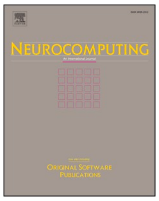
Neurocomputing 577 (2024) 127323

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Hierarchical text classification with multi-label contrastive learning and KNN

Jun Zhang [a,*], Yubin Li [a], Fanfan Shen [b], Yueshun He [a], Hai Tan [b], Yanxiang He [c]

[a] School of Information Engineering, East China University of Technology, Nanchang 330013, China
[b] School of Information Engineering, Nanjing Audit University, Nanjing 211815, China
[c] Computer School, Wuhan University, Wuhan 430072, China

ARTICLE INFO

Communicated by Zidong Wang

Keywords:
Hierarchical text classification
Label hierarchy
Multi-label contrastive learning
KNN

ABSTRACT

Given the complicated label hierarchy, hierarchical text classification (HTC) has emerged as a challenging subtask in the realm of multi-label text classification. Existing methods enhance the quality of text representations by contrastive learning, but this supervised contrastive learning is designed for single-label setting and has two main limitations. On one hand, sample pairs with completely identical labels which should be treated as positive pairs are ignored. On the other hand, a simple pair is deemed as an absolutely positive or negative pair, which lacks consideration about the situation where sample pairs share some labels while having labels unique to each sample. Therefore, we propose a method combining multi-label contrastive learning with KNN (MLCL-KNN) for HTC. The proposed multi-label contrastive learning method can make text representations of sample pairs having more shared labels closer and separate those with no labels in common. During inference, we employ KNN to retrieve several neighbor samples and regard their labels as additional prediction, which is interpolated into the model output to further improve the performance of MLCL-KNN. Compared with the strongest baseline, MLCL-KNN achieves average improvements of 0.31%, 0.76%, 0.83%, and 0.43% on Micro-F1, Macro-F1, accuracy, and HiF respectively, which demonstrates its effectiveness.

1. Introduction

Hierarchical text classification (HTC) is widely applied in natural language processing (NLP), such as news categorization, scientific paper classification, and book categorization. HTC, whose labels are organized in a structured hierarchy, is well known as a special multi-label text classification problem. In general, the label hierarchy of HTC is modeled as a tree [1,2], in which each node corresponds to a label. As depicted in Fig. 1, HTC aims to predict all labels of a given text in a specific label hierarchy, which usually builds one or multiple paths in a top-to-bottom fashion [3]. Labels at a deeper level are called fine-grained labels, which describe the input text more appropriately. While labels at a shallower level are coarse-grained labels that express more general concepts.

In HTC, there usually exists a number of infrequently occurring tail labels [4], especially when the label sets are large. This phenomenon is known as the long-tail effect, which makes it difficult to train an effective classifier as tail labels lack training samples. Fortunately, some existing studies demonstrate the effectiveness of utilizing label correlation in addressing this problem. Therefore, it is a critical challenge for HTC to model the large-scale, imbalanced, and structured taxonomic hierarchy [5]. Recently, some approaches introducing various strategies to coalesce label structure have been proposed, which can solve this issue effectively, such as reinforcement learning [5], graph neural networks [6], and capsule networks [7],[8]. In addition, text representation is a very important basis for a model to efficiently carry out NLP downstream tasks especially like text classification. Some researchers strive to improve text classification performance by obtaining superior text representation via constructing a high-quality text encoder.

Contrastive learning [9,10] is one type of research method whose purpose is obtaining exceptional text representations by concentrating positive sample pairs and separating negative pairs [11], as illustrated in Fig. 2. Wang et al. [12] proposed hierarchy-guided contrastive learning (HGCLR) text classification mechanism, which combined label hierarchy modeling and contrastive learning to achieve excellent text classification performance in HTC. They constructed positive samples of the input text guided by the label hierarchy modeled with a customized

* Corresponding author.
E-mail addresses: zhangjun_whu@whu.edu.cn (J. Zhang), 2021110291@ecut.edu.cn (Y. Li), ffshen@whu.edu (F. Shen), heys@ecit.edu.cn (Y. He), 270602@nau.edu.cn (H. Tan), yxhe@whu.edu.cn (Y. He).

https://doi.org/10.1016/j.neucom.2024.127323
Received 7 May 2023; Received in revised form 16 November 2023; Accepted 21 January 2024
Available online 26 January 2024

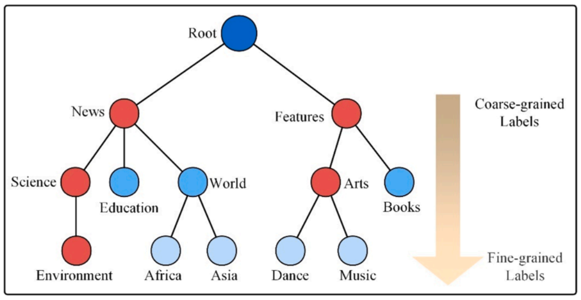


Fig. 1. A real-world sample of HTC on NYT dataset with multi-path and non-mandatory leaf node labels. The red nodes indicate labels assigned to a certain text and they cover two paths of the label hierarchy. Some other HTC datasets may only contain single-path labels. Non-mandatory leaf node prediction indicates that no child of 'Arts' is assigned, which means that child labels cannot be assigned when uncertain. Labels at a shallower level are coarse-grained and those at a deeper level are fine-grained.

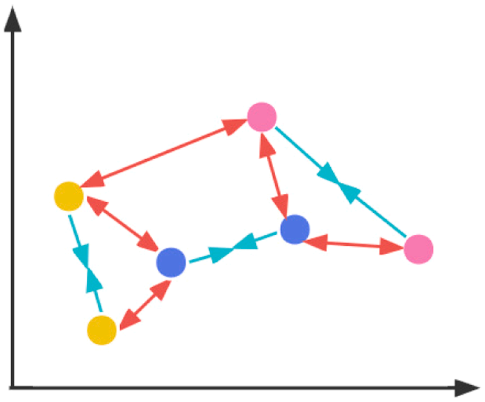


Fig. 2. Diagram of contrastive learning. Two samples with the same color are positive pairs. Two samples with different colors are negative pairs. Contrastive learning aims to pull together representations of positive pairs and push apart those of negative pairs (only part of them is shown).

graph encoder Graphormer, and then generated hierarchy-aware text representations by concentrating the representations of the input text and its generated positive sample, while pushing away the representations of negative pairs. However, supervised contrastive learning used in HGCLR is designed for single-label setting and has some limitations. For this contrastive learning, only the original text and its positive sample are deemed as positive pairs, while sample pairs with completely identical labels are ignored, which should also be treated as positive pairs. In addition, in the existing supervised contrastive learning methods [13, 14], two samples are either absolutely positive or negative for each other. Nevertheless, in HTC, simply regarding these sample pairs as positive or negative is sub-optimal as two samples may share some common labels while each one of them may also possess unique labels of their own [15]. To better understand the above limitations, Fig. 3 is created to display three kinds of sample pairs found in HTC: (a) is an absolutely positive pair; (b) is a pair with both the same and different labels, which should not be simply considered as positive or negative pair; and (c) shows a negative pair.

To sort out these two main limitations, we propose a method combining multi-label contrastive learning with KNN (MLCL-KNN) for HTC. MLCL-KNN can determine the positivity degree of each sample pair based on label similarity instead of manually setting positive pairs. The positivity degree of each sample pair is denoted by a dynamic coefficient. Notice that the bigger the value of the dynamic coefficient, the higher the positivity degree of each sample. When the value of the dynamic coefficient is 1, the sample pair is an absolute positive pair, and when it is 0, the sample pair is absolutely negative. Through utilizing multi-label contrastive learning (MLCL) to train the model, closer representations will be generated for the sample pairs with more shared labels, and the representations of sample pairs with completely different labels will be pushed apart. What is mentioned above focuses on building better training models in the training phase. However, there is rich and valuable knowledge that can be gained from training samples in the inference phase. To fully utilize the knowledge, we exploit the k-nearest neighbor mechanism (KNN) [16] to further improve classification accuracy. Generally, input texts with more common labels are likely to share more common keywords, which results in closer text representations. Namely, sample pairs with closer text representations are expected to have more shared labels, which can be reached by MLCL. Given this observation, during inference, we adopt KNN to retrieve the nearest neighbor samples according to text representations that are generated by the trained model, and output the KNN prediction via summing their ground-truth label vectors which are assigned weights based on the cosine similarity between neighbor samples and test sample. The final prediction incorporates the training model prediction and KNN prediction in a certain proportion. Furthermore, neighbor samples retrieved by KNN are high-quality as MLCL pulls together the text representations of sample pairs with shared labels and pushes apart those with completely different labels. Consequently, KNN can retrieve samples containing more labels with more relevance, which will provide a

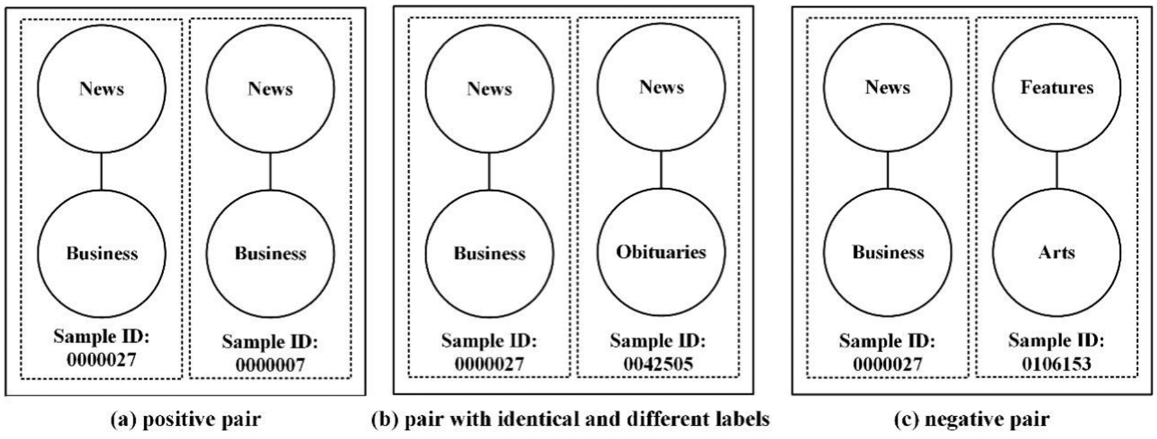


Fig. 3. Examples of positive and negative pairs from the dataset NYT. (a) shows an absolutely positive pair with all labels in common. Two Samples in (b) share the label "News" but also have their unique labels, which should not be simply considered as a positive or negative pair. (c) displays a negative pair without any identical labels.

