Text Matching Model Combining Ranking Information and Negative Example Smoothing Strategies

Xiaodong Cai[1], Lifang Dong[2], Yeyang Huang[3], Mingyao Chen[4]
School of Information and Communication, Guilin University of Electronic Science and Technology, Guilin, China[1, 2, 3]
Guilin Yuanwang Intelligent Communication Technology Co., Ltd., Guilin, China[4]

***Abstract*—Aiming at the problems that current text matching methods are difficult to accurately capture the fine-grained ranking information between texts and the insufficient information interaction between different negative examples, a text matching model combining ranking information and negative example smoothing strategy is proposed. Firstly, it ensures the consistency of the ranking of two sentence representations of the input text obtained after different Dropout masks through Jensen-Shannon Divergence. Secondly, it utilizes the pre-trained SimCSE as the teacher model to obtain coarse-grained ranking information and distills this information into the student model through the ListNet sorting algorithm to obtain fine-grained ranking information. Finally, the negative examples are augmented by a negative example smoothing strategy, which effectively solves the problem of insufficient information interaction between negative examples without increasing the batch size. Experimental results on the standard semantic text similarity task show that the proposed model achieves a significant improvement in the Spearman correlation coefficient evaluation metrics compared with existing state-of-the-art methods, proving its effectiveness.**

Keywords*—*Text matching; ranking information; negative example smoothing strategy; jensen-shannon divergence; listnet sorting algorithm

I. INTRODUCTION

Text matching plays an important role in the field of Natural Language Processing for assessing the similarity or relevance between two text sequences. And the quality of sentence representation is crucial for the text matching task, which can greatly affect the matching effect of the model. Therefore, sentence representation learning has attracted extensive research interest [1,2]. In recent years, with the success of Pre-trained Language Models [3,4], there has been much interest in methods for directly generating sentence representations, such as using [CLS] token embeddings or average token embeddings from the last layer of pre-trained language models. However, several studies [5,6] have found that native sentence representations based on Pre-trained Language Models form a narrow cone in the vector space, which severely limits their representational power and is known as the anisotropy problem. Supervised methods (e.g., SBERT [2]) usually produce better quality sentence representations, but require fine-tuning on large amounts of labeled data. Therefore, to avoid the model's dependence on large amounts of labeled data, unsupervised comparison-based learning methods [7-9] have been proposed and have attracted much attention and exploration.

Among the first methods proposed for unsupervised sentence embedding learning using contrastive learning is SimCSE [10]. This method implicitly assumes that "Dropout" is the smallest data augmentation and assumes that a sentence is semantically more similar to its augmented counterpart than to other sentences. Despite the simplicity of this approach, the performance of SimCSE is surprisingly well, and therefore subsequent research methods are based on SimCSE to further optimize and improve it. These methods usually use different enhancement algorithms to generate positive examples. For example, ESimCSE [11] enhances input samples by simply adding insertions and deletions to words repeatedly; ConSERT [12] effectively improves model matching by comparing multiple data enhancement strategies such as feature culling and random discarding to generate two different enhanced versions of the same sentence, and further combining them with in-batch negative examples. Although these methods achieve satisfactory results in obtaining coarse-grained ranking information between texts, they only consider each sample as a positive and negative example, and have some limitations in ranking highly similar and moderately similar sentences in more detailed distinctions. In practical applications, it is necessary to calculate the similarity between the query sentences and the sentences in the database and classify the sentences with more detailed ranking, which generally covers the levels of highly similar, moderately similar, generally similar and not similar. Refined ranking can effectively improve the performance of search and recommendation systems, enhance the model's ability to differentiate between semantically nuanced texts, and thus improve the relevance and accuracy of matching results, which in turn improves the user experience. Therefore, it is particularly important to learn how to accurately capture fine-grained ranking information of texts from unsupervised data.

