[0, 1]$ into n equal parts to get the dataset SST-n. The larger n is, the more delicate the sentiment classification is. In the experiments, the value of n is assigned with 2, 3 and 5, the corresponding datasets are SST-2, SST-3, and SST-5 respect-

ively.

MR is a two-pole sentiment classification dataset of movie reviews. In dataset MR, label 0 identifies negative reviews, and label 1 identifies positive reviews. In addition, this paper uses Douban Movies review datasets [40], [41] including Db-2 and Db-5 as Chinese standard datasets to evaluate the performance of WNSCL in Chinese environment, which is provided by a big Chinese Internet enterprise named Douban. Douban Movies is the largest movie sharing and commenting community in China, which contains millions of videos and profiles of filmmakers. Db-2 and Db-5 are obtained from the official website of Douban, which are two-pole and five-pole sentiment classification datasets respectively. In order to input the entire dataset into the model in batches according to the size of mini-batch for training, we need to divide the train dataset into multiple batches with the size of mini-batch.

3. Network setting details

In order to make the model converge better, we have performed a normalization operation on each layer in WNSCL. In order to make the nodes of each layer obey the standard normal distribution, a layer_normalization layer [42] is added to each layer of WNSCL, which is conducive to the convergence of the model. Due to the complex structure and numerous parameters in WNSCL, the number of samples in the train dataset is relatively small compared to the number of parameters, which is prone to over-fitting. While training the model, we introduce the dropout mechanism [31] in WNSCL and inactivate some nodes temporarily and randomly to improve the generalization ability of the model.

In addition, we use the optimizer Adam [43] to update the parameters during the process of back propagation. Adam is an extension of SGD (stochastic gradient descent) [44]–[46], which can replace the traditional stochastic gradient descent method to update the network weights more effectively. In SGD, it maintains a single learning rate which is not changed during the training process for all weights updating. Different from the classic SGD, Adam maintains a learning rate for each weight in some network, and adjusts each learning rate individually based on the budget of the first and second moments of the gradient during the process of model training.

4. Experimental results

1) Accuracy of models

In rating text classification tasks, accuracy of the prediction value is usually used to measure the performance of the model. And equation (5) is usually used to calculate the prediction value of some model.

$$pre = \max(l_2) \tag{5}$$

where pre is the prediction value of the model, and l_2 is the output of WNSCL, max() is a function which can return the index of the maximum value in a vector. For the test dataset, the ratio of the number of samples whose prediction value is consistent with the target on the test dataset to the total number of samples in the test dataset is the accuracy of the model. Equation (6) shows how to calculate the accuracy.

$$\text{Acc} = \frac{\text{num}(pre_x == target_x)}{\text{num}(\text{dataset})} \tag{6}$$

where num() is a function can calculates the length of a set, x is a sample in dataset. We split the samples into mini-batches with the size 24 discussed above. If the size is bigger, memory overflow problems may occur. And then the accuracy rate samples arecorrectly predicted can be calculated according to equation (6).

Base on the analysis in Section III.2, negative supervision is moved into the classifier layer in WNSCL. To verify the effectiveness of this idea, we design a comparative experiment compared with AAN. In this experiment, we modify the architecture of AAN and move negative supervision into its classifier layer from the output layer of BERT. The modified AAN is called negative supervision on classifier layer (NSCL). And then the difference information of the output value can be generated in the classifier layer, but not involve the magnitude of the difference. The experimental results are shown in Table 2.

Table 2. Evaluation results of performance between NSCL and the other models

Dataset		SST-2	SST-3	SST-5	MR	Db-2	Db-5
Language		English	English	English	English	Chinese	Chinese
Model	Baseline	85.1	68.9	54.0	86.5	81.4	54.7
	AAN	85.6	68.3	53.7	86.8	81.7	54.7
	NSCL	85.9	69.1	54.0	86.8	82.2	55.2

In Table 2, the scores above the Baseline are underlined. It can be seen that the performance of NSCL is better than AAN in most tasks, and its performance is relatively stable regardless of the Chinese environment and the English environment. In English rating text classification tasks, NSCL achieves 0.94%, 0.29%, 0.12% improvement relative to the Baseline in SST-2, SST-3, MR tasks respectively, and 0.35%, 1.2%, 0.56%

improvement relative to AAN in SST-2, SST-3 and SST-5 tasks respectively. For Chinese rating text classification tasks Db-2 and Db-5, NSCL also achieves 0.98%, 0.91% improvement relative to the Baseline, and 0.61%, 0.91% improvement relative to AAN. This shows that the classifier layer has learned the difference between different text labels better, which is beneficial for rating text classification tasks.

In the next experiments, we consider the degree of difference between different text labels and add weight parameter based on NSCL to represent this degree of difference, which is the main idea of WNSCL. We compared WNSCL proposed in this paper with the Baseline model and AAN, and all results are shown in Table 3.

Table 3. Evaluation results of performance between WNSCL and the other models

Dataset		SST-2	SST-3	SST-5	MR	Db-2	Db-5
Language		English	English	English	English	Chinese	Chinese
Model	Baseline	85.1	68.9	54.0	86.5	81.4	54.7
	AAN	85.6	68.3	53.7	**86.8**	81.7	54.7
	NSCL	**85.9**	69.1	54.0	**86.8**	82.2	55.2
	WNSCL	85.2	**69.9**	**55.4**	**86.8**	**82.5**	**56.0**

In Table 3, the scores above the Baseline are underlined and the highest scores are in bold. It can be seen that WNSCL performs better than the Baseline in all datasets, and also better than AAN and NSCL in most datasets. Especially in SST-5 task, WNSCL can obtain 3.2% improvement relative to AAN. The reason is AAN does not take into account the degree of difference between different labels, which may lead to small difference degree for the samples with large label distance, and large difference degree for the samples with small label distance. In addition, AAN learns the difference in the output vector of the encoder, and its classification layer cannot learn the difference information, which also affects the classification performance to a certain extent. However, WNSCL we proposed can not only learn the difference in the output vector of the classifier layer, but control the influence degree of the difference between the output vectors of samples with different labels via adding weights. It also can be seen that WNSCL adding weight mechanism based on NSCL achieves 2.6% accuracy rate improvement relative to NSCL 2.6%.