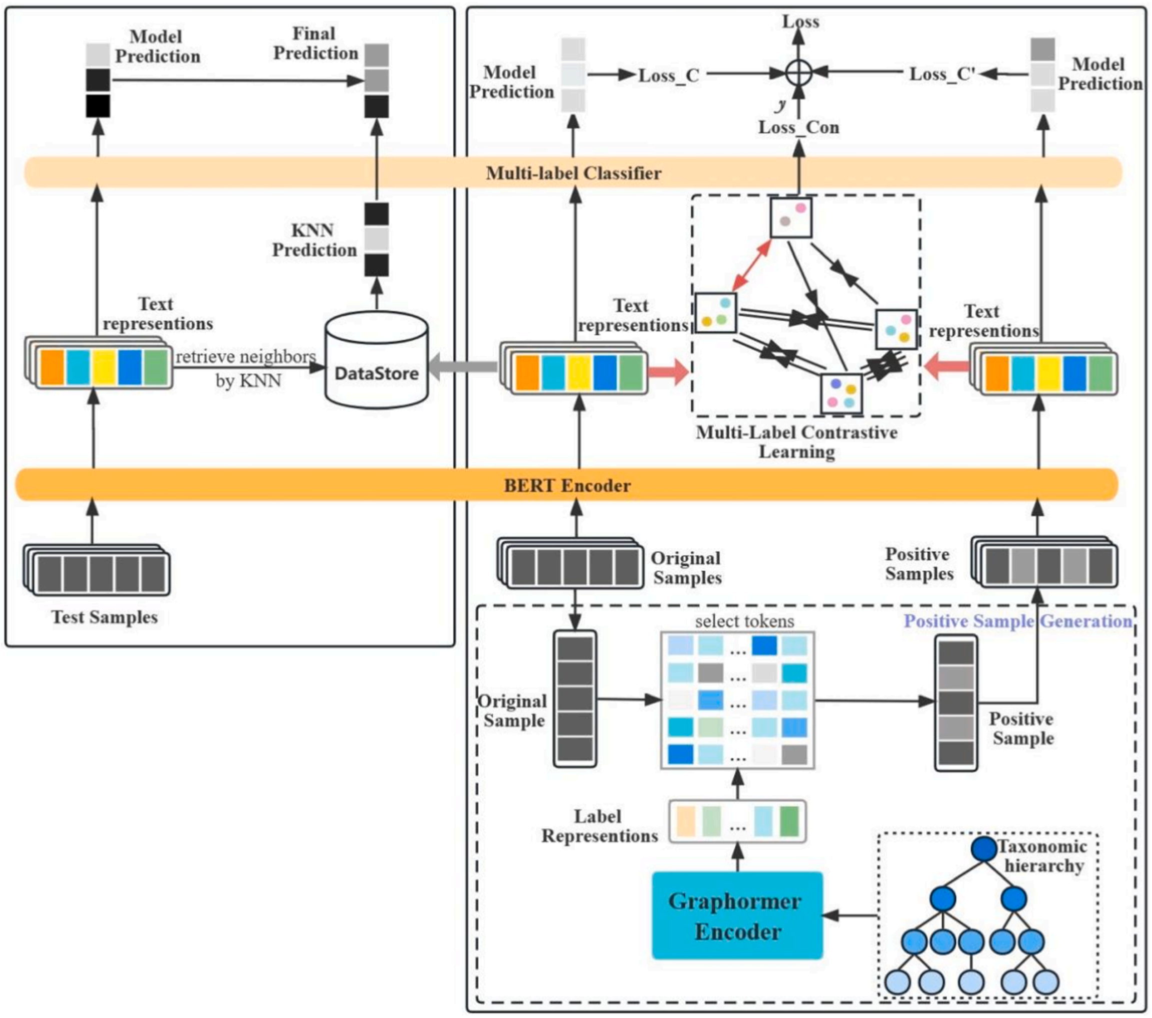


Fig. 4. The overall framework of the proposed MLCL-KNN. During training, positive samples are constructed by masking unimportant tokens guided by label hierarchy, and text representations are improved by multi-label contrastive learning. During inference, neighbor samples are retrieved using KNN, and their labels are viewed as additional prediction which is interpolated into the model output to further improve classification performance.

further boost in classification performance.

Our contributions are the following:

- We propose a multi-label contrastive learning method, which concentrates text representations of sample pairs having more common labels and separating those with absolutely different labels. It is more suitable for HTC tasks compared with supervised contrastive learning used in HGCLR.
- During inference, we employ KNN to retrieve nearest neighbor samples and output KNN prediction by fusing their label vectors according to the corresponding weights, which can further improve the text classification performance.
- Extensive experiments on three benchmark datasets demonstrate the effectiveness of our proposed MLCL-KNN for HTC.

The rest of the paper is organized as follows: Section 2 presents an overview of the related work. Next, our proposed MLCL-KNN is described in detail in Section 3. Then we design some experiments, report the experimental results, and provide the corresponding discussion in Section 4. Finally, Section 5 draws a conclusion and summarizes promising future research directions.

2. Related work

2.1. Hierarchical text classification

Existing methods for HTC are commonly divided into three categories: flat approach, local approach, and global approach [17]. The simplest approach for HTC is the flat approach, which absolutely ignores the taxonomic hierarchy and treats HTC as a general multi-label classification problem [18]. However, this method is sub-optimal as it fails to utilize the overall information of label hierarchy. Local method usually trains classifiers for each category or level. Banerjee et al. [19] trained a classifier per class and initialized the child models using the parameters of the parent model through transfer learning. Similarly, Shimura et al. [20] also adopted the transfer learning method but trained a multi-classifier per level. Nevertheless, local method has two main shortcomings. One is error propagation, which makes errors easy to be propagated downwards the label hierarchy while using a local method with the top-down class-prediction strategy. Another is the high computational cost. Commonly, local method needs to train the vast array of classifiers, which leads to a substantial increase in the number of model parameters and results in high computational cost. Global methods, most of which are based on flat methods, place more emphasis on leveraging holistic information. They usually utilize structural information of label hierarchy and build only one classifier to predict all the labels of a given text at the same time. Many researchers put their eyes on solving HTC problems through global methods, most of which encode label hierarchy in a global view. Zhou et al. [21] exploited the prior hierarchy knowledge of labels to learn label representations for building a hierarchy-aware global model. Chen et al. [22] formulated HTC as a semantic matching problem and utilized graph neural network (GCN) [23] as label encoder. Ma et al. [24] proposed to integrate information of label hierarchy to learn label-specific representations from texts, and to conduct semantic interaction learning among these representations based on dual GCN. In addition, some researchers proposed hybrid approaches with both local and global features via combining local and global methods. Wehrmann et al. [25] first proposed a hybrid method for HTC, which is used to optimize local and global loss at the same time to obtain local and global information about label hierarchy.

2.2. *Contrastive learning*

Contrastive learning is initially proposed as a weakly supervised approach for representation learning in the realm of computer vision. Later, contrastive learning has also gained widespread recognition in NLP, particularly in improving sentence representations [9,10,26]. A crucial factor influencing the effectiveness of contrastive learning applied in NLP is the way to construct positive pairs [27,28]. Kim et al. [29] utilized the hidden representations from intermediate layers of BERT [30] as pivots that [CLS] token embeddings acting as BERT sentence vectors should be close to or be away from. Wang et al. [31] generated both positive and negative samples by substituting representative words in sentences with its synonym and antonym. These methods all achieve excellent sentence representations via contrasting positive and negative pairs. Our proposed approach takes use of contrastive learning to learn more outstanding text representations for better classification performance.

Some researchers seek to incorporate supervised information about labels into contrastive learning to enhance the performance of text classification. Wang et al. [12] built label-guided and hierarchy-involved positive samples for contrastive learning to generate hierarchy-aware text representations for superior performance in HTC. Chen et al. [32] pulled closer text representations with the same category and pushed away those with different categories by contrastive learning to boost few-shot text classification. Both of the above methods are designed for single-label text classification. For contrastive learning designed for multi-label setting, Bai et al. [33] proposed supervised contrastive learning that selected sample features as anchors and their corresponding label embeddings as the positive and negative samples, which could capture the label dependence and align latent spaces for both features and labels. Xie et al. [34] built the multi-label joint loss to improve the semantic representations by utilizing supervised contrastive learning. We propose an effective multi-label contrastive learning method, which designs a dynamic coefficient based on label similarity for each sample pair and maximizes the cosine similarity of text representations of the sample pairs with shared labels.

2.3. *KNN mechanism*

The KNN mechanism is a simple but effective machine learning algorithm. Recently, it has achieved striking success in many NLP tasks such as text classification [15], language modeling [35], and machine translation [36–38]. In these applications, KNN is utilized during inference to retrieve text representations produced by a converged model. Khandelwal et al. [35] leveraged the KNN mechanism to retrieve the predicted neighbors of samples, which can augment the language model. Kassner et al. [39] applied nearest neighbors as extra predictions to promote the question answering task. Li et al. [14] utilized KNN in fine-tuning pre-trained models to get better results on General Language Understanding Evaluation (GLUE) [40] datasets. Notably, the space and time overhead is tolerable compared to the performance gains brought by KNN. Besides, contrastive learning mentioned above can help KNN retrieve high-quality neighbors. Accordingly, KNN is used to further promote our model by interpolating relevant labels from nearest neighbors into the output of our proposed model.

3. Proposed method

This section presents the problem formulation of HTC and provides a detailed description of our proposed MLCL-KNN illustrated in Fig. 4. MLCL-KNN consists of two main modules that correspond to the training and inference stages respectively. In Fig. 4, the right part expresses the training process, and the left part describes the inference process.

