Journal of King Saud University – Computer and Information Sciences 35 (2023) 161–171

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

GUDN: A novel guide network with label reinforcement strategy for extreme multi-label text classification☆,☆☆

Qing Wang, Jia Zhu *, Hongji Shu, Kwame Omono Asamoah, Jianyang Shi, Cong Zhou

Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua, China

ARTICLE INFO

Article history:
Received 19 November 2022
Revised 16 March 2023
Accepted 17 March 2023
Available online 24 March 2023

Keywords:
Extreme multi-label
Neural networks
Text classification
Label semantic
Long text

ABSTRACT

Extreme multi-label text classification (XMTC) is an emerging and essential task in natural language processing. Its objective is to retrieve the most relevant labels for a text from a large set of labels while balancing time and accuracy. Although large-scale pre-trained models have brought new perspectives to this task, more attention should be given to valuable fine-tuned methods and the significant semantic gap between texts and labels. In this paper, we propose a novel guide network (GUDN) with a label reinforcement strategy based on label semantics to help fine-tune pre-trained models for classification. Experimental results demonstrate that GUDN outperforms state-of-the-art methods on Eurlex-4k and achieves competitive results on other popular datasets. In addition, we find that meaningless tokens can harm the Transformer-based model's classification accuracy in another experiment. We conclude that GUDN is effective in the presence of solid semantics. Our source code is available at https://t.hk.uy/aFSH.

1. Introduction

The goal of extreme multi-label text classification (XMTC) is to accurately recall some of the most relevant labels for a given text from an extremely large label set within a reasonable amount of time. This problem is ubiquitous, from e-commerce platforms to term search sites. Fig. 1 depicts an article from Wikipedia introducing artificial intelligence, which contains many labels related to the topic. For example, "Google", "automated decision-making", "machines", and other interdisciplinary concepts. Typically, on Wikipedia, the number of these labels or terms can reach hundreds of thousands. Clicking on these labels provides access to related articles. As the number of labels continues to increase, the XMTC method is needed to match articles and labels effectively. The core challenge of XMTC is how to match these labels to texts accurately and efficiently.

It has been emphasized that extreme multi-label text classification (XMTC) differs from multi-class or multi-label text classification. In XMTC tasks, the number of labels can reach hundreds of thousands or even more, which impairs predictive accuracy and increases computation time. The ample but sparse label space has made the "long tail" distribution apparent, leading to poor accuracy in some samples. In this situation, the length of the texts is also long, requiring a lot of memory and time for training. XMTC has attracted much research interest in the last decade due to its wide downstream applications, such as advertising, user profiles, and web search.

Many traditional machine learning methods, including FastXML (Prabhu and Varma, 2014), Bloom Filters (Cisse et al., 2013), Karimi Jafarbigloo and Danyali, 2021 and Dismec (Babbar and Shoelkopf, 2016), have been proposed to solve the XMTC problem, with relatively impressive results in some ways. However, many of these methods often rely upon some specific setting with low efficiency. For instance, Bloom Filters is only feasible when the label matrix is low rank. Moreover, they often use sparse features like the TF-IDF

* Corresponding author at: Zhejiang Normal University, No. 688 Yingbin Avenue, Jinhua City 321004, China.
E-mail addresses: wq2481@zjnu.edu.cn (Q. Wang), jiazhu@zjnu.edu.cn, zhujia@stanford.edu (J. Zhu), shj451148969@zjnu.edu.cn (H. Shu), koasamoah2014@gmail.com (K.O. Asamoah), shijianyang@zjnu.edu.cn (J. Shi), zhoucong@zjnu.edu.cn (C. Zhou).

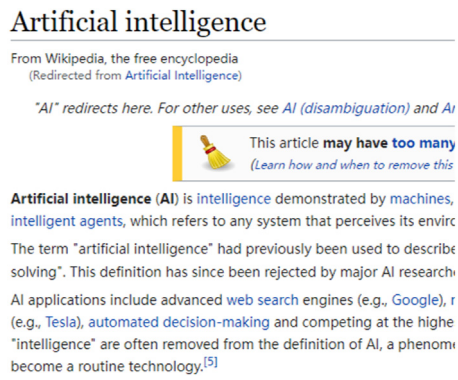
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

☆ This document is the results of the research project funded by the National Science Foundation.

☆☆ This note has no numbers. In this work we demonstrate a_b the formation Y_1 of a new type of polariton on the interface between a cuprous oxide slab and a polystyrene micro-sphere placed on the slab.

https://doi.org/10.1016/j.jksuci.2023.03.009
1319-1578/


Artificial intelligence

From Wikipedia, the free encyclopedia
(Redirected from Artificial Intelligence)

"AI" redirects here. For other uses, see AI (disambiguation) and Ar

This article may have too many
(Learn how and when to remove this

Artificial intelligence (**AI**) is intelligence demonstrated by machines, intelligent agents, which refers to any system that perceives its envirc

The term "artificial intelligence" had previously been used to describe solving". This definition has since been rejected by major AI research

AI applications include advanced web search engines (e.g., Google), r (e.g., Tesla), automated decision-making and competing at the highe "intelligence" are often removed from the definition of AI, a phenom become a routine technology.[5]

Fig. 1. An example of XMTC extracted from https://en.wikipedia.org/wiki/Artificial_intelligence.

and Bag of Words (BOW) as the input, lacking semantic information, which is hard to optimize. They need to break through the limitations of traditional methods to achieve high prediction accuracy.

Deep learning methods have flourished in XMTC tasks in recent years. One example of a representative method is C2AE (Yeh et al., 2017), which uses a sparse linear network to explore the latent space between texts and labels. RankAE (Wang et al., 2019) further improved C2AE and spread it to extreme multi-label text classification. Some recent works, such as DECAF (Mittal et al., 2021a), indicate that label metadata such as label structure and description are helpful for XMTC. Effectively extracting descriptive label semantic features can provide an immediate performance boost. Nevertheless, C2AE and RankAE ignore the label metadata, using the multi-hot label representation vector to explore the latent space between text and label. Additionally, more than just using sparse linear layers to find latent space is required.

DXML (Zhang et al., 2018) used the label structure to build a graph and reveal the potential relationship between texts and labels. LAHA (Huang et al., 2019) further introduced the attention mechanism for label embedding. Even though DXML and LAHA consider the metadata of the labels, the graph structure hides the original features of the labels. Moreover, DXML and LAHA did not address the semantic gap between text and label. From the perspective of natural language, there must be some connection between the semantics of text and label. However, these connections have not been precisely exploited, resulting in unaddressed issues. Furthermore, more attention is needed to the substantial semantic gap between texts and labels.

Recent work has adopted large-scale pre-trained models as the backbone network and tamed them for high predictive accuracy. X-Transformer (Chang et al., 2019), which used pre-trained models such as BERT (Devlin et al., 2018), effectively extracted features from raw texts to significantly improve accuracy. However, X-Transformer could be more efficient when it comes to computational cost. LightXML (Jiang et al., 2021) improved upon X-Transformer, making it lighter and faster and achieving state-of-the-art results. However, LightXML and X-Transformer can be further improved with a valid fine-tuned strategy. Furthermore, they need to consider the critical label semantics that is easy to find.