In addition, to address the problem of insufficient information interaction between negative examples, related studies consider different enhancement strategies to generate negative examples. For example, SNCSE [13] enables the model to learn the semantic differences between sentences more comprehensively by introducing the negative form of the original sentence as a soft negative example. However, this approach to the selection and construction of soft negative examples may have an impact on model performance. In

practice, the computational complexity of the SNCSE approach is higher, requiring more computational resources and time to train and infer the model. In contrast, SSCL [14] obtains additional negative examples from the middle layer of the pre-trained language model, which helps to improve the quality of sentence representation. However, if the additional negative examples are not properly selected or are too many, they may lead to overfitting of the model and impair its generalization ability. Therefore, when considering the generation of negative examples for adequate comparison between negative examples, the impact of computational resource consumption, the number of negative examples, and other factors on the performance need to be taken into account to obtain the best results.

In summary, to accurately capture the fine-grained ranking information between texts and to address the problem of insufficient information interaction between different negative examples, this paper, inspired by the related literature [15,16] proposes a Text Matching Model with Ranking Information and Negative Example Smoothing Strategy (TMRNS). The model does this by ensuring that the rankings between two textual representations that have gone through different Dropout masks have consistency and minimizing the Jensen-Shannon (JS) Divergence as the learning objective. Meanwhile, TMRNS uses the ListNet [17] sorting method to distill the coarse-grained ranking information from the teacher model into the student model, thus capturing the fine-grained ranking information of the text. Finally, by adding random Gaussian noise as an extension of the negative examples, the full interaction of information between different negative examples is realized, which significantly improves the relevance of the semantic similarity task.

The main contributions and contents of this paper can be summarized as follows:

- Proposed methods include Rank Consistency based on JS Divergence and Ranking Distillation based on the ListNet sorting algorithm, aiming to capture fine-grained ranking information between texts to represent sentences with subtle semantic differences effectively.
- Integrated the Smooth Positive Example Construction method based on dynamic buffers into the TMRNS model to enhance positive example.
- Introduced a Smooth Negative Example Construction method based on Gaussian noise to address the issue of insufficient interaction between negative examples within batches.
- Through empirical validation, it was demonstrated that capturing fine-grained ranking information of texts and the proposed negative instance smoothing strategies complement each other, leading to the development of a text matching model that integrates ranking information and negative instance smoothing strategies.

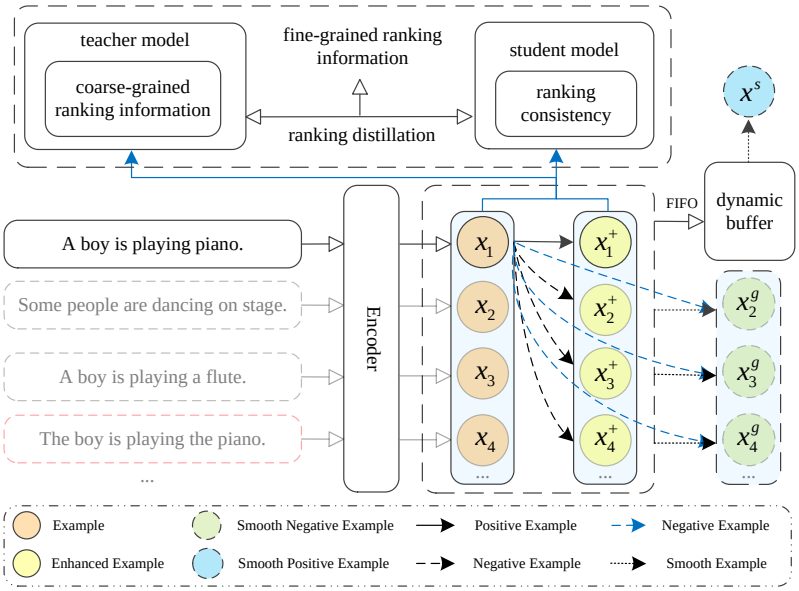

Fig. 1. TMRNS overall framework

II. TMRNS Model Design

A. TMRNS Overall Framework

The overall framework of the TMRNS model is shown in Fig. 1. In the TMRNS model, multiple texts within the same batch are input to the encoder at the same time, and the encoder successively obtains the corresponding example $\boldsymbol{x}_i$ and augmented example $\boldsymbol{x}_i^+$ after two different dropout masks for each text in the input, at which time the teacher model obtains the two lists of similarity scores from both the example and the augmented example to extract coarse-grained information about the ranking of the sentence representations. The student model then utilizes JS Divergence to ensure that the ranking of each sentence representation in the two similarity score lists is consistent. The teacher model then distills its extracted coarse-grained ranking information into the student model using the ListNet sorting algorithm to obtain fine-grained ranking information. Next, the augmented example $\boldsymbol{x}_i^+$ is reconstructed using a first-in-first-out dynamic buffer to obtain the smoothed