The whole process of training consists of four stages. (1) Positive sample generation: modeling label hierarchy by a graph encoder called Graphormer [41] and masking unimportant tokens according to label representations. (2) Text encoding: encoding text of original and positive samples by a text encoder BERT. (3) Multi-label contrastive learning: pulling together and pushing apart text representations according to dynamic coefficient based on label similarity. (4) Classifier training: training one multi-label classifier to classify text into multiple categories.

During inference, MLCL-KNN retrieves the nearest neighbor samples taking the use of KNN, and inserts the retrieved labels into the prediction results for further performance promotion.

3.1. *Problem formulation*

In the HTC task, given an input text $x = \{x_1, x_2, \ldots, x_n\}$, the purpose of HTC is to predict a subset y of the complete label set Y corresponding to x, where n denotes the length of the input sequence. The size of set Y is denoted as $|L|$. The subset y contains all target labels corresponding to the text x.

We predefine label hierarchy as a directed acyclic graph $G = (Y, E)$ because it is more convenient to express the encoding process of graph encoder, where node set Y stands for the complete label set and edge set E represents the hierarchical relations between these label nodes. Furthermore, all labels contained in the predicted label subset y belong to one or multiple sub-paths in G, and any label node $y_i \in y$ except the root must have its parent label node in the subset y.

3.2. *Positive sample generation*

Given a token sequence and its labels, it is required for positive samples to mask unimportant tokens that barely impact classification and to retain a few important tokens that can maintain their labels. Our proposed model constructs hierarchy-aware positive samples by modeling the label hierarchy and selecting tokens based on label representations.

3.2.1. *Label hierarchy modeling*

To model the taxonomic hierarchy, our proposed model makes use of a particular graph encoder Graphormer to encode label features. Graphormer is a variant self-attention layer with a Query-Key product matrix, which has spatial encoding and edge encoding. For the encoding process, the feature f_i for each label node y_i should be first initialized using learnable label embedding used to store the information of label dependence plus its name embedding calculated as the average of BERT token embedding. Eq. (1) describes the computing equation of f_i.

$$f_i = Emb^{label}(y_i) + Emb^{name}(y_i) \tag{1}$$

Both label embedding and its name embedding have the same size which is d_h. All label feature vectors should be stacked into a matrix $F \in R^{|L| \times d_h}$ to migrate features by the modified self-attention layer. And then the Query-Key product matrix A^G belonging to the self-attention layer can be calculated via the use of Eq. (2).

$$A_{ij}^{G} = \frac{\left(f_i W_Q^G\right)\left(f_j W_K^G\right)^T}{\sqrt{d_h}} + b_{\phi(y_i, y_j)} + c_{ij}, where \quad c_{ij} = \frac{1}{D}\sum_{n=1}^{D} w_{e_n}, D = \phi\left(y_i, y_j\right) \tag{2}$$

In Eq. (2), A_{ij}^G denotes each element of A^G, and $W_Q^G \in R^{d_h \times d_h}$ and $W_K^G \in R^{d_h \times d_h}$ stand for the weight matrix of query and key respectively. The first part of Eq. (2) calculates the standard scale-dot attention weight. $b_{\phi(y_i,y_j)}$, a learnable scalar indexed by $\phi(y_i, y_j)$, denotes the spatial encoding measuring the connectivity between nodes y_i and y_j. $\phi(y_i,y_j)$ indicates the distance between two nodes y_i and y_j, which is equal to the number of edges between them. c_{ij} is the edge encoding, and $w_{e_n} \in R^1$ represents a learnable weight for each edge. The graph G in our problem is actually a tree, so there is one and only one path $(e_1, e_2, \ldots, e_D)$ between node y_i and y_j in the graph.

Finally, the final label representation L is computed by Eq. (3), which takes use of residual connection & layer normalization.

$$L = LN\left(softmax\left(A^G\right)V + F\right) \tag{3}$$

In Eq. (3), $A^G \in R^{|L|\times|L|}$, $V \in R^{|L|\times d_h}$, and $L \in R^{|L|\times d_h}$. LN denotes layer normalization, F is the residual connection module. The matrix A^G is fed into the *softmax* function and multiplied with the matrix *value*.

3.2.2. Token selecting

Token selecting depends on label features, which is necessary for building positive samples. It needs to reserve a fraction of important tokens that can sustain the labels of the whole sample and replace unimportant tokens that have little influence with a special token. To measure the importance of a token corresponding to a label, it is necessary to compute the attention weight between token embedding and label representation. If the weight exceeds a fixed threshold, the token will be considered as an important one on the label. Otherwise, it will be considered unimportant.

Given a token sequence, $x = \{[CLS], x_1, x_2, \ldots, x_{n-2}, [SEP]\}$ where n is the length of the sequence, $[CLS]$ and $[SEP]$ denote two special tokens that refer to the beginning and the end of sentence respectively. The token embedding of x generated by BERT is expressed as $\{e_1, e_2, \ldots, e_n\}$. To measure the importance of the token x_i on the label y_j, it is necessary to compute the scale-dot attention weight by Eq. (4).

$$A_{ij} = \frac{(e_i W_Q)(l_j W_K)^T}{\sqrt{d_h}} \tag{4}$$

In Eq. (4), $W_Q \in R^{d_h \times d_h}$ and $W_K \in R^{d_h \times d_h}$. In addition, e_i stands for token embedding, and l_j denotes label feature. Next, the probability P_{ij} of x_i belonging to the label y_j can be calculated using the *gumbel_softmax* function [42] which is sampling differentiable. Eq. (5) describes the computing equation of P_{ij}.

$$P_{ij} = gumbel\_softmax\left(A_{i1}, A_{i2}, \ldots, A_{i|L|}\right)_j \tag{5}$$

Given that a sample has more than one label in HTC, we should calculate the probability of a token x_i corresponding to the whole label set y of the sample as the probability summation over all labels in y: $P_i = \sum_{j \in y} P_{ij}$, which measures the importance of x_i on all labels of the sample. Finally, tokens with the probability exceeding a constant threshold ϵ are selected for positive samples $\widehat{x}$. The equation about how to select tokens is as Eq. (6).

$$\widehat{x}_i = \{x_i \quad if \quad P_i > \epsilon \quad else \quad \mathbf{0}\} \tag{6}$$

0 is the special token with an embedding of all zeros in Eq. (6). We replace unimportant tokens with it so that import tokens can keep their original location.

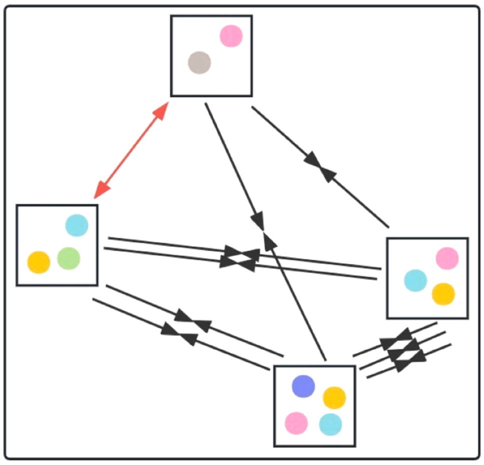


Fig. 5. Example diagram of multi-label contrastive learning. A square represents a sample. Dots with different colors stand for different labels of the sample. Black arrow lines denote making sample pairs with more shared labels closer. And the red arrow line denotes pushing apart those with completely different labels.

3.3. Text encoder

We choose BERT to encode the input text. Given an input token sequence described in Subsection 3.2, hidden representation for each input text can be output by feeding the sequence into BERT encoder, which is shown in Eq. (7).

$$H = BERT(x) \tag{7}$$

In Eq. (7), $H \in R^{n \times d_h}$ is hidden representation and d_h indicates its size. $h_{[CLS]}$, the hidden representation of the first token $[CLS]$, is exploited to represent the entire sequence. In our work, BERT is also used to generate the embedding of positive samples. Referring to Eq. (7), feed positive sample $\widehat{x}$ to BERT to obtain its hidden representation, which is as shown in Eq. (8).

$$\widehat{H} = BERT(\widehat{x}) \tag{8}$$

Similar to the original sample, the representationof the first token $\widehat{h}_{[CLS]}$ is employed to represent the token sequence of the positive sample.

3.4. Multi-label contrastive learning (MLCL)

To obtain excellent text representations that can facilitate classification performance and improve the quality of neighbors retrieved by KNN during inference, the MLCL method is proposed to concentrate representations of sample pairs with more shared labels and separate those with absolutely different labels. We argue that MLCL outperforms supervised contrastive learning that is designed for single-label setting and used in HGCLR.

Contrastive learning in HGCLR essentially regularizes text representation and aims to assign the labels belonging to a training sample to the unseen samples that have the same keywords as the training sample. Nevertheless, this contrastive learning has two main limitations in addressing HTC task. On one hand, it only defines the original sample and its positive sample as positive pairs but ignores sample pairs with perfectly identical labels. On the other hand, it lacks consideration of the situation where sample pairs in HTC may share some labels while also having unique labels of their own. To address these limitations, our proposed MLCL designs a dynamic coefficient based on label similarity to determine the positivity degree of each sample pair instead of manually setting positive pairs. In this way, our proposed multi-label contrastive learning also works even without generating positive samples, which is impossible for contrastive learning used in HGCLR. This strategy makes the text presentations of sample pairs with more labels in common to be closer, and pushes apart those with absolutely different labels, which is illustrated in Fig. 5.