It is crucial to explore the importance of label semantics to find the latent space. The valuable fine-tuned methods need to be considered for the Transformer-based model. To address this, we designed a feature extractor that uses an improved Transformer-based model as the backbone network for the text features. The Transformer-based model has demonstrated the ability to extract word-level and sentence-level features, making it a suitable candidate for feature extraction. Additionally, we propose a novel guide network (GUDN) that works in tandem with the feature extractor to improve the performance of XMTC tasks. The GUDN provides a straightforward guide (fine-tuning) for the text feature, reducing irrelevant information. Given the vast semantic gap between text and label, which increases prediction difficulty, we developed a label reinforcement strategy to assist in this task.

The main contributions of our work are summarized as follows:

- A novel guide network (GUDN) that includes two modules of guide and two loss functions is proposed. GUDN helps fine-tune the Transformer-based model to capture the label-aware features to improve performance.
- GUDN, from the perspective of label semantics, considers raw label semantics with a practical and succinct label reinforcement strategy and combines a refined deep pre-trained model to extract features for predictive accuracy.

The remainder of the paper is structured as follows: Section 2 reviews related works in the field. Section 3 provides a detailed description of the proposed methods, including the Transformer-based feature extractor, the novel guide network (GUDN), and the label reinforcement strategy. We present the experimental results in Section 4. Finally, we summarize the paper and suggest future work in Section 5.

2. Related work

Over the years, numerous approaches have been proposed to tackle the XMTC problem, which can be broadly categorized into two groups. The first group includes traditional methods, which can be further classified into three categories: embedding-based, tree-based, and one-vs-all (OVA) strategies. The second group comprises deep learning methods, which have recently gained popularity. In the following section, we provide a brief overview of these methods.

Embedding-based methods: Embedding-based methods aim to reduce label redundancy under low-rank assumptions, thereby alleviating storage and computing overhead. This behavior can also be thought of as coding, making the training process a natural codec process. The low-dimension theory for label embedding was first proposed by Hsu et al. (2009). To capture label correlations non-linearly and reduce the number of required labels, SLEEC (Bhatia et al., 2015) clusters the data to speed up the training stage. Building on SLEEC, (Xu et al., 2016) improved the optimization strategy and achieved good results. AnnexML (Tagami, 2017) also advanced SLEEC by addressing unreasonable data partitioning, indirect objective function, and slow prediction speed. However, overfitting has been identified as the primary reason for the poor performance of embedding-based methods (Guo et al., 2019). In response, GLaS was developed to reduce overfitting. However, the label information is inevitably lost due to the process of embedding and de-embedding. Moreover, the "long tail" of the label distribution tends to challenge the low-rank assumption.

Tree-based methods: Tree-based methods offer a hierarchical approach for label set division by creating a label tree. When the tree is balanced, the prediction time becomes sub-linear and even logarithmic. FastXML is a typical instance of a tree-based method that optimizes the ranking loss function. SwiftXML (Prabhu et al., 2018a) uses label features and adopts a warm-start strategy for improved performance. Hierarchical softmax approaches were introduced by Wydmuch et al. (2018) to reduce training time, while Parabel (Prabhu et al., 2018b) improved the probabilistic label tree (PLTs) (Liu et al., 2013). CRAFTML (Siblini et al., 2018) quickly divides the label tree using a modified random forest algorithm. While the tree's structure reduces prediction time, accuracy may be affected as the tree becomes deeper.

OVA methods: OVA methods create a binary classifier for each label, such as PPDSparse (Yen et al., 2017), Slice (Jain et al., 2019), Dismec, and Bonsai (Khandagale et al., 2020), leading to a significant improvement in prediction accuracy. PPDSparse uses a novel loss function to expand training, while Slice employs negative sampling to address XMTC challenges effectively. Dismec's layer parallelization strategy and Bonsai's label tree structure overcome the limitations of OVA methods by reducing the model size and computational complexity to some extent. However, these techniques may still be unsuitable for real-world applications due to their resource requirements.

Deep learning methods: In consideration of OVA methods requiring a lot of computational resources, embedding-based methods relying on low-rank label assumptions and tree-based methods leading to reduced accuracy and large model sizes highlight the need for an advanced methodology to address these challenges. Deep learning (DL) methods have recently flourished in XMTC tasks. Since introducing the first DL method, XML-CNN, researchers have proposed many DL-based solutions. The current trend suggests that DL is gradually dominating this field, and our proposed method is based on DL.

To reduce computation time, APLC-XLNet introduced a new approach for label partitioning called probabilistic label clusters. At the same time, Niculescu-Mizil and Abbasnejad (2017) presented a label filter to accelerate label prediction. Jasinska et al. (2016) raised sparse probability estimates to reduce calculation costs, and Jain et al. (2016a) improved the loss function to make it more reasonable. Additionally, Babbar and Schölkopf (2019) focused on addressing the issue of the "long tail" distribution, hoping to alleviate this phenomenon and increase accuracy.

C2AE first proposed the hypothesis of a latent space between the label and text and conducted a preliminary exploration. At the same time, RankAE suggested a margin-based ranking loss and dual-attention mechanism establish a common latent space between text and label features. DXML also aimed to find the latent space and integrated label structure information and metadata to connect text and label. Lastly, LAHA employed the attention mechanism with a label co-exist graph to integrate label and text semantics.

However, the semantic information of labels is significantly reduced in LAHA and DXML due to the graph structure that hides the semantics. RankAE requires a more robust feature extractor for labels rather than a sparse linear neural network. Recent studies (Mittal et al., 2021a; Mittal et al., 2021b) have emphasized the importance of label metadata, such as label structure or label text description. In our approach, we fully consider the label semantics and use a deep, pre-trained model to extract features from the raw labels directly.

Inspired by the success of deep pre-training models in natural language processing, X-Transformer utilizes pre-trained Transformer-based models to handle XMTC tasks. However, considering the X-Transformer's computational complexity and model size, LightXML was proposed to obtain a lightweight and faster model. While LightXML has reached an advanced level, both X-Transformer and LightXML still need to explore a better approach for fine-tuning, as they rely only on the final objective function, which remains challenging for extreme classification. To address these issues, we propose a novel guide network to further guide the Transformer-based model in extracting label semantics.

3. Proposed method

This section presents a detailed description of the proposed method, GUDN, an end-to-end and easily extensible model consisting of three parts: the feature extractor, guide network, and ranking classifier. Before the extraction process, the label reinforcement strategy is applied to the label inputs to improve the semantic information. During the extraction process, the feature extractor first extracts the features of both texts and labels, which are then fed into the guide network to establish a close relationship between them. The feedback from this relationship is used to optimize the feature extractor continuously. Finally, the ranking classifier leverages the accurate semantic information obtained from the guide network to perform the classification. Fig. 2 illustrates the proposed framework, and the upper part shows the training stage of GUDN.