In SST-3 task, the performance of WNSCL is also improved. It can obtain 1.5%, 2.3%, 1.2% accuracy rate improvement relative to the Baseline model, AAN and NSCL respectively.

In Table 3, for the SST-2 task, WNSCL does not achieve accuracy rate improvement compared with AAN and NSCL. In contrast, the performance of WNSCL reduces. The reason is the label distances between all sample pairs with different labels are the same, that is, the differences between these samples should be learned are the same. In this case, the weight is less than 1, which reduces the proportion of difference information in the loss function and leads to insufficient learning of NST.

WNSCL also performed well in the Chinese datasets. Whether it is a 2-class rating classification task or a 5-class rating classification task, the accuracy rate of WNSCL is always higher than that of the Baseline, NSCL and AAN model. For Db-5 task, WNSCL obtains 2.4%, 2.4%, 1.4% accuracy rate improvement relative to the Baseline model, AAN and NSCL respectively. In Db-2 task, WNSCL obtains less accuracy rate improvement, with only 1.4%, 0.98%, 0.36% improvement relative to the Baseline model, AAN and NSCL respectively. The reason is 2-class task has less difference information, so that WNSCL learns less difference degree compared with multi-class task.

For the same 2-class task, WNSCL can achieve the accuracy rate improvement in Db-2, while reduce the performance in SST-2. It is because single Chinese word has more plentiful information compared with single English word, so that WNSCL can learn more plentiful difference information even in 2-class Chinese task.

In order to measure the effect of the coefficient α in equation (2) on the performance of WNSCL, we designs a set of contrastive experiments to determine the most appropriate value of α. The experimental results are shown in Figs.2 and 3. Fig.2 shows the accuracy rate of WNSCL in 2-class tasks with different α values. Fig.3 shows the accuracy rate of WNSCL in multi-class tasks with different α values.

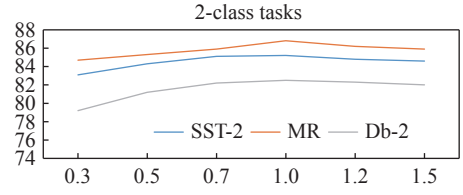


Fig. 2. The accuracy rate of WNSCL in 2-class tasks with different α values.

As can be seen from Figs.2 and 3, in 2-class and 3-class tasks, WNSCL achieves the highest classification accuracy when we set α with value of 1. While in 5-

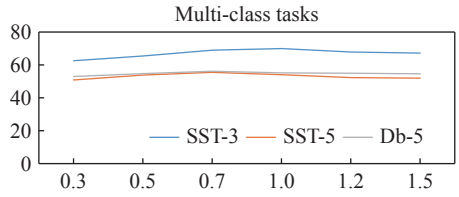


Fig. 3. The accuracy rate of WNSCL in multi-class tasks with different α values.

class tasks, WNSCL reaches the highest classification accuracy when we set α with value of 0.7. This is because the dimensionality of the prediction vector is higher in 5-class tasks, and the calculated cosine distance value is also larger, so the value of NST loss function can be more appropriate while the value of α is smaller. While for 2-class and 3-class tasks, the situation is the opposite.

Base on the above analysis, it can be seen that the accuracy of WNSCL and NSCL is stable in Chinese and English rating text classification tasks. And even compared with NSCL, the performance improvement of WNSCL is more obvious. This is because WNSCL carefully considers the degree of difference between different labels, so that labels with large distance have more different representations. The data in Table 3 proves that and the accuracy rate of WNSCL is not less than that of NSCL in most tasks via adding the weight that represents the degree of label difference.

2) Precision and Recall of models

In addition to accuracy, there are some other indicators also used to evaluate the performance of classification models. This section mainly introduces some common indicators for 2-class and multi-class tasks briefly, which are used to evaluate the model proposed in this paper.

$$\text{precision} = \frac{TP}{TP + FP} \tag{7}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{8}$$

For evaluating the performance of 2-class classification model, precision and recall are two important indicators. Equations (7) and (8) show the relative calculation formulas.

For 2-class classification tasks, their samples can be divided into positive samples and negative samples according to the predictive value, and also divided into true samples and false samples according to whether the predictive value and the target value are consistent. The parameters including *TP*, *FP*, *FN*, *TN* in equations (7) and (8) represent these four sample categories of the 2-class classification tasks: true positive (target value and predictive value are both positive, *TP*), false positive (predictive value is positive, target value is negative, *FP*), false negative (the predictive value is negative, the target value is positive, *FN*), and the true negative (the predictive value and target value are both negative, *TN*).

As shown in equations (7) and (8), precision is the ratio of the number of *TP* samples to the number of all positive predicted samples, which can measure the accuracy of the model's prediction. Recall is the ratio of the number of *TP* samples to the number of samples with a positive target value, which is used to measure the model's ability recognizing a certain category of samples.

Different from 2-class classification tasks, the situation becomes more complicated and it is difficult to measure the performance of multi-class tasks model. At present, there are two ways to measure the performance of multi-class tasks model: i) Treat each multi-class task as multiple 2-class classification tasks and calculate average precision and average recall; ii) Define the multi-class task evaluation standard directly. In order to maintain consistency with the evaluation criteria of the 2-class classification task, the first way is used in this paper. As a result, in order to evaluate the performance of the multi-class classification tasks, we treat each label one by one as the positive label and other labels as negative labels, and then count the values of *TP*, *FP*, *FN*, and *TN*, which are used to calculate precision and recall for different multi-class classification model.