positive example x^s. In addition, the negative example x_i^+ is extended by adding random Gaussian noise to obtain the smoothed negative example x_i^g. Finally, the above methods are incorporated into the TMRNS model using different cross-entropy loss functions for the training of the text matching model.

B. Ranking Consistency based on JS Divergence

Although SimCSE's contrastive learning approach performs well in distinguishing between positive and negative examples, it has some limitations in capturing the continuum of sentence similarity. Specifically, it may not work well in distinguishing between very similar and relatively similar sentences. This is mainly because the method does not sufficiently consider the differences between different examples in the batch, leading to weak results in capturing fine-grained ranking information. In this paper, we explicitly model fine-grained ranking information within sentences by using JS Divergence to ensure ranking consistency between two similarity lists for the same text.

The student model modeling diagram is shown in Fig. 2.

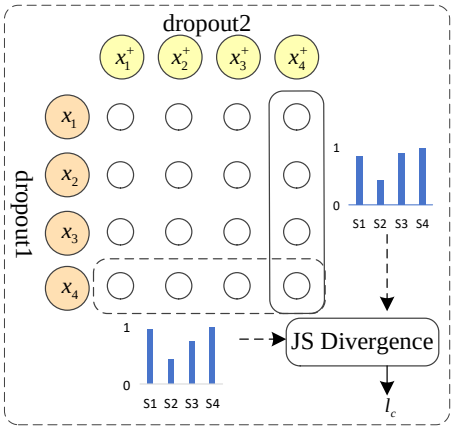

Fig. 2. Student model modeling diagram

Each small white circle in Fig. 2 represents the similarity score of the two texts. There are two sets of sentences denoted x_i and x_i^+, which come from the text vectors obtained from the in-batch examples after two different Dropout masks. For example, the four circles circled in the bottom dashed circle represent the four similarity scores computed for the encoding obtained from text S_4 after dropout1 separately from the encoding obtained from text S_1、S_2、S_3、S_4 after dropout2, and then passed through the classification layer SoftMax to get the bottom bar graph, where it is clear that x_4 and x_4^+ have the highest similarity. The rightmost solid circle similarly demonstrates the similarity computation process. Thus, for each example, two lists of similarities to other examples can first be obtained from its two vector representations, i.e.:

$$T(S_i) = \{s(x_i, x_j^+)\}_{j=1}^N \quad (1)$$

$$T(S_i)' = \{s(x_i^+, x_j)\}_{j=1}^N \quad (2)$$

Where $s(\cdot)$ is the cosine similarity operation and N is the batch size.

Second, based on these two similar lists, their previous probability distributions [17], i.e., the probability that a list element can be ranked first in the sequence among all similar lists, are calculated separately. The calculation formula is as follows:

$$\tilde{T}_{\tau_1}(S_i) = \phi(T(S_i)/\tau_1) \quad (3)$$

$$\tilde{T}_{\tau_1}(S_i)' = \phi(T(S_i)'/\tau_1) \quad (4)$$

Where τ_1 is the temperature hyperparameter and $\phi(\cdot)$ is the previous probability distribution function.