Our proposed method MLCL is presented as an objective function. The method minimizes the loss of multi-label contrastive learning to pull together or push away the text representations of sample pairs. During the computation of loss for multi-label contrastive learning, a non-linear layer is added for a batch of hidden state of the input text and its generated positive sample $(h_i, \widehat{h}_i)$ with the size of N, which is shown in Eq. (9).

$$\begin{cases} a_i = W_2 ReLU(W_1 h_i) \\ \widehat{a}_i = W_2 ReLU(W_1 \widehat{h}_i) \end{cases} \tag{9}$$

$W_1 \in R^{d_h \times d_h}$ and $W_2 \in R^{d_h \times d_h}$ in Eq. (9). There are $2N$ examples that can be expressed as $\boldsymbol{Z} = \{z \in \{a_i\} \cup \{\widehat{a}_i\}\}$ in each batch. The loss of multi-label contrastive learning for each sample pair is calculated by multiplying the standard NT-Xent loss by the dynamic coefficient. Eq. (10) displays the calculation of the loss of multi-label contrastive learning. The dynamic coefficient can be calculated by Eq. (11).

$$L_{ij}^{con} = -\delta_{ij} \log \frac{\exp\left(sim\left(z_i, z_j\right)/\tau\right)}{\sum_{k=1,k\neq i}^{2N} \exp(sim(z_i, z_k)/\tau)} \tag{10}$$

$$\beta_{ij} = \frac{{y_i}^T \cdot y_j}{{y_i}^T \cdot y_i}, \delta_{ij} = {\beta_{ij}}^m \tag{11}$$

In Eq. (10), $sim(\bullet, \bullet)$ is the cosine similarity function which is computed as $sim(A, B) = A \bullet B / ||A|| \ ||B||$. τ is the temperature hyperparameter of contrastive learning. In Eq. (11), β_{ij} is the label similarity whose numerator counts the number of shared labels between the i-th and j-th samples, and its denominator counts the number of labels for the i-th sample. The dynamic coefficient δ_{ij} is the m-th power of β_{ij} where m is an adjustable hyperparameter that can be used to search a more appropriate dynamic coefficient for contrastive learning. We argue that β is not suitable enough to act as the dynamic coefficient for MLCL as it does not provide a very good measure of the relevance between two samples. And overlarge coefficient can be assigned to the sample pair with low label similarity especially when the label hierarchy is relatively deep, which can downgrade the classification performance of HTC. As a result, for obtaining a more appropriate dynamic coefficient δ, the hyperparameter m is used and adjusted, which is demonstrated in Subsection 4.2.3.

For a pair of samples (z_i, z_j), a higher label similarity β_{ij} makes dynamic coefficient δ_{ij} larger, leading to a larger loss L_{ij}^{con}. Therefore, their cosine similarity $sim(z_i, z_j)$ should be optimized to a higher value. Meanwhile, if they do not share any labels ($\delta_{ij} = \beta_{ij} = 0$), the value of L_{ij}^{con} will be zero and their cosine similarity $sim(z_i, z_j)$ should be optimized to lower as they only appear in the denominator term of other pairs. Briefly, MLCL makes the text representations of positive pairs closer and pushes apart those of negative pairs by minimizing the loss of multi-label contrastive learning.

The total contrastive loss for the whole batch is computed as the sum of multi-label contrastive learning losses over all the sample pairs, as described in Eq. (12).

$$L^{con} = \sum_{i=1}^{2N} \sum_{j=1, j\neq i}^{2N} L_{ij}^{con} \tag{12}$$

3.5. Objective function

To train a multi-label classifier for hierarchical text classification, MLCL-KNN flattens the label hierarchy by deeming all nodes as leaf nodes. The hidden feature of text i is fed into a fully connected layer and then followed by a sigmoid function to compute the probability of text i concerning label j. The equation of the probability is as Eq. (13).

$$\widehat{y}_{ij} = sigmoid(W_c h_i + b_c)_j \tag{13}$$

In Eq. (13), $W_c \in R^{|L| \times d_h}$ and $b_c \in R^{|L|}$ denote weights and bias. $\widehat{y}_{ij}$ is the predicted probability. Next, we employ a binary cross-entropy loss as the classification loss of our model, which is shown in Eq. (14).

$$L^C = -\sum_{i=1}^{N} \sum_{j=1}^{|L|} y_{ij} \log\left(\widehat{y}_{ij}\right) + \left(1 - y_{ij}\right) \log\left(1 - \widehat{y}_{ij}\right) \tag{14}$$

In Eq. (14), y_{ij} is the ground-truth label. Similarly, by replacing h_i with $\widehat{h}_i$, the classification loss of the built positive samples which is denoted as $\widehat{L}^C$ can be calculated by Eqs. (13) and (14). Notice that classification loss of positive samples is employed to guide the encoding of graph encoder and the generation of positive samples due to the necessity for positive samples to sustain ground-truth labels.

The final loss function combines the classification loss of original text and its positive sample, as well as the loss of multi-label contrastive learning. The final loss is computed as Eq. (15).

$$L^{total} = L^C + \widehat{L}^C + \lambda L^{con} \tag{15}$$

λ in Eq. (15) is a hyperparameter for balancing the contrastive loss.

3.6. KNN mechanism

During inference, our proposed model which is enhanced through training with MLCL degenerates into the BERT model with a fully connected layer as the output layer for classification. However, using this text classifier alone for prediction still cannot achieve the expected improvement. Considering the rich and valuable knowledge contained in the existing training samples that is helpful for the model to classify correctly, we utilize this knowledge to further improve text classification performance by KNN that retrieves the nearest neighbor samples embodying many relevant labels. The retrieved nearest neighbors are high-quality as MLCL makes the text representations sample pairs sharing more labels closer and separates those with completely different labels.

The implementation of the KNN strategy used in the model MLCL-KNN consists of two steps. One is to construct a datastore, which needs to collect text representations of the training samples generated by the trained model and their corresponding labels. Another is to conduct KNN prediction, which retrieves k training samples with the highest similarity in text representation to the test sample at first and then sums their multi-hot label vectors with weights to obtain the KNN prediction result.

3.6.1. Datastore construction

Let $D = \{(X_i, Y_i)\}_{i=1}^{N}$ be the HTC training set with N samples. X_i represents a text and $Y_i \in \{0, 1\}^{|L|}$ is its label vector where $|L|$ denotes the total quantity of labels. Given a sample $(X_i, Y_i) \in D$ from the training set, h_i is the representation vector generated by the training model of MLCL-KNN. Then the datastore can be built as $D' = \{(h_i, Y_i)\}_{i=1}^{N}$.

3.6.2. KNN prediction

Given a test text x, the trained model MLCL-KNN can generate its text representation h_x. The next step is to calculate the cosine similarity of the text representations between the test sample and all its training samples in the datastore D', and select k samples $\mathscr{N} = \{(h_i, Y_i)\}_{i=1}^{k}$ with the highest similarity as the nearest neighbors. The KNN prediction can be calculated using Eq. (16).

$$\widehat{y}_{kNN} = \sum_{i=1}^{k} \mu_i Y_i, \mu_i = \frac{\exp\left(sim\left(h_i, h_x\right)/\tau'\right)}{\sum_j \exp\left(sim\left(h_j, h_x\right)/\tau'\right)} \tag{16}$$

In Eq. (16), $sim(\bullet, \bullet)$ denotes the cosine similarity, τ' is the temperature hyperparameter of KNN, and μ_i indicates the weight of the i-th neighbor. According to the definition of μ_i, it is clear that a selected neighbor that is more similar to the test sample can be assigned a larger weight. The final prediction combines the model prediction and the KNN prediction, which is computed by using Eq. (17).

$$\hat{y} = \gamma \hat{y}_{kNN} + (1-\gamma)\hat{y}_{Mo} \tag{17}$$

γ in Eq. (17) is the proportion parameter designed to control the weights of model prediction and KNN prediction. $\hat{y}_{Mo}$ is the model prediction output by the classification head of BERT, which is computed by Eq. (13).

The above describes the implementation principles of all the components of MLCL-KNN. To further describe the mechanism of MLCL-KNN, we present the classification procedure of MLCL-KNN in Algorithm 1.

Algorithm 1. the classification procedure of MLCL-KNN.

```
Input: All samples, entire label hierarchy.
Output: Predicted labels.
1:  //In the training stage.
2:  for epoch=1, 2, ..., E do
3:    for batch=1, 2, ..., B do
4:      //Positive samples generation.
5:      Initialize label features $f$ by using Eq. (1).
6:      Stack $f$ as $F$.
7:      Feed $F$ into the Graphormer to get label features $L$ by using Eqs. (2) and (3).
8:      Calculate the probability $P_{ij}$ of a token on a label by using Eqs. (4) and (5).
9:      Calculate the probability of a token on a sample's label subset $y$ as $P_i$ by accumulating $P_{ij}$.
10:     for i=1, 2, ..., n do
11:       if $P_i > \epsilon$ then
12:         $\hat{x}_i = x_i$.
13:       else
14:         $\hat{x}_i = \mathbf{0}$
15:       end if
16:     end for
17:     //Calculate the loss.
18:     Obtain text representations $h_{[CLS]}$ and $\hat{h}_{[CLS]}$ of the original sample and positive sample by using
        Eqs. (7) and (8).
19:     Calculate the multi-label contrastive loss $L^{con}$ by using Eqs. (9) to (12).
20:     Calculate the classification loss of the original sample as $L^C$ by using Eqs. (13) and (14).
21:     Calculate the classification loss of the positive sample as $\hat{L}^C$ by referring to Eqs. (13) and (14).
22:     Calculate the total loss: $L^{total} = L^C + \hat{L}^C + \lambda L^{con}$.
23:     Update model parameters by minimizing $L$.
24:   end for
25: end for
26: //In the inference stage.
27: for batch=1, 2, ..., B do
28:   Feed test samples to the converged model to obtain text representations $h$.
29:   Calculate the model prediction by using Eq. (13)
30:   Retrieve $k$ nearest neighbors which is closest to the test sample.
31:   Calculate the KNN prediction by using Eq. (16).
32:   Calculate the final prediction by using Eq. (17).
33: end for
```

4. Experiment

This section describes the extensive experiments conducted to evaluate the effectiveness of our proposed MLCL-KNN method for HTC task. In Subsection 4.1, we display the experimental settings in detail, including datasets, evaluation metrics, baseline methods, and parameter settings. In Subsection 4.2, numerous experiments are implemented and the experimental results of MLCL-KNN are compared with several baseline methods on three datasets. Meanwhile, we also study and analyze the effectiveness of each module of the MLCL-KNN method in this subsection.