In Fig.4, WNSCL obtains 1.4% and 1.3% average precision improvement relative to the Baseline model and AAN respectively for all the 2-class classification tasks. Moreover, NSCL also achieves better precision than the Baseline model and AAN, which can obtains 2.7% and 2.6% average precision improvement respectively. This further shows that it is effective shifting negative supervision module into the classification layer from the output layer of BERT, which is one of the main ideas this paper proposed. However, it can be seen

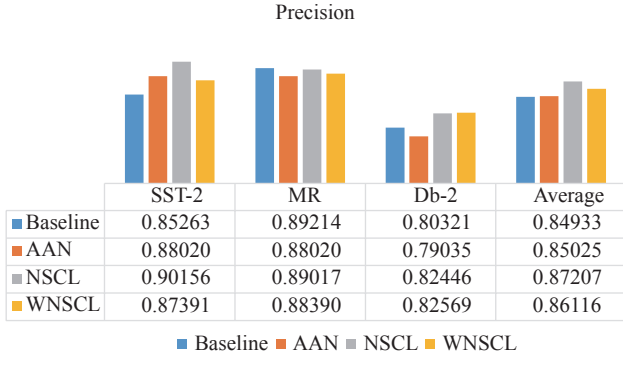


Fig. 4. Precision of each model in all 2-class tasks.

that NSCL achieves better average precision than WNSCL for 2-class classification tasks. The reason is as the distance of different labels is 1 and the value of weight introduces in WNSCL is less than 1, which makes WNSCL generates less value of loss function and achieves worse average precision than NSCL. But in the following section, the experimental results show WNSCL behaviors better than NSCL for multi-class classification tasks.

Fig.5 shows recall of each model for all 2-class tasks. Both WNSCL and NSCL also behavior better than the Baseline model and AAN. Relative to the Baseline model and AAN, NSCL can obtain 1.7% and 0.4% average recall improvement, and WNSCL can obtain 2.7% and 1.4% precision improvement respectively. It can be seen here WNSCL achieves bigger recall than NSCL.

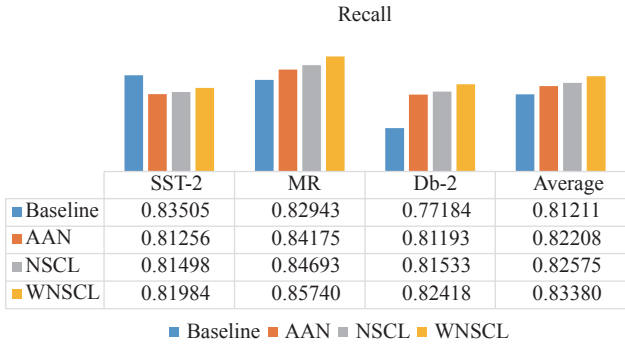


Fig. 5. Recall of each model in all 2-class tasks.

Figs.6 and 7 show the calculating results of precision and recall for all multi-class classification tasks. It can be seen from Fig.6 that WNSCL achieves the greatest average precision in all multi-class classification tasks. Relative to the Baseline model and AAN, WNSCL can obtain 3.4% and 1.4% average precision improvement respectively. Moreover, NSCL also can obtain 3.1% and 1.2% average precision improvement relative to the Baseline model and AAN respectively. Fig.7 shows that WNSCL achieves the biggest recall in all multi-class classification tasks. Relative to the Baseline model and AAN, WNSCL can obtain 3.4% and 5.1% average recall improvement respectively.

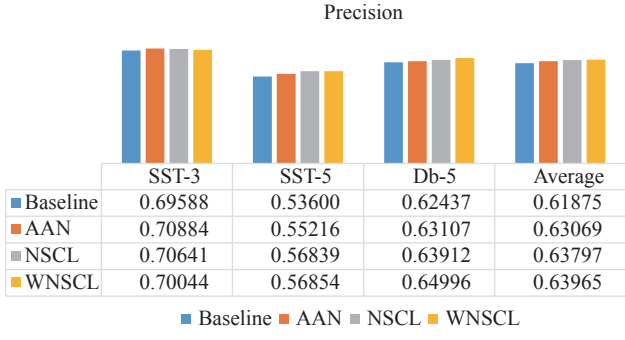


Fig. 6. Precision of each model in all multi-class tasks.

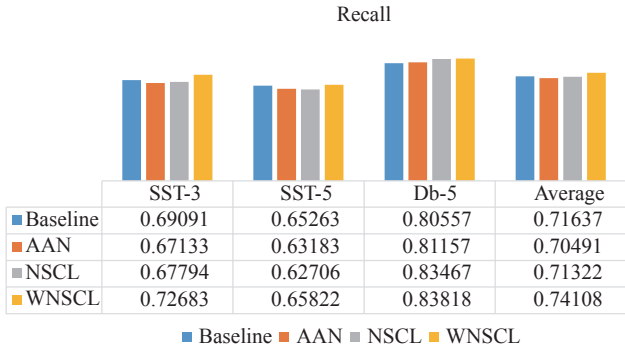


Fig. 7. Recall of each model in all multi-class tasks.

3) F-Measure of Models

Precision and recall are useful parameters evaluating the performance of 2-class and multi-class classification models. However, recision and recall are in conflict with each other to some extent. The reason is as the sum of FP and FN is a fixed value, if FP is big, FN is going to be small, and vice versa, which cannot make these two parameters become big or small at the same time. Another important indicator F-Measure based on precision and recall can solve this problem effectively, which can also be used to measure the performance of the 2-class and multi-class classification models. Equation (9) shows its calculating formula.