Finally, the consistency of the ranking is ensured by minimizing the JS Divergence between the two previous probability distributions. The specific procedure involves taking the mean of the two distributions as the intermediate value, and then calculating the average of the KL scatter between the two distributions and the intermediate value to finally obtain the corresponding loss l_c. The calculation formula is as follows:

$$\begin{aligned} l_c &= \sum_{i=1}^{N} JS(Q_i \,\|\, V_i) \\ &= 0.5 * \sum_{i=1}^{N} (KL(Q_i \,\|\, \frac{Q_i + V_i}{2}) + KL(V_i \,\|\, \frac{Q_i + V_i}{2})) \\ &= 0.5 * \sum_{i=1}^{N} (Q_i \log(\frac{2Q_i}{Q_i + V_i}) + V_i \log(\frac{2V_i}{Q_i + V_i})) \end{aligned} \quad (5)$$

Where Q_i and V_i denote $\tilde{T}_{\tau_1}(S_i)$ and $\tilde{T}_{\tau_1}(S_i)'$, respectively.

Since the input text undergoes different dropout masks resulting in two textual representations, it is natural to obtain two lists of similarity scores with other texts from these two textual representations, and each element corresponding to these two lists should have the same ranking order. Thus, by enforcing rank consistency between these two lists of similarity scores based on JS Divergence, this task can ultimately be achieved by transforming it into a process of minimizing the loss l_c.

C. Ranked Distillation based on ListNet Sorting Algorithm

Since the SimCSE algorithm performs well in the downstream task, this suggests that although it does not capture fine-grained ranking information, it is still able to capture coarse-grained ranking information. Therefore, a list of similarity scores can be generated using the trained teacher model as pseudo-ranking labels for ranking distillation in all examples of the same batch. In ranking distillation, inspired by the literature in [17], this paper proposes a ranking distillation method based on the ListNet sorting algorithm, which aims to learn finer-grained ranking information from pseudo-ranking labels, which are defined as:

Guangxi Innovation Drive Development Special Project (AA20302001)

$$l_{rank} = \sum_{i=1}^{N} rank(T_s(S_i), T_t(S_i)) \tag{6}$$

Where $T_s(S_i)$ and $T_t(S_i)$ denote the list of similarity scores obtained from the student model and the teacher model, respectively, and $rank(\cdot)$ denotes the sorting algorithm.

The ListNet sorting algorithm used in this paper is trained by maximizing the probability of the correct order of the elements in a sorted list. Specifically, a SoftMax function is used to convert the scores into probability distributions to minimize the cross-entropy loss between the predicted probabilities and the true probability distribution. When the number of examples N is large, the computational complexity increases dramatically because the number of permutations is N! To reduce the computational complexity, the previous probability distribution is used as an alternative, so that it is not necessary to consider all the permutations, but only focuses on the ranking probability of each element in the list. The calculation formula is as follows:

$$l_{listnet} = -\sum_{i=1}^{N} \sigma(T_t(S_i)/\tau_2 \cdot \log(T_s(S_i)/\tau_3)) \tag{7}$$

Where τ_2 and τ_3 are temperature hyperparameter. $\sigma(\cdot)$ denotes the $\mathrm{softmax}$ function.

In order to construct the teacher model, this paper adopts the weighted average similarity scores of two teachers as pseudo-ranking labels, aiming to achieve better transfer and preservation of knowledge sorted by list by integrating the knowledge of two teachers. By a weighted average of similarity scores of two teachers, the opinions of multiple teachers can be integrated, which improves the quality and reliability of pseudo-ranking labels. The specific calculation formula is as follows:

$$T_t(S_i) = \alpha T_t^1(S_i) + (1-\alpha) T_t^2(S_i) \tag{8}$$

Where α is the hyperparameter to balance the weights of the teachers.

D. Smooth Positive Case Construction based on Dynamic Buffers

In this paper, we adopt the first-in-first-out dynamic buffer designed in the literature [18] to construct the smoothed positive examples. This dynamic buffer retrieves sentence embeddings based on cosine similarity and performs a weighted average operation on positive examples to obtain smoothed positive examples. The smoothed positive examples embedding loss is calculated as follows:

$$l_p = -\log \frac{e^{sim(h_i, h_i^{p+})/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j^{p+})/\tau}} \tag{9}$$

Where (h_i, h_i^{p+}) denotes an augmented positive example pair, (h_i, h_j^{p+}) denotes the current positive example and the augmented positive examples of other sentences as negative pairs, and the loss function l_p learns the sentence representation information of the augmented positive example pairs by bringing (h_i, h_i^{p+}) closer and (h_i, h_j^{p+}) farther apart.