3.1. Preliminaries

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ represent the training dataset with n samples where $x_i \in R^d$ is the input of raw text, $y_i \in \{0, 1\}^L$ denotes multi-hot vector of true label. The real semantic labels which belong to a sample text are also a part of the input during the training stage. Note that each raw text length is equal to the d, and the sum of the number of labels is L. We want to find a function f to map x_i and y_i. If the $y_{ij} = 1$ then the function f will output a high score, wherein $j \in [1, L]$. The mapping function f can be expressed as follows:

$$f(x_i, k) = W_k B(x_i), \tag{1}$$

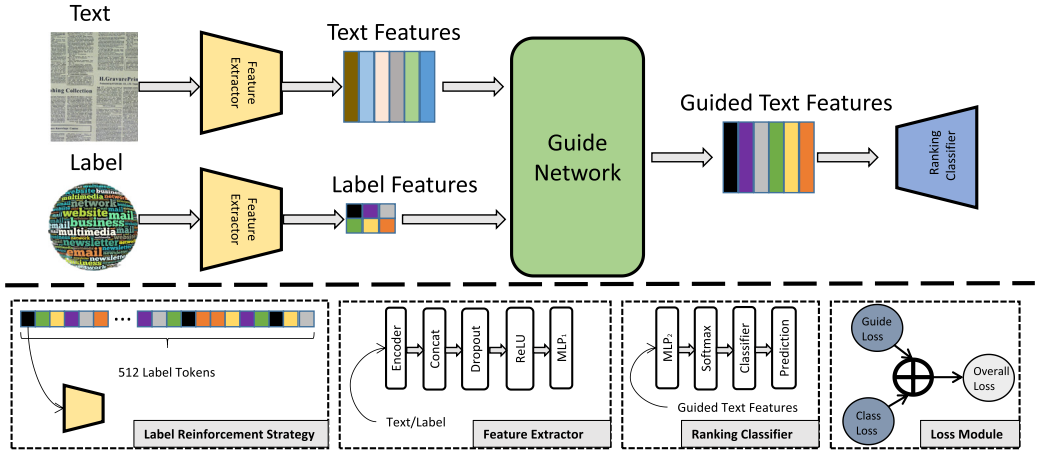


Fig. 2. GUDN: A novel guide network with label reinforcement for extreme multi-label text classification. The lower part of the figure includes label reinforcement strategy, feature extractor, ranking classifier, and loss module.

where $B(x_i)$ represents the i-th text features generated from an encoder B and W is the classifier, usually a fully connected layer. f output the score of k-th label. If the score is high, the label is likely to belong to the text.

3.2. Feature extractor

GUDN builds on the success of X-Transformer and LightXML by leveraging a well-designed BERT to extract basic features, which has shown impressive performance in various natural language processing tasks. However, previous methods only utilize the label multi-hot vector in a sparse linear network, lacking sufficient semantic information to capture the latent space between labels and texts. GUDN incorporates raw label semantic information in its feature extractor to address this issue.

As illustrated in Fig. 2, the feature extractor consists of an Encoder layer (BERT), a Concatenation layer, a Dropout layer, a ReLU layer, and an MLP layer. Specifically, GUDN employs an adapted BERT with 12 layers and 768 hidden dimensions to extract the original text features, while the labels share the same BERT with the texts to obtain their respective features. Sharing one BERT can significantly reduce the model's size and complexity, accelerating convergence. Although text and label features are extracted asynchronously during the training phase, they are used together to calculate the loss.

To prevent overfitting, GUDN incorporates a dropout layer with high dropout rates. Since label descriptions typically have less semantic information than text, we concatenate the output of the last eight layers of the "[CLS]" token to represent the extracted text features. Additionally, we add two (an empirical parameter) extra layers for label features to enhance their semantic information. After the dropout layer, a ReLU activation function and an MLP layer refine the features. Thus, the feature extractor F can be formulated as follows:

$$E = W_e\sigma(D(f)) + b_e, \tag{2}$$

where f represents the splicing features from the encoder, D is the dropout layer, σ is the ReLU. W_e and b_e are the parameters of MLP_1. The outputs of the feature extractor are text features E_t, and the label features E_l.

3.3. Guide network

Relying solely on a simple classification network to link texts to labels can be likened to being lost at sea without a guide - it is unstable and uncertain. A simple and effective solution is to create a guide mechanism for labels and texts to address this issue. This concept is inspired by contrastive learning methods used in CLIP (Radford et al., 2021), and MICoL (Zhang et al., 2022b) to handle text. In GUDN, texts and labels are treated like visual and textual inputs.

Previous works such as C2AE and RankAE have attempted to establish a bridge between texts and labels by finding a latent space with a sparse linear network trained to guide classification. However, this approach is only practical with solid label semantics. To achieve reasonable label representations, the guide network must be disciplined enough to incorporate raw label semantics. DXML and LAHA construct a label graph structure for prediction. However, this approach ignores label metadata and partially conceals the original label semantics. Consequently, we propose the guide network as a solution to these issues.

Feature extractor is designed to extract semantic-grained features, while the guide network provides a mapping mechanism between texts and labels. As illustrated in Fig. 3, the guide network has a simple structure. However, its effectiveness depends on two crucial components.

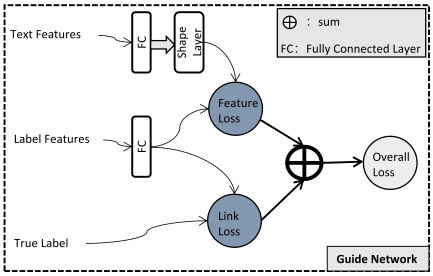


Fig. 3. The Guide Network.

The first guide is designed to instruct BERT to learn the most representative label features from text features, allowing the network to discover an effective latent space between the text and label semantics. Specifically, the text and label features obtained from the feature extractor are passed through a fully connected (FC) layer. However, before they are concatenated, the text features need to pass through a shape layer to match the shape of the label features. The other guide is responsible for establishing a direct mapping relationship between label features and true labels, which helps alleviate the pressure on the ranking classifier. For simplicity, we use E_t and E_l to represent the text and label features, respectively.

The guide network ultimately enables us to link texts and labels. We utilize two loss functions, namely $L_{feature}$ and L_{link}, which serve as solid bridges in the guide network. The former allows text space and label space to blend, while the latter connects labels and label features. The loss functions can be expressed as follows:

$$L_{feature}(E_t, E_l) = \frac{1}{2n}\sum_{i=1}^{n}\|E_{ti} - E_{li}\|^2, \tag{3}$$

$$L_{link}(y, \hat{y}) = \sum_{i=1}^{n}\sum_{j=1}^{L} - y_{ij}\log(\hat{y_{ij}}) - (1 - y_{ij})\log(1 - \hat{y_{ij}}). \tag{4}$$

Eq. (3) is the mean square error loss (MSE) calculated from label features E_t and text features E_l. Eq. (4) is the binary cross-entropy loss (BCE) calculated from true label y and predicted label $\hat{y}$. $\hat{y}$ is not produced from text features but label features. The sum loss of the guide network L_{guide} is the sum of $L_{feature}$ and L_{link}, which is described as follows:

$$L_{guide} = L_{feature} + L_{link}. \tag{5}$$

Theoretically, minimizing L_{guide} can make the feature extractor and ranking classifier independent of the guide network, which means that label information is not needed during the test stage. This is because the feature extractor and ranking classifier have learned to find a path from text to the correct labels alone, with the guidance of the guide network during training.

The guide network presented in this paper is not limited to the classification problems described in this work. It can be applied to more general cases, including multi-label and multi-category classification. Moreover, the network's structure is straightforward, making it easy to extend and adapt to various scenarios.