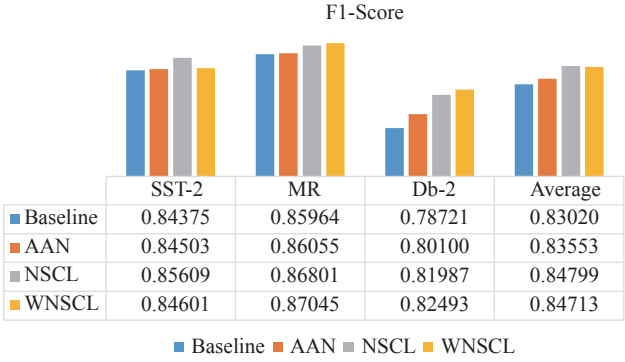
$$\text{F-Measure} = \frac{(\beta^2+1)\cdot \text{precesion}\cdot \text{recall}}{\beta^2\cdot(\text{precision}+\text{recall})} \tag{9}$$

It can be seen that the calculation of F-Measure considers pecision and recall comprehensively in equation (9). For better use F-Measure to evaluate the performance of text classification models, many researchers set parameter β with the value 1 in equation (9), and denote F-Measure as F1-Score. Equation (10) shows the calculating formula of F1-Score. In this section, we analyze performance of different classification models mainly using F1-Score.

$$\text{F1-Score} = \frac{2\cdot \text{precesion}\cdot \text{recall}}{\text{precision}+\text{recall}} \tag{10}$$

Fig.8 shows F1-Score of each model for 2-class classification tasks. It can be seen that both WNSCL and NSCL perform better than the Baseline model and AAN for all 2-class tasks. Relative to the Baseline model and AAN, WNSCL can achieve 2.0% and 1.4% average F1-Score improvement, and NSCL can obtain 2.1% and 1.7% average F1-Score improvement respectively. In Fig.8, for Db-2, the performance of WNSCL is significantly better than that of the Baseline model and AAN, the reason is NST is better at learning difference

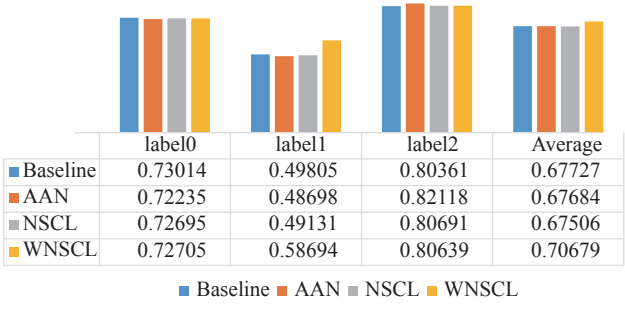
information between labels in Chinese environment. However, it can be seen that NSCL achieves a little better average F1-Score than WNSCL for 2-class classification tasks. The reason is as the distance of different labels is 1 for 2-class classification tasks, and the value of weight introduces in WNSCL is less than 1, which makes WNSCL generate less value of loss function and achieve worse average F1-Score than NSCL.



	SST-2	MR	Db-2	Average
Baseline	0.84375	0.85964	0.78721	0.83020
AAN	0.84503	0.86055	0.80100	0.83553
NSCL	0.85609	0.86801	0.81987	0.84799
WNSCL	0.84601	0.87045	0.82493	0.84713

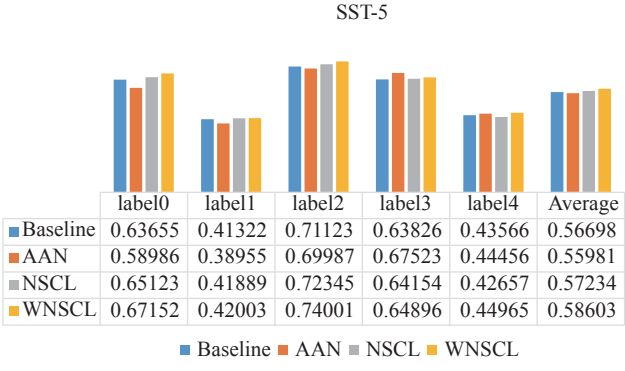
Fig. 8. F1-Score of each model for 2-class tasks.

Figs.9–11 show F1-Score of each label in different multi-class classification tasks. For all multi-class classification tasks, WNSCL achieves bigger F1-Score for nearly all labels than that of the Baseline model and AAN. In order to compare F1-Score of WNSCL, the Baseline model and AAN directly, we average F1-Score on every multi-class classification task for each model,



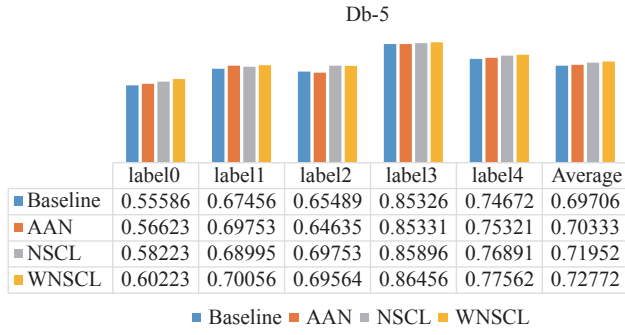
	label0	label1	label2	Average
Baseline	0.73014	0.49805	0.80361	0.67727
AAN	0.72235	0.48698	0.82118	0.67684
NSCL	0.72695	0.49131	0.80691	0.67506
WNSCL	0.72705	0.58694	0.80639	0.70679

Fig. 9. F1-Score of each label in SST-3 task.



	label0	label1	label2	label3	label4	Average
Baseline	0.63655	0.41322	0.71123	0.63826	0.43566	0.56698
AAN	0.58986	0.38955	0.69987	0.67523	0.44456	0.55981
NSCL	0.65123	0.41889	0.72345	0.64154	0.42657	0.57234
WNSCL	0.67152	0.42003	0.74001	0.64896	0.44965	0.58603