E. Gaussian Noise-based Construction of Smoothed Negative Examples

In order to solve the problem of insufficient information interaction between different negative examples, and at the same time consider the impact of computational resource consumption and the number of negative examples, this paper, inspired by the literature [16], proposes a Gaussian noise-based smoothing negative example construction method under the premise of fixing the number of batch examples.

Specifically, a Gaussian noise term is introduced into the InfoNCE loss function commonly used in contrast learning, given the following Gaussian distribution (with mean μ and variance σ^2): $G \sim N(\mu, \sigma^2)$. In this study, M Gaussian noise vectors of the same dimensions as the sentence vectors are randomly selected, and these vectors form high similarity negative pairs with each example in the batch in order to fill and smooth the representation space. These Gaussian noise vectors will not participate in the composition of positive examples. The InfoNCE loss function is improved to be expressed as:

$$l_i = -\log \frac{e^{sim(h_i, h_i^+)/\tau}}{\sum_{j=1}^{N} e^{sim(h_i, h_j)/\tau} + \lambda \cdot \sum_{k=1}^{M} e^{sim(h_i, g_k)/\tau}} \tag{10}$$

Where τ denotes the temperature hyperparameter, $sim(h_i, h_i^+)$ denotes the similarity measure function, g_k denotes the random Gaussian noise vector, M denotes the number of Gaussian noise vectors involved in the computation, and λ denotes the equilibrium hyperparameter.

F. Total Loss Function

In summary, the TMRNS model combines fine-grained ranking information, smoothing positive examples, and smoothing negative examples into a unified text matching model, which effectively improves the performance of the model.

During the training process, each key part of the model is given a loss function l_c, l_{rank}, l_p and l_i. These cross-entropy loss functions are jointly learned to get the final loss function L, which completes the overall training of the TMRNS model. The total loss function L is formulated as follows:

$$L = l_c + l_{rank} + l_p + l_i \tag{11}$$

III. Experimental Results and Analysis

A. Experimental Dataset and Evaluation Metrics

In order to verify the effectiveness of the TMRNS model, experimental evaluation was conducted on the semantic text similarity task. The task consists of seven datasets: STS2012-STS2016, STS-B, and SICK-R. These datasets contain a series of text pairs, each of which has a manually labeled similarity scoring label ranging from 0 to 5, indicating the degree of semantic similarity between the two texts. A higher value of the scoring label indicates a higher degree of similarity between the texts. The specific information of the dataset is shown in Table I and Table II.

TABLE I. Example Dataset

Text A	Text B	Score
A girl in white is dancing.	A girl is wearing white clothes and is dancing.	4.9
A woman is riding a horse.	A man is opening a small package that contains headphones.	1
Three boys in karate costumes aren't fighting.	Three boys in karate costumes are fighting.	3.3

TABLE II. Statistical Information on the Dataset

Dataset	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R
Example size	3108	1500	3750	8500	9183	8624	9927

In order to evaluate the semantic similarity performance of the model in this paper, the Spearman correlation coefficient is used as the evaluation index. Spearman is a statistical index that measures the correlation between two variables, and its value ranges from -1 to 1, where 1 indicates that the two variables are completely positively correlated, -1 indicates completely negatively correlated, and 0 indicates no correlation.

B. Experimental Environment and Parameter Settings

The TMRNS model was built using the Pytorch deep learning framework with the programming language Python, the operating system Ubuntu18.04.6LTS, and the hardware configuration 11thGenIntelCorei7-11700with2.50GHz×16, NVIDIATITANRTX/PCIe/SSE2.

In this experiment, training is performed based on BERT encoders, with the temperature hyperparameter set to 0.05, the learning rate to 3e-5, the teacher weight α to 0.33, the weight decay rate dropout to 0.2, the balancing hyperparameter to 0.5, the batch size set to 64, and the number of Gaussian noise vectors to 192. Furthermore, for the BERT encoder, SimCSE-BERT-base and SimCSE-BERT-large models are used for the teacher model.