3.4. Ranking classifier

The ranking classifier, which is illustrated in the lower part of Fig. 2, consists of an MLP layer, a Softmax layer, and the classifier. Its main task is to rank the candidate labels and output the final classification result. The formula for the ranking classifier can be expressed as follows:

$$y' = W_c\theta(W_1(E_t) + b_1), \tag{6}$$

where W_1 and b_1 are the parameters of MLP_2, θ represents the softmax layer, and W_c denotes the classifier. For medium-size datasets Eurlex-4K, AmazonCat-13K, and Wiki10-31K, we do not change the original output space of the ranking classifier. However, for the large-scale dataset Wiki-500K, we go after LightXML to adopt a dynamic negative sampling strategy. The k label clusters with the highest recall probability are selected from the output space. Then the candidate labels are selected from clusters.

The proposed dynamic negative sampling strategy in the large-scale dataset Wiki-500K selects the k label clusters with the highest recall probability from the output space. The candidate labels are then chosen from these clusters, resulting in a final candidate set that contains all positive and many "hard negative" samples. This effective strategy not only compresses the output space but also enhances accuracy. Although label clustering is generally required according to bag-of-words (BOW) before dynamic negative sampling, X-Transformer provides new insights, such as positive instance feature aggregation. Nevertheless, in this work, we opted to use BOW for simplicity. The binary cross-entropy (BCE) loss is used as the classification loss and can be expressed as follows:

$$L_{class}(y, y') = \sum_{i=1}^{n}\sum_{j=1}^{L} -y_{ij}\log(y'_{ij}) - (1-y_{ij})\log(1-y'_{ij}), \tag{7}$$

where y_i is the ground truth, and y'_i is the labels predicted by text information. They are both L-dimensional multi-hot vectors.

3.5. Label reinforcement strategy

We observe a significant disparity in both length and semantics between long text and short label sequences in extreme text classification scenarios. While the number of labels may exceed several thousand, the entire label sequences per sample are very short, resulting in the addition of meaningless tokens like "Padding" to the end of the label input. On the other hand, long text can have tens of thousands of characters, making it challenging to match.

To address this issue, we propose an enhanced label input approach for GUDN. We have devised two methods for label combination to improve our approach significantly. While (Zhang et al., 2022a) may have similar strategies, their aims differ. Zhang et al. (2022a) presents a method for adding keywords to the input to improve the label description. However, the computing cost for keyword extraction is very high. Our approach is a simple yet effective method for enriching label semantics. We employ two label reinforcement standards: an ordered label-fill method and a disordered label-fill method.

For the ordered label fill method, we copy the label sequence of a specific sample and add it to the end of the original sequence. We repeat this operation until the sequence length reaches 512. For the disordered label fill method, we use a random sampling method. For each label set related to a sample, we shuffle the position of each label and add the set to the end of the original sequence. This random sampling process follows a normal distribution, resulting in the sequence being composed of disordered tokens.

3.6. Training process

GUDN is finally implemented since we have constructed the feature extractor, guide network, and ranking classifier. The objective function $L_{overall}$, which contains two losses, L_{guide} and L_{class}, is minimized by GUDN. The sum of two losses caused by the guide network is L_{guide}, and the classification loss is L_{class}. The overall loss function is given in Eq. (8).

$$L_{overall} = L_{guide} + L_{class}. \tag{8}$$

We incorporate the two losses generated by the guide network with the classification loss as they are all indispensable to GUDN. The interaction between the losses is key to achieving optimal prediction accuracy. Although challenging to optimize, GUDN's simplicity enables convergence with all three losses.

During the training stage, we utilize label semantics with a label reinforcement strategy. Initially, we extract the most basic and crucial semantic information from text and label features using the feature extractor. Subsequently, we feed the text and label features into the guide network, which employs the feature loss as a guide to train both the feature extractor and the ranking classifier. After the guidance process, only the feature extractor and the ranking classifier remain, resulting in a lighter model that is well-suited for time-sensitive user applications. Ultimately, GUDN provides fast and accurate predicted results.

4. Experiment

We perform experiments on Linux (Ubuntu 20.04.1). The experiments use four Nvidia GeForce RTX 3090 GPUs with Intel(R) Xeon (R) Gold 6254 CPU @ 3.10 GHz to do calculations in parallel. Every GPU memory is 24 GB, but the training phase occupies less than 20 GB. We repeated every experiment three times with different globally random numbers for the whole training stage. The results in this paper are the average of these three times experiments.

4.1. Datasets and evaluation metrics

The datasets for the experiments are collected from http://manikvarma.org/downloads/XC/XMLRepository.html (Bhatia et al., 2016). Eurlex-4K, AmazonCat-13K, Wiki10-31K, and Wiki-500K were the four representative datasets. Eurlex-4K is text data about European Union law, containing nearly four thousand labels formed according to EU-ROVOC descriptors. Amazon-13K is a product-to-product recommendation dataset, and labels are the product categories in this dataset. Wiki10-31K and Wiki-500K are excerpts from Wikipedia articles containing about thirty-one thousand and five hundred thousand labels, respectively. Table 1 can provide details about the four datasets. It is worth noting that the texts in these datasets are exceedingly long. Generally, their length increases as the label number grows. For example, the article in Wiki-500K is longer than it is in Eurlex-4K. Merrillees and Du (2021) proposed a parting strategy making new distribution of data samples, which is promising for tests.

We use three extensively used metrics in XMTC tasks for comparison. One of these is a simple but intuitive evaluation metric named precision performance at the top ($P@k$). The calculation formula for $P@k$ is as follows:

$$P@k = \frac{1}{k}\sum_{i\in \mathrm{rank}_k(\hat{y})} y_i, \tag{9}$$

where k is constant and it is usually $1, 3$ or 5. We rank the predicted results, $\hat{y}$, by probability, then select the top k with the highest probability and record their index number. The $P@k$ score is higher if the k indexes have more 1 values corresponding to the label vector position.

Another metric is the normalized discounted cumulative gain $nDCG@k$, which is defined as follows:

$$DCG@k = \sum_{i\in \mathrm{rank}_k(\hat{y})} \frac{y_i}{log(i+1)}, \tag{10}$$

Table 1
A specific description of datasets. The training and testing set numbers are denoted by *TRN* and *TST*, respectively. *LBL* refers to the number of labels. *SPL* represents the average sample per label, and *LPS* represents the average label per sample.

Datasets	*TRN*	*TST*	*LBL*	*SPL*	*LPS*
Eurlex-4K	15539	3809	3993	25.73	5.31
AmazonCat-13K	1186239	306782	13330	448.57	5.04
Wiki10-31K	14146	6616	30938	8.52	18.64
Wiki-500K	1813391	783743	501070	23.62	4.89

$$iDCG@k = \sum_{i=1}^{min(k,||y||_0)} \frac{1}{log(i+1)}, \tag{11}$$

$$nDCG@k = \frac{DCG@k}{iDCG@k}. \tag{12}$$

Except for the two metrics mentioned before, the third metric is the propensity-scored performance at the top (*PSP@k*) (Jain et al., 2016b). This metric allows for avoiding the "head label strength." Typically, in the setting of the XMTC, the number of the "tail label" can run high. However, the metrics of *P@k* and *nDCG@k* always ignore this phenomenon. So bringing *PSP@k* to evaluate the model seems more objective. The *PSP@k* can be defined as follows:

$$PSP@k = \frac{1}{k} \sum_{i \in \text{rank}_k(\hat{y})} \frac{y_i}{p_i}, \tag{13}$$

where p_i is the propensity score for a certain label.