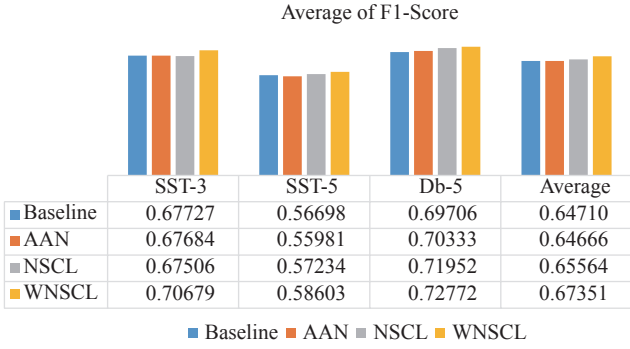
Fig. 10. F1-Score of each label in SST-5 task.



	label0	label1	label2	label3	label4	Average
Baseline	0.55586	0.67456	0.65489	0.85326	0.74672	0.69706
AAN	0.56623	0.69753	0.64635	0.85331	0.75321	0.70333
NSCL	0.58223	0.68995	0.69753	0.85896	0.76891	0.71952
WNSCL	0.60223	0.70056	0.69564	0.86456	0.77562	0.72772

Fig. 11. F1-Score of each label in Db-5 task.

which is shown in Fig.12. Relative to the Baseline model and AAN, WNSCL can obtain 4.1% and 4.2% average F1-Score improvement respectively. Even NSCL also can obtain 1.3% and 1.4% average F1-Score improvement respectively.

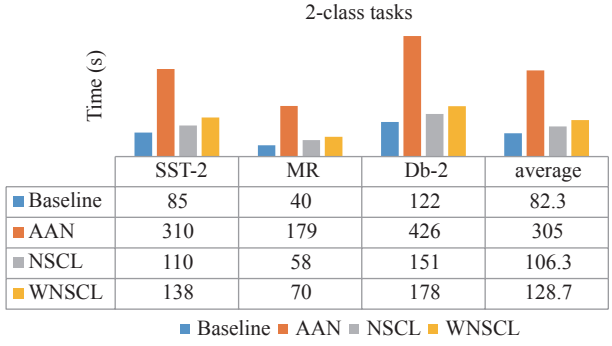


	SST-3	SST-5	Db-5	Average
Baseline	0.67727	0.56698	0.69706	0.64710
AAN	0.67684	0.55981	0.70333	0.64666
NSCL	0.67506	0.57234	0.71952	0.65564
WNSCL	0.70679	0.58603	0.72772	0.67351

Fig. 12. Average F1-Score of each label in multi-class tasks.

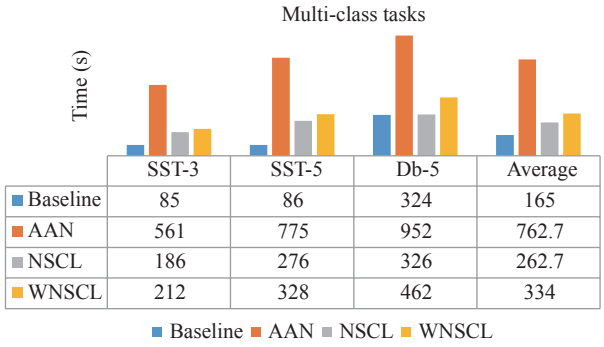
4) Efficiency

In addition to measuring the performance of the proposed model, this paper also analyzes its efficiency. Compared with the 768-dimensional output vector of BERT, the dimensionality of the prediction vector output by the classifier layer is much smaller. As WNSCL shifts negative supervision to the classifier layer, the training time of WNSCL is greatly reduced and the efficiency is improved. In order to measure the training efficiency of these models, we measure the average time executing an epoch for each model. The experimental results are shown in Figs.13 and 14.



	SST-2	MR	Db-2	average
Baseline	85	40	122	82.3
AAN	310	179	426	305
NSCL	110	58	151	106.3
WNSCL	138	70	178	128.7

Fig. 13. Train time of an epoch in 2-class tasks.



	SST-3	SST-5	Db-5	Average
Baseline	85	86	324	165
AAN	561	775	952	762.7
NSCL	186	276	326	262.7
WNSCL	212	328	462	334

Fig. 14. Train time of an epoch in multi-class tasks.

It can be seen from Fig.13 and 14 that the training efficiency of NSCL and WNSCL is significantly higher than that of AAN. For 2-class tasks, the average training time of NSCL and WNSCL reduce 65.1% and 57.8% respectively relative to AAN. For multi-class tasks, the average training time of NSCL and WNSCL reduce 65.6% and 56.2% respectively relative to AAN. Compared with the Baseline model, the training time of WNSCL increases a lot due to its calculation of the loss function of NST is more time-consuming. In addition, as the limited capacity of memory, it is necessary for AAN to reduce the mini-batch size for it is hard to carry out many high-dimensional vector operations, which makes AAN lower training efficiency.

5) Epoch Analysis

Fig.15 shows why the value of epoch is 100 mentioned in Section IV.3. It can be seen that the change trend of the loss function value is almost converged as the value of epoch is 100 for SST-2, SST-3, SST-5, MR, Db-2 and Db-5 tasks in Fig.15.

6) Applicability of WNSCL

In addition to BERT, there are some other typical pre-trained language models including ALBERT [47] and RoBERTa [48]. In order to verify the applicability of WNSCL, we still conduct several sets of experiments via combining WNSCL with ALBERT and RoBERTa respectively. The text classification model combining WNSCL with ALBERT is called WNSCL_AL. The other model is called WNSCL_Ro.