4.1. Experimental settings

4.1.1. Dataset

To conduct a fair and comprehensive evaluation of our model, we implement a series of experiments on three widely studied benchmark datasets: Web-of-Science (WOS) [43], RCV1-V2 [44], and NYTimes (NYT) and a real-world dataset Amazon. For WOS and NYT, we randomly divide them into training/test sets in an 80%/20% partition. In each run, we randomly sample 80% of the training samples to train the model and apply the remaining 20% data as the validation set. For the datasets RCV1-V2 and Amazon with the split training/test sets, 90% of the training samples are randomly sampled for training and the rest 10% for validation. Table 1 displays the details of these benchmark datasets and more introduction about them is as follows.

- **WOS**[1] is crawled from the website Web of Science and is composed of paper abstracts published on this site. WOS a shallow and wide label hierarchy and has two labels per sample. It is suitable for single-path HTC.
- **RCV1-V2**[2] is a news categorization corpus collected from Reuters News, which includes multi-path labels. It consists of 23,149 training samples and 781,265 test samples.
- **NYT**[3] is known as a news classification corpus from the New York Times. Both NYT and RCV1-V2 are applicable for multi-path HTC.
- **Amazon**[4] is a real-world dataset with Amazon product reviews. Each sample of Amazon has three labels but the label "unknown" at the third level belonging to some samples is removed during data pre-processing. Amazon is a dataset with single-path labels.

4.1.2. *Evaluation metrics*

Macro-F1 and Micro-F1 are the main evaluation metrics we use to measure the experimental results. In order to further demonstrate the performance of our proposed model, we also use the metrics including accuracy, and three hierarchical measures HiP, HiR, and HiF.

Micro-F1 is calculated via using both overall precision and recall for all labels. And Macro-F1 expresses the average value of F1 for all labels. Furthermore, Micro-F1 is more influenced by labels appearing more frequently. In contrast, Macro-F1 is more sensitive to the fine-grained labels that have difficulty in being predicted correctly and is applicable to evaluate the performance of classification methods when there is quite imbalanced sample number of different classes. The formulas [45] for calculating Micro-F1 and Macro-F1 are as Eqs. (18) and (19).

$$Micro-F1 = \frac{2\times P\times R}{P+R}, where \quad P = \frac{\sum_{t\in S} TP_t}{\sum_{t\in S}(TP_t+FP_t)}, R = \frac{\sum_{t\in S} TP_t}{\sum_{t\in S}(TP_t+FN_t)} \tag{18}$$

$$Macro-F1 = \frac{1}{|S|}\sum_{t\in S}\frac{2\times P_t\times R_t}{P_t+R_t}, where \quad P_t = \frac{TP_t}{TP_t+FP_t}, \quad R_t = \frac{TP_t}{TP_t+FN_t} \tag{19}$$

TP_t, FP_t, and FN_t in Eqs. (18) and (19) respectively denote the true positive, false positive, and false negative cases of the t-th label in the label set S of a certain dataset. P and R represent the micro-precision and micro-recall. P_t and R_t are the precision and recall of the t-th label.

Accuracy is calculated based on the results of all samples. Eq. (20) is the formula of accuracy [46].

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i\cap\hat{Y}_i|}{|Y_i\cup\hat{Y}_i|} \tag{20}$$

where Y_i and $\hat{Y}_i$ are the ground-truth labels and the predicted labels of the i-th sample, respectively. N is the total number of samples.

All of the above are "flat" metrics that do not take the label hierarchy into account. To incorporate the label hierarchy into performance evaluation and better measure statistical consistency, we adopt three hierarchical measures HiP, HiR, and HiF to evaluate the performance of HTC. Unlike the "flat" metrics which only consider ground-truth and predicted label nodes, hierarchical measures take every node and its ancestors into account. Their formulas [47] are as equations Eqs. (21) to (23).

Table 1
Data statistics. Depth denotes the maximum label hierarchy level. $|L|$ denotes the number of categories. $\overline{|L|}$ means the average amount of labels for each sample. Train/Dev/Test represents the sizes of training/development/test sets.

Dataset	Depth	$\|L\|$	$\overline{\|L\|}$	Train	Dev	Test
WOS	2	141	2.0	30,070	7518	9397
RCV1-V2	4	103	3.24	20,833	2316	781,265
NYT	8	166	7.6	23,345	5834	7292
Amazon	3	580	2.94	36,000	4000	10,000

$$HiP = \frac{\sum_i|C_i\cap\hat{C}_i|}{\sum_i|\hat{C}_i|} \tag{21}$$

$$HiR = \sum_i\frac{\sum_i|C_i\cap\hat{C}_i|}{\sum_i|C_i|} \tag{22}$$

$$HiF = \frac{2\times HiP\times HiR}{HiP+HiR} \tag{23}$$

where C_i and $\hat{C}_i$ are all the ancestors of the ground-truth and predicted labels, respectively. And HiF is a comprehensive evaluation of HiP and HiR.

4.1.3. *Baseline methods*

To prove the effectiveness of our proposed MLCL-KNN, a few representative state-of-the-art hierarchical text classification methods are selected as baselines that are described as follows. Except for BERT, all other baseline methods are the recent works for modeling the label hierarchy for HTC.

- **BERT**[5] [30] is an extensively used pre-trained language model that can be fine-tuned for downstream tasks by simply adding a specific output layer.
- **HiAGM**[6] [21] is a hierarchy-aware global model exploiting soft attention to mix text features and label features. As two variants of HiAGM, HiAGM-LA is based on hierarchy-aware multi-label attention, and HiAGM-TP utilizes the strategy of text feature propagation. We compare our model with the better one HiAGM-TP.
- **HTCInfoMax**[7] [48] imposes regularization constraints on label representations with prior distributions to improve HiAGM-LA.
- **HiMatch**[8] [22] treats text classification as a semantic matching problem between text and labels, which projects text representations and label representations into the same space and conducts text-label semantic matching.
- **HGCLR**[9] [12] integrates label hierarchy information into the text representation by contrast learning, which aims to obtain hierarchy-aware text representation for classification.

Note that, as with MLCL-KNN, all baseline methods use BERT as the text encoder. This is to facilitate a fair comparison between our MLCL-KNN and baseline models.

1 https://doi.org/10.17632/9rw3vkcfy4.6.
2 https://trec.nist.gov/data/reuters/reuters.html.
3 https://doi.org/10.35111/77ba-9× 74.
4 https://www.kaggle.com/datasets/kashnitsky/hierarchical-text-classification.
5 https://github.com/google-research/bert.
6 https://github.com/Alibaba-NLP/HiAGM.
7 https://github.com/RingBDStack/HTCInfoMax.
8 https://github.com/qianlima-lab/HiMatch.
9 https://github.com/wzh9969/contrastive-htc.

4.1.4. Parameters settings

The experiments are conducted on a computer with 20-core 12th Gen Intel(R) Core(TM) i7–12700KF @ 3.60 GHz CPU, NVIDIA GeForce RTX 3080 GPU, and 32 GB RAM. Our proposed model is carried out based on PyTorch in Python 3.8.

For the text encoder, *bert-base-uncased* from Transformer is used as the base framework. We set feature size d_h to 768 and set the attention head of Graphormer to 8. The mini-batch size is set to 5. Adam with a learning rate $lr = 1 \times 10^{-5}$ and default value of other parameters is employed as the optimizer. We leverage early stopping to avoid over-fitting, i.e., stop training if the value of Macro-F1 or Micro-F1 no longer increases for 6 epochs. If a label has a probability higher than a constant threshold (0.5), it can be considered as one of the predicted labels. As for the hyperparameters of our model, $\tau = \tau' = 1$ and $\gamma = 0.6$ are adopted for all models. In addition, $\epsilon = 0.02$, $\lambda = 0.0055$, $m = 2$, $k = 20$ is used for all models on the dataset WOS, $\epsilon = \lambda = 0.005$, $m = 1$, $k = 10$ for those on RCV1-V2, $\epsilon = \lambda = 0.005$, $m = 3$, $k = 20$ on NYT and $\epsilon = 0.002$,$\lambda = 0.001$, $m = 1$, $k = 20$ on Amazon. These hyperparameters can be selected through grid search on the development set.

4.2. Experiments and analysis

4.2.1. Experimental results

To demonstrate the performance of our model, we first compare the most important metrics Micro-F1 and Macro-F1 on three benchmark datasets. Experimental results of MLCL-KNN and other baseline models that use BERT as a text encoder are shown in Table 2. It can be observed that MLCL-KNN significantly outperforms all baselines on WOS and RCV1-V2 and obtains better results than all baseline methods except HGCLR on NYT. This demonstrates the superior capability of MLCL-KNN for HTC.

For more in-depth performance analysis, we note that compared with BERT, our proposed MLCL-KNN achieves an average improvement of 1.3% on Micro-F1 and 2.4% on Macro-F1. Meanwhile, compared with the strong baseline HGCLR, MLCL-KNN separately achieves an improvement of approximately 0.3% and 0.7% on Micro-F1 and Macro-F1 for both WOS and RCV1-V2. For the dataset NYT, MLCL-KNN also performs better and achieves up to 0.4% improvement in terms of Micro-F1, and reaches the comparable Macro-F1 score. Additionally, for Macro-F1 on RCV1-V2, MLCL-KNN outperforms HiMatch which produces the second-best result by 0.4%. These results show the effectiveness of MLCL-KNN. However, in terms of Macro-F1 on NYT, MLCL-KNN only reaches a similar performance compared with HGCLR, which is not as high as expected. This is because the label hierarchy of NYT is deep, while multi-label contrastive learning introduces too many sample pairs with low label similarity as positive pairs when the label hierarchy is deep, which allows coarse-grained labels to be easier distinguished but causes fine-grained labels to become more indistinguishable. This results in a higher Micro-F1 score and a lower Macro-F1 score for NYT.

Next, to further verify the superiority of our model, we also compare MLCL-KNN with BERT and the strongest baseline HGCLR on accuracy and hierarchical measures. Table 3 shows the results. It can be seen that our model achieves the best results on both accuracy and comprehensive hierarchical measure HiF which comprehensively evaluates HiP and HiR. Compared with BERT, MLCL-KNN achieves an average improvement of 1.83% in accuracy and of 1.15% in HiF. And it improves the accuracy by an average of 0.73% and HiF of 0.44% over HGCLR.