4.2. Experiments setting

We limited the length of the input texts to 512 to comply with BERT's limitation. When the labels are short and cannot reach 512, we treat them as a whole and feed the entire label sequence to the feature extractor. However, the label reinforcement strategy can increase the label sequence to 512. If the text is longer than 512, we select the head, tail, or middle section to keep, which can result in some information loss, but we only use the first 512 words.

Regarding the training epochs, we set 40 epochs for Eurlex-4K and Wiki10-31K and 20 epochs for the datasets Wiki-500K and AmazonCat-13K due to their large sample numbers. For all the datasets, the training batch size is 8, and the testing batch size is 16.

4.3. Experiments and discussion

We conduct a total of three experiments. The first experiment tests the accuracy of GUDN for comparison. The second experiment is to prove the efficiency of every part of GUDN. The last experiment is used to investigate label reinforcement strategy and how it influences classification accuracy.

The experimental results of *P@k*, *nDCG* and *PSP@k* are shown in Table 2–4 respectively. We experiment with four datasets and compare the results of GUDN with seven representative approaches. The data of *P@k* of these models were obtained from their original papers. As far as possible, we refer to the published experimental data for the results of *nDCG@k* and *PSP@k* of AttentionXML (You et al., 2018) and LightXML, though it is incomplete.

Comparing with representative methods: We used the reproduced model test to obtain results for the data we could not collect. Among the most representative models, DXML and RankAE were similar to C2AE, as they aimed to find a latent space between texts and labels, which is also one of the aims of GUDN. Although their achievements have been surpassed, their ideas remain inspiring. In comparison, GUDN uses a deep, pre-trained Transformer-based model to directly extract the raw label semantics, which is more conducive to finding the latent space.

To our knowledge, XML-CNN was the first method to use a deep network for the XMTC task. The results of AttentionXML showed that the accuracy was significantly improved compared to XML-CNN. KTXMLC (Prajapati and Thakkar, 2022) is a tree-based method with powerful performance. The main competitors of GUDN are X-Transformer and LightXML, both based on pre-trained models to encode the text and used to be state-of-the-art. Like X-Transformer and LightXML, GUDN also uses BERT as the backbone network. With feature loss as the guideline and under the guidance of the guide network with label reinforcement strategy, GUDN also achieves significant results.

After training, all three losses decreased considerably. Among the three losses, the feature loss decreased the most, followed by the class loss, and the link loss was the least obvious. The ablation study also shows how significant the three losses were when learned separately.

One of the core challenges of XMTC is accuracy. Table 2 shows that GUDN achieves state-of-the-art performance on Eurlex-4K, especially for *P@5*. GUDN also has some advantages on AmazonCat-13K, Wiki-500K, and Wiki10-31K. However, it is

Table 2
Using *P@k*, we compared the results of experiments with several representative methods on Eurlex-4K, AmazonCat-13K, Wiki10-31K, and Wiki-500K. The font in bold indicates the best score, and the underlined font indicates the sub-best score. The 'Difference' represents the difference between the experimental results of GUDN and the results of using state-of-the-art methods.

Datasets	P@k	XML-CNN (Liu et al., 2017)	DXML (Zhang et al., 2018)	AttentionXML (You et al., 2018)	RankAE (Wang et al., 2019)	X-Transformer (Chang et al., 2019)	LightXML (Jiang et al., 2021)	KTXMLC (Prajapati and Thakkar, 2022)	GUDN	Difference
Eurlex-4K	P@1	75.32	-	87.12	79.52	87.22	87.63	82.02	**88.13**	+0.50
	P@3	60.14	-	73.99	65.14	75.12	75.89	69.11	**77.06**	+1.17
	P@5	49.21	-	61.92	53.18	62.90	63.36	57.85	**65.49**	+2.13
AmazonCat-13K	P@1	93.26	-	95.92	-	96.70	**96.77**	93.97	96.71	-0.06
	P@3	77.06	-	82.41	-	83.85	84.02	79.85	**84.19**	+0.17
	P@5	61.40	-	67.31	-	68.58	**68.70**	65.09	67.96	-0.74
Wiki10-31K	P@1	81.41	86.45	87.47	83.60	88.51	89.45	85.6	**89.75**	+0.30
	P@3	66.23	70.88	78.48	72.07	78.71	**78.96**	73.45	78.58	-0.38
	P@5	56.11	61.31	69.37	62.07	69.62	69.85	64.34	**69.86**	+0.01
Wiki-500K	P@1	-	-	76.95	-	77.28	77.78	-	**77.89**	+0.11
	P@3	-	-	58.42	-	57.47	58.85	-	**59.15**	+0.30
	P@5	-	-	**46.14**	-	45.31	45.57	-	46.01	-0.13

Table 3
The experimental results of *nDCG@k* on Eurlex-4K, AmazonCat-13K, Wiki10-31K, and Wiki-500K compared GUDN with two other significant models. Since the *nDCG*@1 is equal to the *P*@1, we do not list them. The font in bold indicates the best score.

	EURLex-4K		AmazonCat-13K		Wiki10-31K		Wiki-500K	
	nDCG@3	nDCG@5	nDCG@3	nDCG@5	nDCG@3	nDCG@5	nDCG@3	nDCG@5
AttentionXML(You et al., 2018)	77.44	71.53	91.17	89.48	80.61	73.79	**76.56**	**74.86**
LightXML (Jiang et al., 2021)	78.00	71.87	91.77	**90.58**	**81.81**	**74.67**	74.71	72.19
GUDN	**78.19**	**72.75**	**91.98**	90.06	81.38	74.63	75.16	73.78

Table 4
The experimental results of *PSP@k* on Eurlex-4K, AmazonCat-13K, Wiki10-31K, and Wiki-500K compared GUDN with two other significant models. The font in bold indicates the best score.

	PSP@k	AttentionXML(You et al., 2018)	LightXML(Jiang et al., 2021)	GUDN
EURLex-4K	PSP@1	42.31	42.18	**43.89**
	PSP@3	49.17	48.97	**50.61**
	PSP@5	52.19	53.99	**54.98**
AmazonCat-13K	PSP@1	53.76	**54.88**	54.69
	PSP@3	68.72	70.21	**70.98**
	PSP@5	76.38	76.54	**77.04**
Wiki10-31K	PSP@1	15.57	**16.00**	15.99
	PSP@3	16.80	**16.99**	16.87
	PSP@5	17.82	**18.97**	18.16
Wiki-500K	PSP@1	**34.00**	31.99	32.71
	PSP@3	**44.32**	42.00	41.91
	PSP@5	**50.15**	46.53	46.53

noticeable that GUDN does not perform as well on AmazonCat-13K, Wiki10-31K, and Wiki-500K as it does on Eurlex-4K. Due to the diverse applications of the industry, XMTC methods need to consider training and prediction efficiency. The GUDN's simple architecture still allows it to remain competitive regarding model size and calculation time. Due to hardware limitations, we could not reproduce some models, so we could not obtain the training time and memory usage of these models for comparison. However, we provide the training time and model size of GUDN. Please refer to Table 5 for specific experimental data.