The comparison of the performance between these three models including WNSCL, WNSCL_AL and WNSCL_Ro can be seen from Figs.16 and 17. In Figs.16 and 17, for all tasks of the experiments, the accuracy and F1-Score of these three modes are almost same.

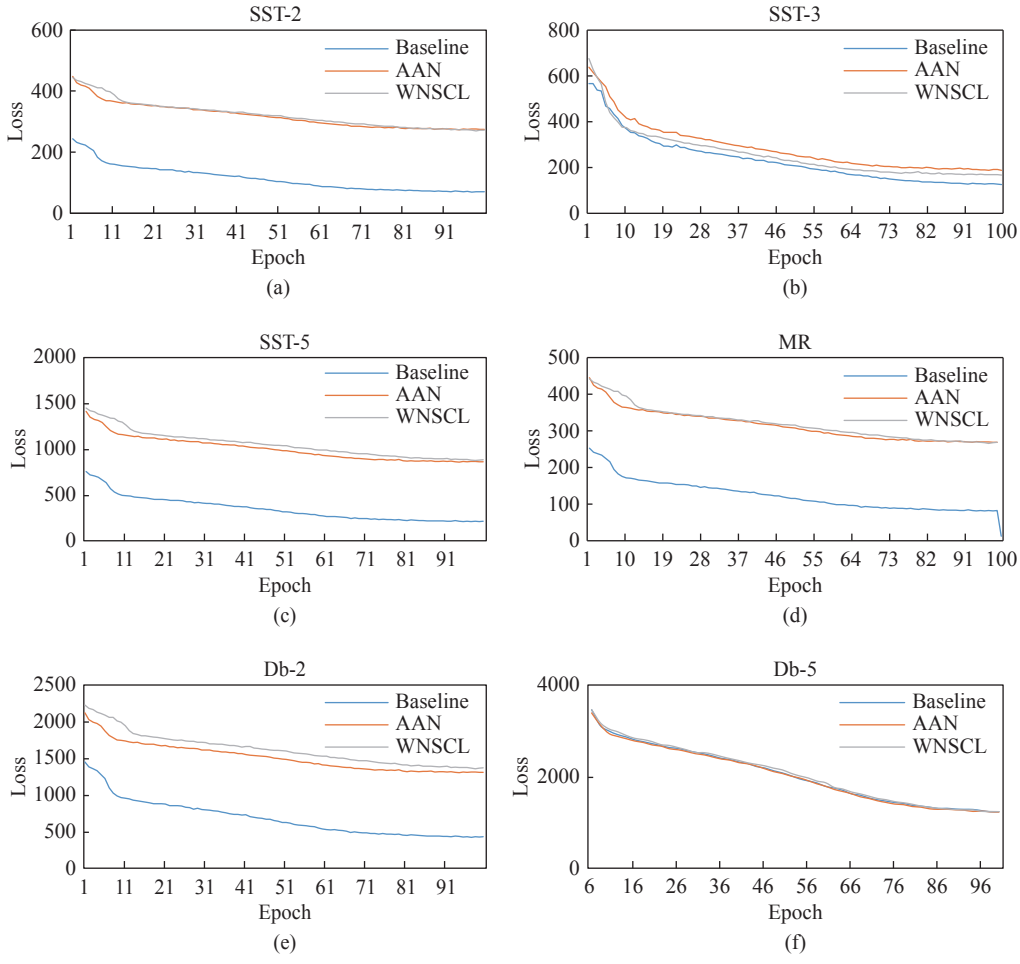


Fig. 15. Change trend of loss function value.

This shows that the weighted negative supervision method proposed in this paper can obtain good performance improvement while it is combined with other pre-trained text classification model, which can furtherly verify applicability and effectivity of our proposed method.

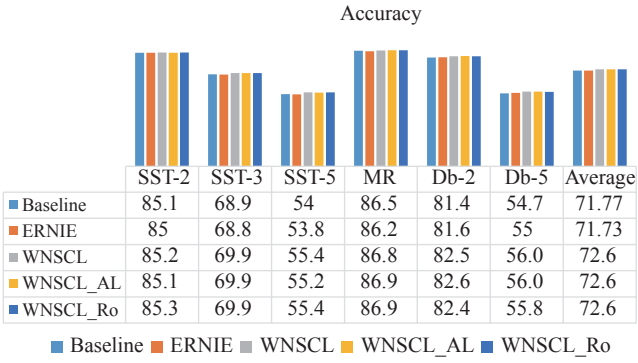


Fig. 16. Accuracy of ERNIE, WNSCL, WNSCL_AL and WNSCL_Ro for all tasks.

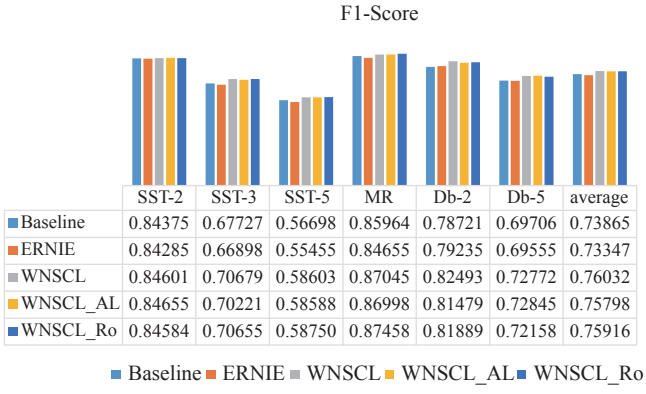


Fig. 17. F1-Score value of ERNIE, WNSCL, WNSCL_AL and WNSCL_Ro for all tasks.

ERNIE [49] is another typical pre-trained language model proposed by a Chinese big company Baidu in 2019, which is a variant model of BERT. It is mainly optimized for the characteristics of Chinese. We also perform several experiments to compare the performance of WNSCL with ERNIE. In Figs.16 and 17, it can be seen that the average accuracy and the average F1-Score of ERNIE for the English data set are slightly lower than those of BaseLine (BERT). And the comprehensive performance of WNSCL surpasses ERNIE.