Finally, in addition to the above three datasets, we also conduct experiments on the real-world dataset Amazon, and the results are shown in Table 4. For Amazon, MLCL-KNN achieves the best on the main metrics Micro-F1, Macro-F1, accuracy, and HiF. MLCL-KNN achieves improvements of 2.09%, 2.97%, 2.40% and 1.69% on Micro-F1, Macro-F1, accuracy, and HiF respectively over BERT. Compared with the strongest baseline HGCLR, improvements of 0.25%, 1.58%, 1.13%, and 0.41% are obtained respectively.

4.2.2. Ablation study

The main distinction between our model MLCL-KNN and the strong baseline HGCLR includes multi-label contrastive learning and KNN mechanism. To testify the effectiveness of these two modules, we carry out ablation experiments. Table 5 shows all the results of the ablation models.

To demonstrate the effectiveness of multi-label contrastive learning, two control groups are set up: MLCL-KNN vs. *-r.m.* MLCL, and *-r.m.* KNN vs. HGCLR. We find that the model performance decreases after removing the multi-label contrastive learning module (*-r.m.* MLCL), indicating the usefulness of MLCL. With KNN removed (*-r.m.* KNN), our model differs from HGCLR only in terms of contrastive learning approach, allowing for a fair comparison between general contrastive learning and ours. We observe *-r.m.* KNN shows improvement of all metrics over HGCLR on WOS and RCV1-V2.

and increasement of Micro-F1 on NYT. However, the Micro-F1 score on NYT declines. This is due to the excessive number of label hierarchy levels for NYT, which introduces a large number of positive pairs with low similarity for multi-label contrastive learning. This means that multi-label contrastive learning tends to pull together sample pairs with the same coarse-grained labels but different fine-grained labels, causing fine-grained labels to become harder to distinguish and worse performance in Macro-F1 which is sensitive to difficult fine-grained labels. Overall, our proposed multi-label contrastive learning outperforms supervised contrastive learning in HGCLR.

The results of two control groups, MLCL-KNN vs. *-r.m.* KNN as well as BERT-KNN vs. BERT, can reflect the effectiveness of the KNN

Table 2
Micro-F1 and Macro-F1 scores of MLCL-KNN and several baseline models on three benchmark datasets. For a fair comparison, all baselines employ BERT as the text encoder. Note that the best results are marked in bold and the second best underlined.

Model	WOS		RCV1-V2		NYT	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BERT	85.63	79.07	85.65	67.02	78.24	65.62
HiAGM	86.04	80.19	85.58	67.93	78.64	66.76
HTCInfoMax	86.30	79.97	85.53	67.09	78.75	67.31
HiMatch	86.70	81.06	86.33	<u>68.66</u>		
HGCLR	<u>87.11</u>	<u>81.20</u>	<u>86.49</u>	68.31	<u>78.86</u>	<u>67.96</u>
MLCL-KNN	**87.37**	**81.88**	**86.79**	**69.06**	**79.27**	**67.97**

Table 3
Scores for accuracy and hierarchical measures. Note that the best results are marked in bold and the second best underlined.

Dataset	Model	Accuracy	HiP	HiR	HiF
WOS	BERT	82.83	<u>87.87</u>	84.81	86.31
	HGCLR	<u>84.30</u>	87.69	<u>86.04</u>	<u>86.86</u>
	MLCL-KNN	**85.29**	**88.03**	**86.86**	**87.44**
RCV1-V2	BERT	82.28	**87.53**	84.13	85.80
	HGCLR	<u>83.11</u>	86.80	<u>85.93</u>	<u>86.36</u>
	MLCL-KNN	**83.68**	<u>87.16</u>	**86.39**	**86.77**
NYT	BERT	72.96	76.54	79.69	78.08
	HGCLR	<u>73.94</u>	<u>76.93</u>	**81.39**	<u>79.10</u>
	MLCL-KNN	**74.58**	**79.13**	<u>79.72</u>	**79.42**

Table 4
Experimental results on the dataset Amazon. Note that the best results are marked in bold and the second best underlined.

Model	Micro-F1	Macro-F1	Accuracy	HiP	HiR	HiF
BERT	68.69	28.86	61.17	62.10	67.18	64.54
HGCLR	<u>70.53</u>	<u>30.25</u>	<u>62.44</u>	**64.25**	<u>67.47</u>	<u>65.82</u>
MLCL-KNN	**70.78**	**31.83**	**63.57**	<u>64.06</u>	**68.57**	**66.23**

Table 5
Performance when removing some modules of MLCL-KNN. *r.m.* stands for remove. We remove KNN and multi-label contrastive loss by setting $\gamma = 1$ and $\lambda = 1$, respectively.

Ablation Model	WOS		RCV1-V2		NYT	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BERT	85.63	79.07	85.65	67.02	78.24	65.62
HGCLR	87.11	81.20	86.49	68.31	78.86	67.96
MLCL-KNN	**87.37**	**81.88**	**86.79**	**69.06**	**79.27**	**67.97**
-*r.m.* KNN	87.27	81.53	86.56	68.59	78.99	67.85
-*r.m.* MLCL	87.27	81.64	86.48	68.67	79.16	67.12
BERT-KNN	86.88	81.01	86.45	67.99	78.30	67.03

mechanism. Comparing MLCL-KNN with -*r.m.* KNN, decreased performance is shown to point out that the KNN module is beneficial to the performance improvement of MLCL-KNN. Moreover, BERT-KNN, which adds KNN to the model BERT during inference, further verifies the effectiveness of KNN by comparing it with BERT. As shown in Table 5, compared with BERT, BERT-KNN obtains better scores in all metrics on three benchmark datasets.

4.2.3. *Analysis of dynamic coefficient*

In contrast to general contrastive learning, multi-label contrastive learning does not simply consider each sample and its corresponding positive sample as a positive sample pair, but models the complex correlation of multi-label sample pairs via using dynamic coefficient based on label similarity. For the definition of dynamic coefficient, the first consideration is label similarity β_{ij}, which simply measures the similarity between the i-th and j-th samples by the proportion of the quantity of shared labels to the quantity of labels the i-th sample has. However, label similarity cannot adequately express the complex correlation between sample pairs. Therefore, sample pairs with low label similarity should be assigned a smaller dynamic coefficientδto match the correlation more effectively. We use the m-th power of β as the final dynamic coefficientδ. m is an adjustable hyperparameter that can be tuned to obtain the best dynamic coefficient to measure the complicated correlation between sample pairs.

Table 6
Performance on NYT when using δ with different values of hyperparameter m as the dynamic coefficient.

Dynamic Coefficient (δ)	Micro-F1	Macro-F1
$m = 1(\beta)$	79.06	67.70
$m = 2$	79.03	67.76
$m = 3$	**79.27**	**67.97**
$m = 4$	79.13	67.85

To get the best value of the hyperparameter m, we carry out a set of comparative experiments on NYT to test the model performance with different values of m. Results are shown in Table 6. It can be seen that the model performance is not the best when $m = 1$ (i.e., $\delta = \beta$). This suggests that the simple label similarity β is still insufficient to represent the correlation between sample pairs. When $m = 3$, the Micro-F1 and Macro-F1 scores reach their peak. This indicates that the optimal dynamic coefficient δ for multi-label contrastive learning can be obtained by adjusting m.

4.2.4. *Analysis of KNN hyperparameters*

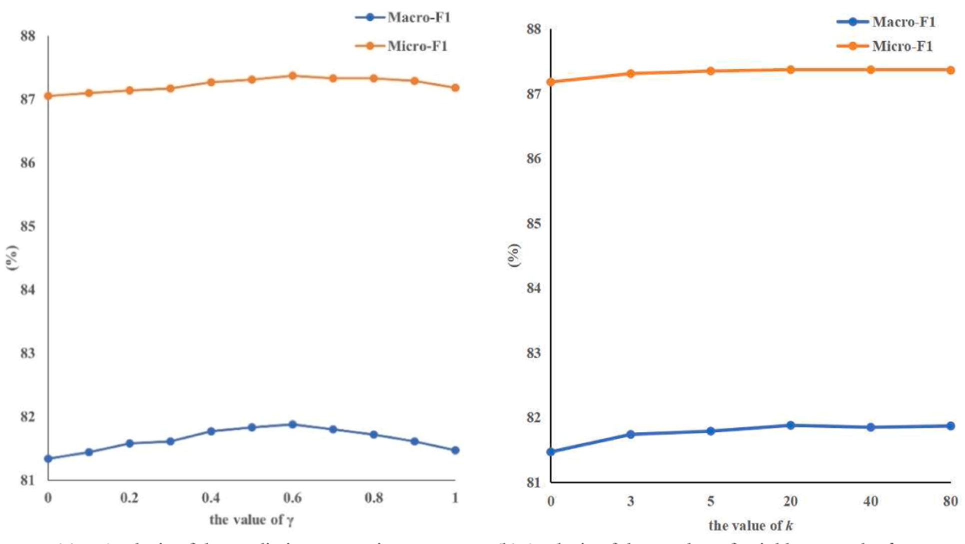
In this subsection, we discuss the setting of hyperparameters of KNN on WOS. As shown in Fig. 6(a), the x-axis represents the value of the proportion parameter γ, which determines the weight of model prediction and KNN prediction, and the y-axis denotes Micro-F1 and Macro-F1. As the value of proportion parameter γ increases, the model performance keeps improving until γ is 0.6 and then declines. This confirms that using only model prediction ($\gamma = 1$) is sub-optimal. In Fig. 6(b), the x-axis indicates the number of nearest neighbors (i.e., k) retrieved by KNN, and the y-axis means Micro-F1 and Macro-F1. Fig. 6(b) shows the trend of model performance with changing the number of KNN neighbors. It can be seen that using neighbor samples ($k > 0$) always yields better performance than using only model output ($k = 0$), and the performance reaches the peak when k is 20. The change of performance is less while k is more than 20, so the value of k is set to 20.