Performance of label reinforcement strategy: We conducted experiments on the label reinforcement strategy using GUDN as the fundamental model. The results of the two standards of label reinforcement are compared in Fig. 5. These improvements are based on the best performance of GUDN.

The label reinforcement strategy improved the matching performance by 0.03% to 0.3% compared to the basic GUDN, whether ordered or disordered. Moreover, under the disordered scenario, the improvements were more pronounced because the label sequence does not have any order in the real world. On the other hand, the Transformer-based model considers the input to have a contextual relation. Thus, with a disordered label sequence, we can let GUDN learn more features about the label and its semantics.

Detail Analysis: In this section, we present an in-depth analysis of the experimental data to explore why GUDN performs less effectively on datasets other than Eurlex-4K. Our research indicates that the label settings in Eurlex-4K are more consistent with the semantics of the natural language. These labels possess robust semantic features and accurately represent their corresponding

Table 5
This table shows the model size of GUDN in GB and the time required for each epoch of training under the current experimental setting in minutes.

Dataset	Model Size	Training Time
Eurlex-4K	2.66	1.50
AmazonCat-13K	2.71	62.16
Wiki10-31K	2.83	3.47
Wiki-500K	3.12	95.43

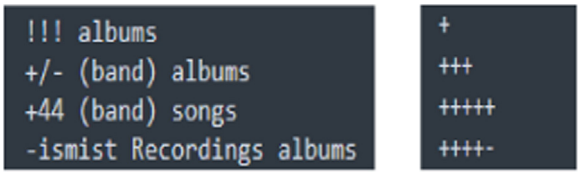


Fig. 4. Examples of the semantically lacking or confusing labels are taken from Wiki-500K and Wiki10-31K. Each line represents a label.

text. However, many labels in AmazonCat-13K, Wiki10-31K, and Wiki-500K are semantically lacking or confusing. We refer to them as symbolic labels. Fig. 4 provides examples of symbolic labels, such as '!!!albums' and '++++-.' Some labels are composed of low-level semantic characters. In contrast, others have words with complete semantics that are disrupted by invalid characters.

Thus, we conclude that GUDN is sensitive to semantic information, adversely affecting its performance on datasets with weak label semantics. However, it is beneficial for GUDN on datasets with strong label semantics. We further clarify this conclusion through ablation experiments. The label reinforcement strategy experiment also supports a similar conclusion.

Ablation Study: We conducted experiments to test the efficacy of the guide network in improving BERT's ability to extract features and solve the XMTC problem. Specifically, we tested three models: a single BERT model, BERT with a feature guide (which helps identify the latent space), and BERT with a link guide (which reduces the pressure on the ranking classifier). These experiments, when conducted separately, also underscored the importance of the three losses: feature loss, link loss, and classification loss. While the decrease in link loss was relatively slight, its effect was still significant. Notably, while GUDN is equivalent to LightXML when using a single BERT model, we were unable to achieve the same level of accuracy. Table 6 demonstrates the impact of different modules on accuracy. Table 7 demonstrates the impact of raw labels and multi-hot vectors on accuracy.

Results show that the model employing the guide network outperformed the single BERT model, demonstrating that the guide network helps fine-tune BERT to capture label-aware features in texts and labels, establish a close connection, and find the latent space. Table 6 also shows that the guide network is sensitive to label semantics, as evidenced by the most significant improvement in accuracy on the Eurlex-4K dataset. We found that both the feature guide and the link guide contributed to the accuracy and were indispensable. The feature guide contributed more to accuracy improvement than the link guide. The model performed best when the two worked in tandem.

Furthermore, we observed from Fig. 6 that using raw label semantics on the four datasets enabled faster convergence than multi-hot vectors. At the same time, losses are reduced even more. Raw labels, more in line with natural language norms, provide richer semantic information to explore the latent space between texts and labels.

Sensitivity analysis: In order to explore the sensitivity of GUDN to label semantics, we conducted a series of in-depth experiments.

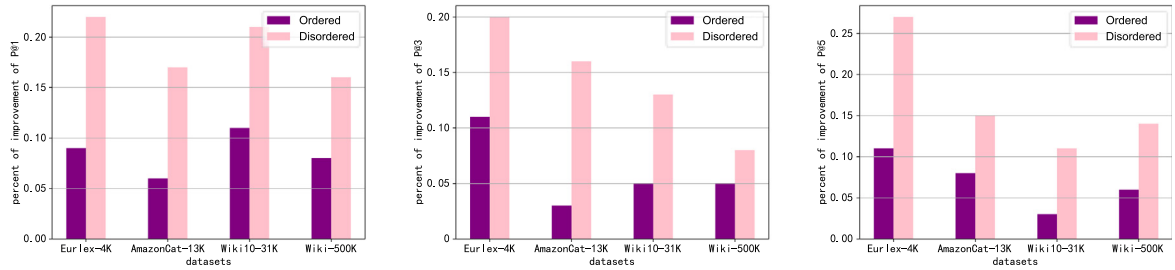


Fig. 5. On four different datasets, the enhanced effect at $P@1, P@3$, and $P@5$ uses the label reinforcement strategy with ordered and disordered settings, respectively.

Table 6
Comparison of ablation results on four datasets for each network module. GUD-F and GUD-L represent the feature guide and link guide, respectively.

	Eurlex-4K			AmazonCat-13K			Wiki10-31K			Wiki-500K		
Modules	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
BERT	86.68	75.04	63.03	96.10	82.89	67.01	88.76	77.65	68.51	77.37	58.09	45.46
BERT + GUD-F	**87.93**	**76.64**	**65.02**	**96.56**	**83.98**	**67.66**	**89.62**	**77.98**	**69.67**	**77.76**	**58.98**	**45.69**
BERT + GUD-L	87.21	75.09	63.93	96.12	83.10	67.28	89.21	77.63	68.94	77.41	58.16	45.62

Table 7
Experimental results on the Eurlex-4K, AmazonCat-13K, Wiki10-31K, and Wiki-500K datasets, comparing the improvement of $P@k$ over basic BERT using multi-hot vectors and raw labels as input.

	P@k	Raw labels	Multi-hot vectors
Eurlex-4K	P@1	+1.25	+0.03
	P@3	+1.60	+0.32
	P@5	+1.99	+0.13
AmazonCat-13K	P@1	+0.67	+0.18
	P@3	+0.99	+0.08
	P@5	+0.79	+0.16
Wiki10-31K	P@1	+0.86	+0.15
	P@3	+0.33	+0.18
	P@5	+1.16	+0.10
Wiki-500K	P@1	+0.76	+0.19
	P@3	+0.71	+0.22
	P@5	+0.29	+0.10

Based on the sensitivity assumption, we designed a set of semantic-free symbol labels to replace the labels in Eurlex-4K. Specifically, we randomly selected six characters from '0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, −, *' and arranged them into non-repeating new labels to replace the original labels. We substituted 25%, 50%, and 100% of the labels in the experiment, and the results are shown in Table 9. The experiment showed that GUDN is sensitive to label semantics, which can be both an advantage and a disadvantage.

4.4. Longformer vs BERT

We conducted an additional experiment to explore the influence of input text length. The limited input length of most Transformer-based models is an obvious flaw, as they have a limit of 512 for the input sequence length. The Longformer (Beltagy et al., 2020), on the other hand, can make the input sequence longer than 512, with a length of input that can reach more than 4096 tokens. We set the lengths of text input to 512, 1024, 2048, and 4096 to compare their influence, which can be regarded as a text reinforcement strategy. For simplicity, we only chose the header part of texts.