V. Conclusions

We propose a rating text classifier model WNSCL with weighted negative supervision on the classifier layer in this paper. WNSCL supervises negative samples on the classifier layer, so that the classifier layer also can learn the difference information between labels. In addition, WNSCL assigns weight to the loss of negative samples in NST according to the distance between labels, which makes WNSCL can learn the degree of difference between labels. It can be seen from multiple sets of experiments that WNSCL can achieve better performance in rating text classification tasks compared with the Baseline model and AAN the state-of-the-art method in 2-class and multi-class rating classification tasks. Relative to the Baseline model and AAN, WNSCL can obtain 4.1% and 4.2% average F1-Score improvement respectively. In addition, relative to AAN, WNSCL also can reduce the average training time up to 57.8% and 56.2% for 2-class and multi-class classification tasks respectively. Moreover, the experimental results show for more classes classification, better performance WNSCL can achieve. As a result, the experimental results prove that our proposed model has good generalization ability in English and Chinese rating text classification tasks.

In future work, we will make use of WNSCL to complete more tasks including multi-label classification tasks, sentence pair matching tasks and natural language generation tasks. In order to achieve better results in various NLP tasks, the weight assignment mechanism and model structure should be improved in our future research work.

References

[1] A. Onan, "Ensemble of classifiers and term weighting schemes for sentiment analysis in Turkish," *Scientific Research Communication*, vol.s1, no.1, pp.1–12, 2021.

[2] G. Singh, B. Kumar, L. Gaur, *et al.*, "Comparison between multinomial and Bernoulli naïve Bayes for text classification," in *Proceedings of the 2019 International Conference on Automation, Computational and Technology Management (ICACTM)*, London, UK, pp. 593–596, 2019.

[3] A. W. Haryanto, E. K. Mawardi, and Muljono, "Influence of word normalization and chi-squared feature selection on support vector machine (SVM) text classification," in *Proceedings of the 2018 International Seminar on Application for Technology of Information and Communication*, Semarang, Indonesia, pp.229–233, 2018.

[4] A. Onan, "An ensemble scheme based on language function analysis and feature engineering for text genre classification," *Journal of Information Science*, vol.44, no.1, pp.28–47, 2018.

[5] Y. Ibrahim, E. Okafor, B. Yahaya, *et al.*, "Comparative study of ensemble learning techniques for text classification," in *Proceedings of the 1st International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS)*, Abuja, Nigeria, pp.1–5, 2021.

[6] WU Yujia, LI Jing, SONG Chengfang, *et al.*, "Words in pairs neural networks for text classification," *Chinese Journal of Electronics*, vol.29, no.3, pp.491–500, 2021.

[7] E. Dinan, A. Fan, L. Wu, *et al.*, "Multi-dimensional gender bias classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

(EMNLP), Online, pp.314–331, 2020.

[8] D. Mekala and J. B. Shang, "Contextualized weak supervision for text classification," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp.323–333, 2020.

[9] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, MD, USA, pp.655–665, 2014.

[10] D. H. Shen, Y. Z. Zhang, R. Henao, *et al.*, "Deconvolutional latent-variable model for text sequence matching," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, pp.5438–5445, 2018.

[11] H. Y. Wu, Y. Liu, and S. Y. Shi, "Modularized syntactic neural networks for sentence classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, pp.2786–2792, 2020.

[12] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol.14, no.2, pp.179–211, 1990.

[13] Z. C. Yang, D. Y. Yang, C. Dyer, *et al.*, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, CA, USA, pp.1480–1489, 2016.

[14] D. Yogatama, C. Dyer, W. Ling, *et al.*, Generative and discriminative text classification with recurrent neural networks, arXiv preprint arXiv: 1703.01898, 2017, doi: 10.48550/arXiv.1703.01898.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol.9, no.8, pp.1735–1780, 1997.

[16] Z. H. Huang, W. Xu, and K. Yu, Bidirectional LSTM-CRF models for sequence tagging, arXiv preprint arXiv: 1508.01991, 2015.

[17] K. Greff, R. K. Srivastava, J. Koutník, *et al.*, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol.28, no.10, pp.2222–2232, 2017.

[18] Q. Chen, X. D. Zhu, Z. H. Ling, *et al.*, "Enhanced LSTM for natural language inference," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, pp.1657–1668, 2017.

[19] Z. Q. Geng, G. F. Chen, Y. M. Han, *et al.*, "Semantic relation extraction using sequential and tree-structured LSTM with attention," *Information Sciences*, vol.509, pp.183–192, 2020.

[20] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol.78, no.10, pp.1550–1560, 1990.

[21] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Montreal, Canada, pp.3104–3112, 2014.

[22] J. Gehring, M. Auli, D. Grangier, *et al.*, "Convolutional sequence to sequence learning" in *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp.1243–1252, 2017.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, CA, USA, pp.1–15, 2015.

[24] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, CA, USA, pp.6000–6010, 2017.

[25] J. Devlin, M. W. Chang, K. Lee, *et al.*, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, Minneapolis, MN, USA, pp.4171–4186, 2019.

[26] Y. Arase and J. Tsujii, "Transfer fine-tuning: A BERT case study," in *Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China, pp.5393–5404, 2019.

[27] C. Sun, X. P. Qiu, Y. G. Xu, *et al.*, "How to fine-tune BERT for text classification?," in *Proceedings of the 18th China National Conference Chinese Computational Linguistics*, Kunming, China, pp.194–206, 2019.