C. Experimental Results

In order to verify the effectiveness and sophistication of the TMRNS model, four representative and competitive unsupervised comparative learning models are selected for comparison in this paper, namely, ConSERT [12], SimCSE [10], SSCL [14] and IS-CSE [18]. In Table III, the Spearman correlation coefficients of these models on the STS series dataset are shown for comparative analysis.

Among the baseline models compared, ConSERT tried four methods for data enhancement, including sentence rearrangement, truncated word truncation features, and random discarding, and finally chose the latter two methods for training. In contrast, SimCSE used only one data enhancement method, namely random discard. This parsimonious design makes SimCSE easy to implement and apply and shows high competitiveness by achieving satisfactory performance on several sentence similarity tasks. SSCL obtains better performance by acquiring negative examples based on the middle layer of the pre-trained encoder. IS-CSE, on the other hand, considers the importance of the positive examples and obtains them by designing a dynamic buffer smoothing the boundaries of the feature space embeddings to smooth positive examples, which effectively improves the model performance.

TABLE III. Comparison of Experimental Results

Model	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg. STS
BERT-base								
ConSERT	64.64	78.49	69.07	79.72	75.95	73.97	67.31	72.74
SimCSE	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
SSCL	71.68	83.50	**76.42**	83.46	78.39	79.03	71.76	77.90
IS-CSE	72.86	**84.02**	76.35	82.64	78.65	79.53	**74.05**	78.30
TMRNS	**74.32**	83.17	75.91	**83.05**	**80.71**	**81.01**	72.91	**78.73**
BERT-large								
IS-CSE	**73.76**	85.06	**78.14**	85.02	79.59	80.43	74.30	78.90
TMRNS	73.39	**85.42**	77.95	**85.41**	**81.14**	**81.30**	**74.99**	**79.94**

The TMRNS model proposed in this paper improves on the IS-CSE model. The model focuses on the capture of fine-grained ranking information between texts, meaning that it is able to capture differences and similarities between texts more accurately, especially when it comes to the recognition of discriminative texts. By capturing fine-grained ranking information, TMRNS improves the semantic discriminative power of the model, which is important for many natural language processing tasks such as information retrieval and dialog systems. Meanwhile, the TMRNS model is reconstructed by adding random Gaussian noise to the negative examples, which realizes the full interaction between the negative examples. This approach not only effectively extends the number of negative examples, but also helps the model to better learn the subtle differences between negative examples, which improves the model's generalization ability when facing real-world complex data. Importantly, this methodological improvement does not add additional computational resource costs, which makes the TMRNS model more tractable and practical in real-world applications. The comparison experimental results are shown in Table III, from which it can be intuitively seen that the TMRNS models all outperform the comparative baseline models, and the average Spearman correlation coefficients based on BERT-base and BERT-large were improved by 0.43% and 1.04%, respectively, compared to

the optimal baseline model., which indicates that the model predicts that the text pair similarity scores between them have a stronger positive correlation with the manually labeled similarity scores, reflecting the effectiveness of the model in this paper.

D. Ablation Experiments

To prove the effectiveness of each key component in the TMRNS model, this paper designs the following variants of TMRNS based on the pre-training model BERT-large and analyzes the following variants of TMRNS for ablation experiments using the control variable method:

- TMRNS-RK: denotes the removal of fine-grained ranking information, and experiments were conducted using the base positive and negative examples, the smoothed negative examples, and the smoothed positive examples.
- TMRNS-GS: denotes the removal of smoothed negative examples and experiments using fine-grained ranking information, base positive and negative examples, and smoothed positive examples.

According to Table IV, the average Spearman correlation coefficient of the TMRNS model on the STS task is significantly better than that of its variants, indicating that each key component in the TMRNS model plays an effective role in improving the model performance and collectively contributes to the model's optimal performance.