Table 8 shows the influence of the input length. The results may have a slight discrepancy because the Longformer has a few subtle structural differences from BERT. For Eurlex-4K and AmazonCat-13K, the predictive precision did not increase as expected but

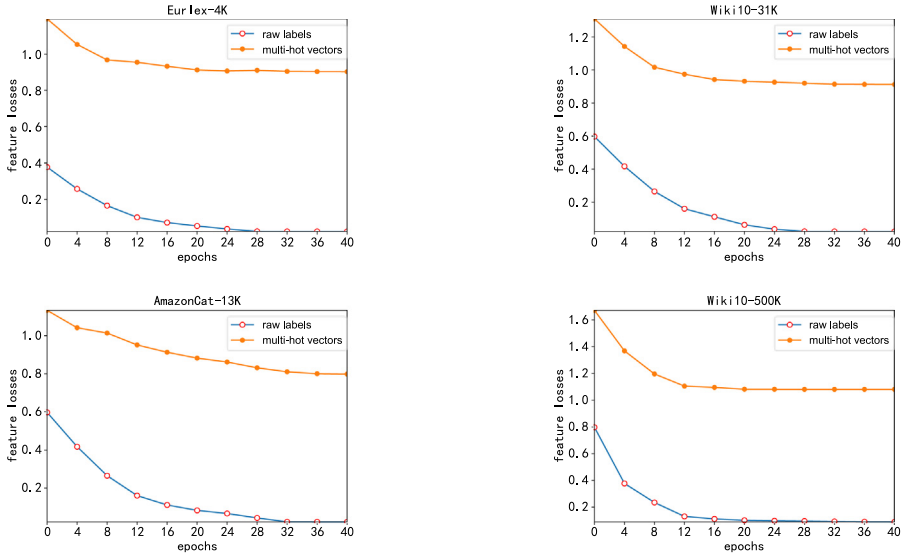


Fig. 6. An experiment showing the comparison of four datasets on the decay of feature losses with epochs when using raw labels and multi-hot vectors.

Table 8
A comparison of the effect of different input lengths on accuracy on four datasets.

	Eurlex-4K			AmazonCat-13K			Wiki10-31K			Wiki-500K		
Input length	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
512	88.01	**77.08**	**65.37**	96.18	84.15	67.86	89.70	78.58	69.87	77.87	59.16	45.98
1024	**88.12**	76.96	65.29	96.26	**84.21**	**67.99**	89.96	78.62	69.93	77.90	59.20	46.07
2048	88.03	76.90	65.13	96.25	84.07	67.79	90.08	78.71	70.03	77.94	59.29	46.12
4096	87.95	76.90	64.98	**96.73**	84.06	67.62	**90.21**	**78.83**	**70.16**	**78.01**	**59.31**	**46.18**

Table 9
Sensitivity Analysis of GUDN for label semantics.

25%			50%			100%		
P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
85.38	73.42	61.66	83.70	70.01	58.35	81.94	67.75	55.86

showed a slight decrease. However, for Wiki10-31K and Wiki-500K, the results showed an appreciable increase.

The length of 512 is sufficient to cover many texts in Eurlex-4K and AmazonCat-13K so it may be enough for these two datasets. However, a length of more than 512 with meaningless tokens may harm the results. On the other hand, the length can be tremendously long in the case of Wiki10-31K and Wiki-500K. Thus additional input length can result in an extra gain in predictive accuracy for an extremely long text. However, considering the computing time, it may not be suitable.

5. Conclusions and future work

This paper presents a novel guide network with a label reinforcement strategy for XMTC tasks. The experimental results demonstrate that GUDN achieves competitive performance on multiple datasets, especially on Eurlex-4K. The label reinforcement strategy overcomes the semantic gap and further improves performance. The ablation experiments prove that the guide network helps fine-tune BERT, and the feature guide and link guide play significant roles. GUDN is sensitive to the semantics of labels and demonstrates remarkable power for datasets with semantically rich labels. Although GUDN does not perform well on datasets with few semantic labels as it does on datasets with many semantic labels, it still achieves competitive results compared with state-of-the-art methods because most real-world labels are composed of natural language.

In an additional experiment testing four different input lengths, we found that expanding the input length is only sometimes helpful for classification precision because of the presence of harmful tokens.

Some studies have effectively repartitioned the datasets, resulting in more reliable data distribution. We may explore how GUDN fits into their new division rules. Furthermore, we have noticed some studies on few-shot and short-text. We will investigate how to combine these approaches with our method.

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.

Acknowledgment

This work was supported by the National Key R&D Program of China under Grant (2022YFC3303600), the Key Research and Development Program of Zhejiang Province under Grant (2022C03106), the National Natural Science Foundation of China under Grant (62077015), the Key Research and Development Program of Zhejiang Province under Grant(No.2021C03141), and the Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Zhejiang, China.

References

Babbar, R., Schölkopf, B., 2019. Data scarcity, robustness and extreme multi-label classification. Machine Learn. 108, 1329–1351.
Babbar, R., Shoelkopf, B., 2016. Dismec - distributed sparse machines for extreme multi-label classification, arxiv:1609.02521. arXiv:arXiv:1609.02521.
Beltagy, I., Peters, M.E., Cohan, A., 2020. Longformer: The long-document transformer. URL: https://arxiv.org/abs/2004.05150, https://doi.org/10.48550/ARXIV.2004.05150.
Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P., 2015. Sparse local embeddings for extreme multi-label classification. Adv/ Neural Informat. Process. Syst. 28.
Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu, Y., Varma, M., 2016. The extreme classification repository: Multi-label datasets and code, http://manikvarma.org/downloads/xc/xmlrepository.html. URL: http://manikvarma.org/downloads/XC/XMLRepository.html.
Chang, W.C., Yu, H.F., Zhong, K., Yang, Y., Dhillon, I., 2019. Taming pretrained transformers for extreme multi-label text classification, arxiv:1905.02331. arXiv:arXiv:1905.02331.
Cisse, M.M., Usunier, N., Artieres, T., Gallinari, P., 2013. Robust bloom filters for large multilabel classification tasks. Adv. Neural Informat. Process. Syst. 26.
Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding, arxiv:1810.04805. arXiv:arXiv:1810.04805.
Guo, C., Mousavi, A., Wu, X., Holtmann-Rice, D.N., Kale, S., Reddi, S., Kumar, S., 2019. Breaking the glass ceiling for embedding-based classifiers for large output spaces. Adv. Neural Informat. Process. Syst. 32.
Hsu, D., Kakade, S.M., Langford, J., Zhang, T., 2009. Multi-label prediction via compressed sensing. URL: https://arxiv.org/abs/0902.1284, doi:10.48550/ARXIV.0902.1284.
Huang, X., Chen, B., Xiao, L., Jing, L., 2019. Label-aware document representation via hybrid attention for extreme multi-label text classification, arxiv:1905.10070. arXiv:arXiv:1905.10070.
Jain, H., Prabhu, Y., Varma, M., 2016a. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 935–944.
Jain, H., Prabhu, Y., Varma, M., 2016b. Extreme multi-label loss functions for recommendation, tagging, ranking and other missing label applications. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 935–944. URL: https://doi.org/10.1145/2939672.2939756.
Jain, H., Balasubramanian, V., Chunduri, B., Varma, M., 2019. Slice: Scalable linear extreme classifiers trained on 100 million labels for related searches. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 528–536. URL: https://doi.org/10.1145/3289600.3290979.
Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., Hullermeier, E., 2016. Extreme f-measure maximization using sparse probability estimates. In: Balcan, M.F., Weinberger, K.Q. (Eds.), Proceedings of The 33rd International Conference on Machine Learning, PMLR, New York, New York, USA. pp. 1435–1444. URL: https://proceedings.mlr.press/v48/jasinska16.html.
Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., Zhuang, F., 2021. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification, arxiv:2101.03305. arXiv:arXiv:2101.03305.
Karimi Jafarbigloo, S., Danyali, H., 2021. Nuclear atypia grading in breast cancer histopathological images based on CNN feature extraction and LSTM