[28] Y. G. Xu, X. P. Qiu, L. G. Zhou, *et al.*, Improving BERT fine-tuning via self-ensemble and self-distillation, arXiv preprint arXiv: 2002.10345, 2020, doi: 10.48550/arXiv.2002.10345.

[29] Y. Kuratov and M. Arkhipov, Adaptation of deep bidirectional multilingual transformers for Russian language, arXiv preprint arXiv: 1905.07213, 2019, doi: 10.48550/arXiv.1905.07213.

[30] S. Ohashi, J. Takayama, T. Kajiwara, *et al.*, "Text classification with negative supervision," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp.351–357, 2020.

[31] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, Lake Tahoe, NV, USA, pp.2814–2822, 2013.

[32] R. Caruana, "Multitask learning," *Machine Learning*, vol.28, no.1, pp.41–75, 1997.

[33] M. T. Luong, Q. V. Le, I. Sutskever, *et al.*, "Multi-task sequence to sequence learning," in *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico, pp.1–10, 2016.

[34] L. Chen, D. H. Chen, F. Yang, *et al.*, "A deep multi-task representation learning method for time series classification and retrieval," *Information Sciences*, vol.555, pp.17–32, 2021.

[35] Z. J. Gao, A. Feng, X. Y. Song, *et al.*, "Target-dependent sentiment classification with BERT," *IEEE Access*, vol.9, pp.154290–154299, 2019.

[36] S. Yadav, S. Ramesh, S. Saha, *et al.*, "Relation extraction from biomedical and clinical text: Unified multitask learning framework," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol.19, no.2, pp.1105–1116, 2020.

[37] X. D. Liu, P. C. He, W. Z. Chen, *et al.*, "Multi-task deep neural networks for natural language understanding," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp.4487–4496, 2019.

[38] G. Y. Wang, C. Y. Li, W. L. Wang, *et al.*, "Joint embed-

ding of words and labels for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, pp.2321–2331, 2018.

[39] J. Lilleberg, Y. Zhu, and Y. Q. Zhang, "Support vector machines and Word2vec for text classification with semantic features," in *Proceedings of the 14th International Conference on Cognitive Informatics & Cognitive Computing*, Beijing, China, pp.136–140, 2015.

[40] J. Yang and B. Yecies, "Mining Chinese social media UGC: a big-data framework for analyzing Douban movie reviews," *Journal of Big Data*, vol.3, no.1, article no.3, 2016.

[41] J. H. Li and W. Y. Chen, "Text sentiment analysis of Douban Moive reviews," *Journal of Hanjiang Normal University*, vol.41, no.6, pp.80–86, 2021.

[42] L. J. Ba, J. R. Kiros, and G. E. Hinton, Layer normalization, arXiv preprint arXiv: 1607.06450, 2016, doi: 10.48550/arXiv.1607.06450.

[43] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv: 1412.6980, 2014, doi: 10.48550/arXiv.1412.6980.

[44] M. A. Zinkevich, M. Weimer, A. Smola, *et al.*, "Parallelized stochastic gradient descent," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, Vancouver, British, pp.2595–2603, 2010.

[45] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, K. R. Müller, Eds., 2nd ed., Springer Berlin, Heidelberg, pp.421–436, 2012.

[46] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Lille, France, pp.448–456, 2015.

[47] Z. Z. Lan, M. D. Chen, S. Goodman, *et al.*, "ALBERT: A lite BERT for self-supervised learning of language representations," in *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2019.

[48] Y. H. Liu, M. Ott, N. Goyal, *et al.*, RoBERTa: A robustly optimized BERT pretraining approach, arXiv preprint arXiv: 1907.11692, 2019, doi: 10.48550/arXiv.1907.11692.

[49] Y. Sun, S. H. Wang, Y. K. Li, *et al.*, ERNIE: Enhanced representation through knowledge integration, arXiv preprint arXiv: 1904.09223, 2019, doi: 10.48550/arXiv.1904.09223.

ZHANG Jun was born in 1978. He received the Ph.D. degree from Computer School of Wuhan University, China, in 2018. He is a Professor of East China University of Technology, China. His research interests include performance and energy optimization of GPU architecture, and high performance computing. (Email: zhangjun_whu@whu.edu.cn)

QIU Longlong was born in 1997. He received the B.S. degree in computer science and technology from Huaibei Normal University, China, in 2019. He is currently a postgraduate of East China University of Technology, China. His research interests include natural language processing. (Email: qiulonglong@ecut.edu.cn)

SHEN Fanfan was born in 1987. He received the B.S. degree from Three Gorges University, China, in 2010 and Ph.D. degree from Computer School of Wuhan University, China. He is an Assistant Professor of Nanjing Audit University, China. His current research interests include performance and energy optimization of non-volatile memory. (Email: ffshen@whu.edu.cn)

HE Yueshun was born in 1971. He received the Ph.D. degree from Nanjing University of Aeronautics and Astronautics, China. He is a Professor of East China University of Technology. His research interests include wireless sensor networks, Internet of things, and artificial intelligence. (Email: heys@ecit.edu.cn)

TAN Hai was born in 1977. He received the B.S. and M.S. degrees from Taiyuan University of Technology, Taiyuan, China, in 2000 and 2003 respectively, and Ph.D. degree from Beijing Institute of Technology, China, in 2016. He is currently a Professor in School of Information Engineering, Nanjing Audit University, Nanjing, China. His current research interests include many-core architecture and big data. (Email: 270602@nau.edu.cn)

HE Yanxiang was born in 1952. He received the B.S. and M.S. degrees in the Department of Mathematics from Wuhan University, China, in 1973 and 1975, respectively, and Ph.D. degree from the Computer School of Wuhan University, China, in 1999. He is a Professor of Wuhan University. His research interests include trustworthy software engineering, performance optimization of multi-core processor, and distribution computing. (Email: yxhe@whu.edu.cn)