4.2.5. *Effectiveness of multi-label contrastive learning*

As mentioned in the previous description, multi-label contrastive learning can benefit KNN in retrieving high-quality neighbors. To verify this, we plot the T-SNE projections of text representations corresponding to a certain test sample and its 100 nearest neighbors on the dataset WOS with a 2-layer label hierarchy, as illustrated in Fig. 7.

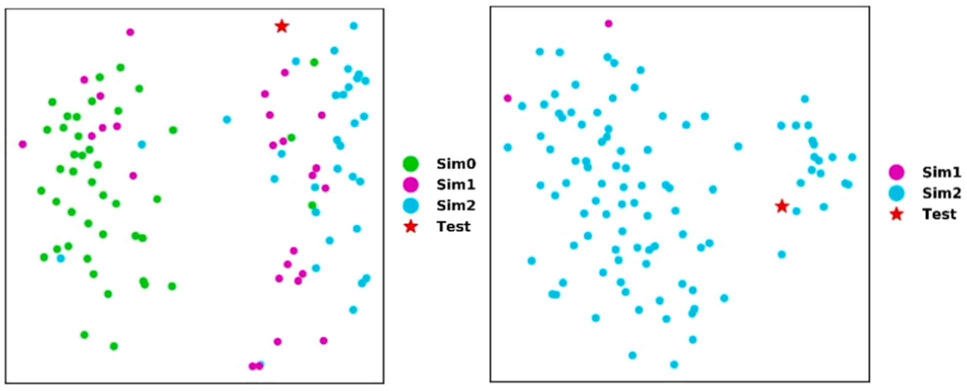
Fig. 7(a) and (b) respectively correspond to the representations generated by BERT-KNN described in Section 4.2.2 and MLCL-KNN. As shown in Fig. 7(a), for BERT-KNN only using BERT.

without multi-label contrastive learning and graph encoder, more than one-third of the nearest neighbors have no labels in common with the test sample. And the remaining neighbors are roughly 50%/50%



(a) Analysis of the prediction proportion γ (b) Analysis of the number of neighbor samples k

Fig. 6. Hyperparameter analysis of KNN on the dataset WOS.



(a) BERT-KNN (b) MLCL-KNN

Fig. 7. T-SNE visualization of the text representations on WOS. The red star stands for the text representation of a certain test sample and dots with the same color are nearest neighbor samples with the same number of shared labels. "Sim* " in the legend means the number of shared labels.

between those sharing one label and those sharing two labels. In contrast, according to Fig. 7(b), almost all neighbors share two labels with the test sample, which means they have completely identical labels. This confirms that the model MLCL-KNN retrieves neighbors that have more common labels with the test sample, which means the fact that our multi-label contrastive learning method indeed makes contributions to retrieving high-quality neighbors by using KNN.

Moreover, for the model BERT-KNN in Fig. 7(a), the nearest neighbor samples with high-similarity text representations have a variety of labels and the distribution of text representations corresponding to different shared label numbers is relatively in chaos, which can lead to more difficulty of distinguishing labels. In comparison, for our model MLCL-KNN in Fig. 7(b), most of the nearest neighbor samples have completely identical labels with the test samples and only two samples sharing one label with the test sample are at the edge of the cluster, which can result in easier classification. These indirectly verify the usefulness of contrastive learning in making sample pairs sharing more labels closer and separating those having no labels in common.

4.2.6. *Effectiveness of KNN*

We select a certain test sample and its neighbors to further illustrate the effectiveness of KNN. Table 7 shows the labels of the test sample and its 20 nearest neighbor samples retrieved by KNN on the dataset NYT with 166 categories. The serial number of one label in the range 0–165 represents one category of NYT. From this table, it can be seen that the neighbors share many labels with the test sample. Concretely, the test sample shares labels numbered as 16, 17, 18, and 19 with all its neighbors, and shares labels 0 and 13 with most of its neighbor samples. The test sample and its neighbor samples have so many common labels that it is useful to consider labels of neighbor samples as extra predictions during inference. In other words, combining KNN prediction with the model output does help to improve classification performance.

Table 7
Labels of a certain test sample and its 20 neighbors retrieved by KNN on NYT dataset. Each serial number in the "Label Number" column stands for a label.

Neighbor Number	Label Number
0(Test Sample)	**0, 2, 13, 16, 17, 18, 19, 47, 76**
1	**0, 2, 13, 16, 17, 18, 19, 47, 76**
2	**0, 2, 13, 16, 17, 18, 19, 47, 76**
3	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, **47,** 48, 50, 73, **76**
4	**0, 13, 16, 17, 18, 19, 76**
5	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, 73, **76**
6	**0, 2, 16, 17, 18, 19,** 72, **76,** 81
7	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, 73, **76**
8	**0, 13, 16, 17, 18, 19, 76**
9	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, **47,** 48, 50, 72, 73,156
10	**0, 2, 16, 17, 18, 19,** 72, **76**
11	**0, 13, 16, 17, 18, 19, 76**
12	**0, 13, 16, 17, 18, 19,** 72, **76**
13	**0,** 4, **13,** 15, **16, 17, 18, 19,** 37, 38, **47,** 73, 75, **76**
14	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, 39, **76**
15	**0,** 4, **13,** 15, **16, 17, 18, 19,** 37, 38, **47,** 75, **76,** 94,138
16	**16, 17, 18, 19,** 50, **76**
17	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, 73, **76**
18	4, **16, 17, 18, 19,** 37, 38, 50, **76,** 138
19	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, **47,** 48, 50, 73
20	**0,** 4, **13, 16, 17, 18, 19,** 37, 38, 73, 75, **76**

5. Conclusion

We propose the model MLCL-KNN combining multi-label contrastive learning with KNN for hierarchical text classification in this paper. Multi-label contrastive learning is more comprehensive and thoughtful so that makes up for the disadvantage of contrastive learning used in HGCLR in addressing HTC problem. Moreover, multi-label contrastive learning is beneficial for KNN to retrieve high-quality neighbors as it concentrates sample pairs that share more labels and separates those with completely different labels. In addition, we apply KNN in the prediction phase to retrieve neighbor samples and consider their labels as extra predictions to interpolate into the model output, which can further improve the text classification performance. A series of experiments on four benchmark datasets confirm the effectiveness of MLCL-KNN and reveal the sources of performance improvement.

However, our method does not achieve the expected improvements on datasets with over deep label hierarchy. The Macro-F1 score of our method is only comparable to the strongest baseline HGCLR on the datasets NYT with over-deep label hierarchy. Therefore, in future research work, we consider introducing negative supervision to address this limitation. Negative supervision is proposed to deal with the conflict between labels and text semantics. Its idea is to separate samples with similar text semantics but different labels. We hope to design a negative supervision method suitable for HTC, so that it can separate sample pairs with different labels without reducing the effect of multi-label contrastive learning, thereby improving the classification performance of the model for the datasets with over deep label hierarchy.

CRediT authorship contribution statement

Li Yubin: Conceptualization, Data curation, Formal analysis, Methodology, Resources, Software, Visualization, Writing – original draft, Writing – review & editing. **Zhang Jun:** Conceptualization, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing. **He Yueshun:** Formal analysis, Methodology, Writing – review & editing. **Shen Fanfan:** Funding acquisition, Methodology, Writing –

review & editing. **He Yanxiang:** Formal analysis, Methodology, Supervision. **Tan Hai:** Methodology, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

Data will be made available on request.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under grant (Nos. 62162002, 61662002), the Natural Science Foundation of Jiangxi Province (No. 20212BAB202002), Basic Science (Natural Science) Research Project of Colleges and Universities in Jiangsu Province (No. 22KJA520004).

References

[1] W. Huang, E. Chen, Q. Liu, Y. Chen, Z. Huang, Y. Liu, Z. Zhao, D. Zhang, S. Wang, Hierarchical multi-label text classification: an attention-based recurrent network approach, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM'19, 2019, pp. 1051–1060. https://doi.org/10.1145/3357384.3357885.

[2] H. Huang, H. Liu, Feature selection for hierarchical classification via joint semantic and structural information of labels, Knowl. - Based Syst. 195 (2020) 105655, https://doi.org/10.1016/j.knosys.2020.105655.

[3] A. Sun, E.P. Lim, Hierarchical text classification and evaluation, in: Proceedings 2001 IEEE International Conference on Data Mining, ICDM, 2001, pp. 521–528. https://doi.org/10.1109/ICDM.2001.989560.

[4] K. Bhatia, H. Jain, P. Kar, M. Varma, P. Jain, Sparse local embeddings for extreme multi-label classification, Adv. Neural Inf. Process. Syst., NIPS (2015) 730–738.

[5] Y. Mao, J. Tian, J. Han, X. Ren, Hierarchical text classification with reinforced label assignment, in: 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, pp. 445–455. https://doi.org/10.18653/v1/D19–1042.

[6] Q. Ma, C. Yuan, W. Zhou, S. Hu, Label-specific dual graph neural network for multi-label text classification, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 3855–3864. https://doi.org/10.18653/v1/2021.acl-long.298.

[7] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li, P. Yu, L. He, Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification, IEEE Trans. Knowl. Data Eng. 33 (2021) 2505–2519, https://doi.org/10.1109/TKDE.2019.2959991.

[8] R. Aly, S. Remus, C. Biemann, Hierarchical multi-label classification of text with capsule networks, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, ACL, Association for Computational Linguistics, 2019, pp. 323–330. https://doi.org/10.18653/v1/P19–2045.

[9] Z. Wu, S. Wang, J. Gu, M. Khabsa, F. Sun, H. Ma, Clear: Contrastive learning for sentence representation, CoRR. abs/2012 15466 (2020), https://doi.org/10.48550/arXiv.2012.15466.

[10] T. Gao, X. Yao, D. Chen, Simcse: Simple contrastive learning of sentence embeddings, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, 2021, pp. 6894–6910. https://doi.org/10.18653/v1/2021.emnlp-main.552.

[11] R. Hadsell, S. Chopra, Y.L. Cun, Dimensionality reduction by learning an invariant mapping, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'06, IEEE, 2006, pp. 1735–1742. https://doi.org/10.1109/CVPR.2006.100.