classification. CAAI Trans. Intell. Technol. 6 (4), 426–439. https://doi.org/10.1049/cit2.12061.
Khandagale, S., Xiao, H., Babbar, R., 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. Machine Learn. 109, 2099–2119.
Liu, B., Sadeghi, F., Tappen, M., Shamir, O., Liu, C., 2013. Probabilistic label trees for efficient large scale image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 843–850.
Liu, J., Chang, W.C., Wu, Y., Yang, Y., 2017. Deep learning for extreme multi-label text classification. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Association for Computing Machinery, New York, NY, USA. p. 115–124. URL: https://doi.org/10.1145/3077136.3080834.
Merrillees, M., Du, L., 2021. Stratified sampling for extreme multi-label data, arxiv:2103.03494. arXiv:arXiv:2103.03494.
Mittal, A., Dahiya, K., Agrawal, S., Saini, D., Agarwal, S., Kar, P., Varma, M., 2021a. Decaf: Deep extreme classification with label features. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 49–57. URL: https://doi.org/10.1145/3437963.3441807.
Mittal, A., Sachdeva, N., Agrawal, S., Agarwal, S., Kar, P., Varma, M., 2021b. Eclare: Extreme classification with label graph correlations. In: Proceedings of the Web Conference 2021, Association for Computing Machinery, New York, NY, USA. p. 3721–3732. URL: https://doi.org/10.1145/3442381.3449815.
Niculescu-Mizil, A., Abbasnejad, E., 2017. Label filters for large scale multilabel classification. In: Artificial intelligence and statistics, PMLR, pp. 1448–1457.
Prabhu, Y., Varma, M., 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA. p. 263–272. URL: https://doi.org/10.1145/2623330.2623651.
Prabhu, Y., Kag, A., Gopinath, S., Dahiya, K., Harsola, S., Agrawal, R., Varma, M., 2018a. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 441–449.
Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., Varma, M., 2018b. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In: Proceedings of the 2018 World Wide Web Conference, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE. p. 993–1002. URL: https://doi.org/10.1145/3178876.3185998.
Prajapati, P., Thakkar, A., 2022. Performance improvement of extreme multi-label classification using k-way tree construction with parallel clustering algorithm. J. King Saud Univ.-Comput. Informat. Sci. 34, 6354–6364.
Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I., 2021. Learning transferable visual models from natural language supervision. In: Meila, M., Zhang, T. (Eds.), Proceedings of the 38th International Conference on Machine Learning, PMLR. pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html.
Siblini, W., Kuntz, P., Meyer, F., 2018. Craftml, an efficient clustering-based random forest for extreme multi-label learning. In: International Conference on Machine Learning, PMLR., pp. 4664–4673.
Tagami, Y., 2017. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In: the 23rd ACM SIGKDD International Conference, Association for Computing Machinery, New York, NY, USA, pp. 455–464. https://doi.org/10.1145/3097983.3097987.
Wang, B., Chen, L., Sun, W., Qin, K., Li, K., Zhou, H., 2019. Ranking-based autoencoder for extreme multi-label classification, arxiv:1904.05937. arXiv:arXiv:1904.05937.
Wydmuch, M., Jasinska, K., Kuznetsov, M., Busa-Fekete, R., Dembczynski, K., 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification, arxiv:1810.11671. arXiv:arXiv:1810.11671.
Xu, C., Tao, D., Xu, C., 2016. Robust extreme multi-label learning. In: Acm Sigkdd International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA, pp. 1275–1284. https://doi.org/10.1145/2939672.2939798.
Yeh, C.K., Wu, W.C., Ko, W.J., Wang, Y.C.F., 2017. Learning deep latent spaces for multi-label classification, arxiv:1707.00418. arXiv:arXiv:1707.00418.
Yen, I.E., Huang, X., Dai, W., Ravikumar, P., Dhillon, I., Xing, E., 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 545–553.
You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., Zhu, S., 2018. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification, arxiv:1811.01727. arXiv:arXiv:1811.01727.
Zhang, W., Yan, J., Wang, X., Zha, H., 2018. Deep extreme multi-label learning. In: The 2018 ACM. Association for Computing Machinery, New York, NY, USA, pp. 100–107.
Zhang, R., Wang, Y.S., Yang, Y., Yu, D., Vu, T., Lei, L., 2022a. Long-tailed extreme multi-label text classification with generated pseudo label descriptions. URL: https://arxiv.org/abs/2204.00958, https://doi.org/10.48550/ARXIV.2204.00958.
Zhang, Y., Shen, Z., Wu, C.H., Xie, B., Hao, J., Wang, Y.Y., Wang, K., Han, J., 2022b. Metadata-induced contrastive learning for zero-shot multi-label text classification. URL: https://arxiv.org/abs/2202.05932, https://doi.org/10.48550/ARXIV.2202.05932.

Qing Wang received a B.E. degree from Zhejiang Wanli University, China, in 2021. Currently, he is a graduate student at the College of Mathematics and Computer Science of Zhejiang Normal University. His research interests include data mining and artificial intelligence.

Jia Zhu is a Distinguished Professor at the School of Teacher Education, Zhejiang Normal University, and the Deputy Director of the Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province. He received his Ph.D. degree from the University of Queensland, Australia. His research interests include intelligent education, theoretical algorithms for database and data mining, federated learning, and blockchain with AI.

Hongji Shu received a B.E. degree from Zhejiang Normal University, China, in 2018. Currently, he is a graduate student at the College of Mathematics and Computer Science of Zhejiang Normal University. His research interests include artificial intelligence and knowledge graph.

Kwame Omono Asamoah received a B.Sc. degree in computer science from the Kwame Nkrumah University of Science and Technology, Ghana, in 2014. He received his master's degree in computer science and technology from the University of Electronic Science and Technology of China in 2018. He had his doctorate degree in computer science and technology from the University of Electronic Science and Technology of China in 2022. He is currently a postdoctoral fellow at Zhejiang Normal University. His current research includes blockchain technology and big data security.

Jianyang Shi is a graduate student in the school of teacher education of Zhejiang Normal University. He received his B.E. degree from the Jiangxi Science and Technology Normal University, China. His research interests lie in Classroom multimodal data mining and analysis.

Zhou Cong received a B.E. degree from Zhejiang Wanli University, China, in 2021. Currently, he is a graduate student at the College of Mathematics and Computer Science of Zhejiang Normal University. His research interests include data mining and text generation.