TABLE IV. ABLATION EXPERIMENT TABLE

Model	Avg. Spearman
TMRNS	**79.94**
TMRNS-RK	79.30
TMRNS-GS	79.34

In the ablation experiments, it can be found that the average Spearman correlation coefficient of the TMRNS-RK model significantly decreases compared to the TMRNS model, which demonstrates the effectiveness of the negative example smoothing strategy proposed in this paper, which positively affects the model performance by augmenting the negative examples. In addition, the TMRNS-GS model also shows a corresponding decrease in performance for the TMRNS model, which indicates that the approach of considering fine-grained ranking information between texts is effective, especially for highly and moderately similar texts, and greatly enhances the user's experience in practical applications.

In summary, by comparing the results of the ablation experiments, the effectiveness of each component can be concluded, and the model performance is significantly improved after their fusion, which further proves the superiority and reliability of the TMRNS model.

E. Parameter Analysis

For the model proposed in this paper, *M* involves the number of Gaussian noise vectors, which form high-confidence negative pairs with the sentences in the batch. This paper further explores the effect of *M* on the performance of BERT-base based TMRNS.

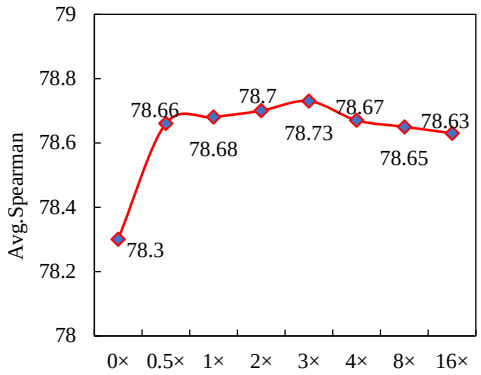

Fig. 3. Impact of hyperparameter *M* based on BERT-base

In this study, optimized hyperparameters of the model were used and only the value of hyperparameter *M* was adjusted. For each *M* value, the model was trained to convergence and then the checkpoints that performed best on the STS validation set were selected for test set evaluation. According to the results shown in Fig. 3, the performance of TMRNS on the test set shows a clear trend of improvement as the value of M increases. In setting *M* as a multiple of the batch size (bs=64), 0× represents the original IS-CSE without using the negative case smoothing strategy. The model reaches its best performance when *M* equals 3, and after that, the model performance starts to decrease. Overall, TMRNS is not sensitive to the value of *M* (the recommended value is less than 8) and is therefore easier to tune in practice.

F. Stability Analysis

To verify the stability of the TMRNS model, experimental evaluations are conducted on seven transfer learning tasks, which are MR, CR, SUBJ, MPQA, SST, TREC, and MRPC. The environment used in this experiment still follows the relevant configurations in subsection III.B. The experimental results for this task are shown in Table V, and the evaluation metric is accuracy. From the table, it can be seen that the average accuracy of the TMRNS model outperforms all the comparison models among the models with different pre-trained encoders, which indicates that the performance of the model in this paper is very stable.

TABLE V. RESULTS OF THE MIGRATION TASK EXPERIMENT (ACCURACY)

Model	MR	CR	SUBJ	MPQA	SST	TREC	MRPC	Avg
BERT-base								
Avg. BERT	78.66	86.25	94.37	88.66	84.40	**92.80**	69.54	84.94
BERT-[CLS]	78.68	84.85	94.21	88.23	84.13	91.40	71.13	84.66
IS-CSE	80.48	85.32	94.67	89.44	85.06	87.40	**75.77**	85.45
TMRNS	**82.69**	**87.18**	**94.99**	**89.78**	**86.99**	88.20	75.71	**86.51**
BERT-large								
IS-CSE	84.27	88.80	95.16	90.04	90.23	91.40	76.29	88.03
TMRNS	**84.80**	**89.27**	**95.40**	**90.23**	**90.72**	**93.20**	**76.75**	**88.62**

IV. Conclusion

In this paper, we presented a text matching model that combines ranking information and negative example smoothing strategies. The model considers the incorporation of fine-grained ranking information into a comparison learning framework that can learn more semantically differentiated sentence representations by ensuring ranking consistency and refining ranking information from teacher models. In addition, the model proposes a negative example smoothing strategy, which smoothes the negative examples by adding Gaussian noise and achieves an adequate comparison between different negative examples without increasing the batch size. The effectiveness of the model is verified by comparison experiments on semantic text similarity tasks, and significant improvement is achieved compared with state-of-the-art models. Meanwhile, after generalizability analysis, the model demonstrates strong stability performance. In the future, our research team will consider applying the method of this paper to more relevant natural language processing tasks.