[12] Z. Wang, P. Wang, L. Huang, X. Sun, H. Wang, Incorporating hierarchy into text encoder: A contrastive learning approach for hierarchical text classification, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL, Association for Computational Linguistics,2022, pp. 7109–7119. https://doi.org/10.18653/v1/2022.acl-long.491.

[13] B. Gunel, J. Du, A. Conneau, V. Stoyanov, Supervised contrastive learning for pre-trained language model fine-tuning, in: 9th International Conference on Learning Representations, ICLR, 2021.

[14] L. Li, D. Song, R. Ma, X. Qiu, X. Huang, KNN-BERT: Fine-tuning pre-trained models with KNN classifier, CoRR. abs/2110 02523 (2021), https://doi.org/10.48550/arXiv.2110.02523.

[15] X. Su, R. Wang, X. Dai, Contrastive learning-enhanced nearest neighbor mechanism for multi-label text classification, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL, Association for Computational Linguistics, 2022, pp. 672–679. https://doi.org/10.18653/v1/2022.acl-short.75.

[16] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1967) 21–27, https://doi.org/10.1109/TIT.1967.1053964.

[17] C.N. Silla, A.A. Freitas, A survey of hierarchical classification across different application domains, Data Min. Knowl. Discov. 22 (2011) 31–72, https://doi.org/10.1007/s10618-010-0175-9.

[18] L. Cai, T. Hofmann, Hierarchical document categorization with support vector machines, in: Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management, CIKM'04, 2004, pp. 78–87. https://doi.org/10.1145/1031171.1031186.

[19] S. Banerjee, C. Akkaya, F.P. Sorrosal, K. Tsioutsiouliklis, Hierarchical transfer learning for multi-label text classification. in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, ACL, Association for Computational Linguistics, 2019, pages 6295–6300. https://doi.org/10.18653/v1/P19–1633.

[20] K. Shimura, J. Li, F. Fukumoto, HFT-CNN: learning hierarchical category structure for multi-label short text categorization, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, 2018, pp. 811–816. https://doi.org/10.18653/v1/D18–1093.

[21] J. Zhou, C. Ma, D. Long, G. Xu, Ning Ding, H. Zhang, P. Xie, G. Liu, Hierarchy-aware global model for hierarchical text classification, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, Association for Computational Linguistic, 2020, pp. 1106–1117. https://doi.org/10.18653/v1/2020.acl-main.104.

[22] H. Chen, Q. Ma, Z. Lin, J. Yan, Hierarchy-aware label semantics matching network for hierarchical text classification, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 4370– 4379. https://doi.org/10.18653/v1/2021.acl-long.337.

[23] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR, 2017.

[24] Q. Ma, C. Yuan, W. Zhou, S. Hu, Label-specific dual graph neural network for multi-label text classification, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 3855–3864. https://doi.org/10.18653/v1/2021.acl-long.298.

[25] J. Wehrmann, R. Cerri, R. Barros, Hierarchical multi-Label classification networks, in: Proceedings of the 35th International Conference on Machine Learning, ICML, 2018, 80:5075–5084.

[26] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, X. Song, COCO-LM: Correcting and contrasting text sequences for language model pretraining, Adv. Neural Inf. Process. Syst. NeurIPS (2021) 23102–23114.

[27] L. Pan, C.W. Hang, A. Sil, S. Potdar, Improved text classification via contrastive adversarial training, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2022 pp. 11130–11138. https://doi.org/10.1609/aaai.v36i10.21362.

[28] M. Alzantot, Y. Sharma, A. Elgohary, B.J. Ho, M.B. Srivastava, K.W. Chang, Generating natural language adversarial examples, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP, Association for Computational Linguistics, 2018, pp. 2890–2896. https://doi.org/10.18653/v1/D18–1316.

[29] T. Kim, K.M. Yoo, S. Lee, Self-guided contrastive learning for BERT sentence representations, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 2528–2540. https://doi.org/10.18653/v1/2021.acl-long.197.

[30] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, 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: Human Language Technologies, NAACL, Association for Computational Linguistics, 2019, pp. 4171–4186. https://doi.org/10.18653/v1/N19–1423.

[31] D. Wang, N. Ding, P. Li, H. Zheng, CLINE: Contrastive learning with semantic negative examples for natural language understanding, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021b, pp. 2332–2342. https://doi.org/10.18653/v1/2021.acl-long.181.

[32] J. Chen, R. Zhang, Y. Mao, J. Xu, ContrastNet: a contrastive learning framework for few-shot text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2022, pp. 10492–10500. https://doi.org/10.1609/aaai.v36i10.21292.

[33] J. Bai, S. Kong, C. Gomes, Gaussian mixture variational autoencoder with contrastive learning for multi-label classification, in: Proceedings of the 39th International Conference on Machine Learning, PMLR, 2022, pp. 1383–1398. ⟨https://proceedings.mlr.press/v162/bai22c.html⟩.

[34] S. Xie, C. Hou, H. Yu, Z. Zhang, X. Luo, N. Zhu, Multi-label disaster text classification via supervised contrastive learning for social media data, Comput.

Electr. Eng. 104 (2022) 108401, https://doi.org/10.1016/j.compeleceng.2022.108401.
[35] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, M. Lewis, Generalization through memorization: Nearest neighbor language models, in: 8th International Conference on Learning Representations, ICLR, 2020.
[36] U. Khandelwal, A. Fan, D. Jurafsky, L. Zettlemoyer, M. Lewis, Nearest neighbor machine translation, in: 9th International Conference on Learning Representations, ICLR, 2021.
[37] X. Zheng, Z. Zhang, J. Guo, S. Huang, B. Chen, W. Luo, J. Chen, Adaptive nearest neighbor machine translation, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 368–374. https://doi.org/10.18653/v1/2021.acl-short.47.
[38] H. Lin, L. Yao, B. Yang, D. Liu, H. Zhang, W. Luo, D. Huang, J. Su, Towards user-driven neural machine translation, in Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, Association for Computational Linguistics, 2021, pp. 4008–4018. https://doi.org/10.18653/v1/2021.acl-long.310.
[39] N. Kassner, H. Schütze, BERT-kNN: Adding a kNN Search Component to Pretrained Language Models for Better QA, Find. Assoc. Comput. Linguist.: EMNLP 2020 Assoc. Comput. Linguist. (2020) 3424–3430, https://doi.org/10.18653/v1/2020.findings-emnlp.307.
[40] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. Bowman, GLUE: A multi-task benchmark and analysis platform for natural language understanding, in: Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP @ EMNLP, Association for Computational Linguistics, 2018, pp. 353–355. https://doi.org/10.18653/v1/W18–5446.
[41] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, T. Liu, Do transformers really perform badly for graph representation? Adv. Neural Inf. Process. Syst., NeurIPS (2021) 28877–28888.
[42] E. Jang, S. Gu, B. Poole, Categorical reparameterization with Gumbel-Softmax, in: 5th International Conference on Learning Representations, ICLR, 2017.
[43] K. Kowsari, D.E. Brown, M. Heidarysafa, K.J. Meimandi, M.S. Gerber, L.E. Barnes, HDLTex: Hierarchical deep learning for text classification, in: 2017 16th IEEE International Conference on Machine Learning and Applications, ICMLA, 2017, pp. 364–371. https://doi.org/10.1109/ICMLA.2017.0–134.
[44] D.D. Lewis, Y. Yang, T.R. Rose, F. Li, RCV1: A new benchmark collection for text categorization research, J. Mach. Learn. Res. 5 (2004) 361–397.
[45] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, A survey on text classification: from traditional to deep learning, ACM Trans. Intell. Syst. Technol. 13 (2020) 1–41, https://doi.org/10.1145/3495162.
[46] G. Tsoumakas, I. Katakis, I. Vlahavas, Mining Multi-label Data, in: O. Maimon, L. Rokach (Eds.), Data Mining and Knowledge Discovery Handbook, Springer US, Boston, MA, 2010, pp. 667–685, https://doi.org/10.1007/978-0-387-09823-4_34.
[47] S. Kiritchenko, S. Matwin, R. Nock, A.F. Famili, Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization, in: L. Lamontagne, M. Marchand (Eds.), Advances in Artificial Intelligence, 19th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2006, Québec City, Québec, Canada, June 7–9, 2006, Proceedings, Springer, 2006: pp. 395–406. https://doi.org/10.1007/11766247_34.
[48] Z. Deng, H. Peng, D. He, J. Li, P. Yu, HTCInfoMax: A global model for hierarchical text classification via information maximization, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL, Association for Computational Linguistics, 2021, pp. 3259–3265. https://doi.org/10.18653/v1/2021.naacl-main.260.

Jun Zhang gained his Ph.D. degree from computer school of Wuhan University, Wuhan, China, 2018. He is currently a professor in school of information engineering at East China University of Technology, Nanchang, China. His main research interests include natural language processing, computer architecture, high performance computing.

Yubin Li is currently a postgraduate in school of information engineering at East China University of Technology, Nanchang, China. Her main research interests include natural language processing.

Fanfan Shen gained his Ph.D. degree from computer school of Wuhan University, Wuhan, China, 2017. He is currently a vice professor in school of information engineering, Nanjing Audit University, Nanjing, China. His main research interests include computer architecture and non- volatile memory.

Yueshun He gained his Ph.D. degree from college of computer science and technology at Nanjing University of Aeronautics and Astronautics, Nanjing, China, 2010. He is currently a professor in school of information engineering at East China University of Technology, Nanchang, China. His main research interests include wireless sensor networks, internet of things, and artificial intelligence.

Hai Tan gained his Ph.D. degree from school of science and Technology at Beijing Institute of Technology, China, in 2016. He is currently a professor in school of information engineering, Nanjing Audit University, Nanjing, China. His main research interests include many-core architecture and big data.

Yanxiang He is currently a CCF Fellow, a Ph.D. supervisor, and a professor at Computer School, Wuhan University, Wuhan, China. His main research interests include trustworthy software engineering, natural language processing, performance optimization of multi-core processor, and distribution computing.