Acknowledgment

I would like to express my sincere gratitude to the Guangxi Innovation Driven Development Special (AA20302001) Fund Project for their generous support of this study. Without your financial support, this study could not have been carried out successfully. Special thanks to you for providing funding and resources, which provided us with good research conditions. I would also like to thank all the selfless participants and collaborators whose hard work has been an important catalyst for this study. In addition, I would like to thank my supervisor and other research members whose expertise, guidance, and motivation were crucial to me during the research process.

Once again, I would like to express my deep gratitude and respect to all of you mentioned above.

References

[1] X. T. Dinh. “Name2Vec: Name Matching using Character-based with Deep Learning”. Procedia Computer Science, 2023, 230: 305-315.

[2] N. Reimers, I. Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, 2019.

[3] J. D. M. W. C. Kenton, L. K. Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Proceedings of NAACL-HLT. 2019: 4171-4186.

[4] Y. Liu, M. Ott, N. Goyal, et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” 2019.

[5] K. Ethayarajh. “How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings,” Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, 2019.

[6] B. Li, H. Zhou, J. He, et al. “On the Sentence Embeddings from Pre-trained Language Models,” Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2020.

[7] F. Carlsson, A. C. Gyllensten, E. Gogoulou, et al. “Semantic re-tuning with contrastive tension,” International conference on learning representations. 2020.

[8] Y. Zhang, R. He, Z. Liu, et al. “Bootstrapped unsupervised sentence representation learning,” Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021: 5168-5180.

[9] J. Giorgi, O. Nitski, B. Wang, et al. “DeCLUTR: Deep Contrastive Learning for Unsupervised Textual Representations,” Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, 2021.

[10] T. Gao, X. Yao, D. Chen. “SimCSE: Simple Contrastive Learning of Sentence Embeddings,”2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021. Association for Computational Linguistics (ACL), 2021: 6894-6910.

[11] X. Wu, C. Gao, L. Zang, et al. “ESimCSE: Enhanced Sample Building Method for Contrastive Learning of Unsupervised Sentence Embedding,” Proceedings of the 29th International Conference on Computational Linguistics. 2022: 3898-3907.

[12] Y. Yan, R. Li, S. Wang, et al. “ConSERT: A Contrastive Framework for Self-Supervised Sentence Representation Transfer,” Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, 2021.

[13] H. Wang, Y. Dou. “Sncse: Contrastive learning for unsupervised sentence embedding with soft negative samples,” International Conference on Intelligent Computing. Singapore: Springer Nature Singapore, 2023: 419-431.

[14] N. Chen, L. Shou, J. Pei, et al. “Alleviating Over-smoothing for Unsupervised Sentence Representation,” The 61st Annual Meeting Of The Association For Computational Linguistics. 2023.

[15] J. Liu, J. Liu, Q. Wang, et al. “RankCSE: Unsupervised Sentence Representations Learning via Learning to Rank,” The 61st Annual Meeting Of The Association For Computational Linguistics. 2023.

[16] X. Wu, C. Gao, Y. Su, et al. “Smoothed Contrastive Learning for Unsupervised Sentence Embedding,” Proceedings of the 29th International Conference on Computational Linguistics. 2022: 4902-4906.

[17] Z. Cao, T. Qin, T. Y. Liu, et al. “Learning to rank: from pairwise approach to listwise approach,” Proceedings of the 24th international conference on Machine Learning. 2007: 129-136.

[18] H. He, J. Zhang, Z. Lan, et al. “Instance smoothed contrastive learning for unsupervised sentence embedding,” Proceedings of the AAAI Conference on Artificial Intelligence. 2023, 37(11): 12863-12871.