Multi-Label text classification (MTC)

1st Danlei Zhu
BNU-HKBU United International College, Zhuhai, China
2130201617

2nd Siyu Teng
BNU-HKBU United International College, Zhuhai, China
2130201702

3rd Junhao Huang
BNU-HKBU United International College, Zhuhai, China
2130201710

***Abstract*—Multi-Label text classification (MTC) is a well-known NLP task that can be applied in many fields, for example, news annotation, web page tagging, and item categorization. Because of the unequal distribution of datasets, improving the overall efficiency and accuracy of MTC task is considered quite difficult. This is mainly due to the fact that there are significantly more positive labels than negative labels. The imbalance between positive and negative labels causes great difficulties for MTC. In order to solve the Extreme multi-label text classification (XMTC) task, we present a new deep learning model, called LightAttention, by integrating the dynamic negative label sampling network into the existing AttentionXML model. The overall structure of LightAttention mainly consists of three components: BiLSTM, multi-label attention layer, and dynamic negative label sampling network. Here, the BiLSTM helps to capture the long-distance context information from texts, thus increasing the matching possibility from text to labels. The multi-label attention layer encode each given text into a specific representation for different label. Then, by combining the multi-label attention with the proposed dynamic label sampling network, we are able to further improve the overall accuracy on tail labels by dynamically generating more negative label samples. Experiment results show that the LightAttention slightly outperform the AttentionXML on three datasets: Eur-Lex, Wiki10-31K, and AmazonCat-13K.**

***Index Terms*—Text classification, NLP, deep learning, multi-label.**

I. INTRODUCTION

Natural Language Processing (NLP) [1], as a multidisciplinary field of linguistics, computer science, and artificial intelligence, has received extensive research attention due to the rapid development of machine learning methods recently. Compared to the traditional statistical NLP, deep neural network shows impressive suitability in NLP, which can achieve state-of-the-art results in many natural language tasks, especially for extreme multi-label text classification (XMTC) task [2], [3]. Due to recent development of the Internet, the application scenarios of MTC are prevalent in our daily life, such as item categorization in Taobao, news annotation, recommendation system, website tagging, and so on. The objective of XMTC is to label a given text with multiple possible labels from an extremely large-scale label set. This is distinct from the multi-class classification [4], which only tag each given text one single label.

During the era of big data, the data scale of the information systems has experienced massive growth. The Internet service providers have to handle millions of labels and samples. Text classification for those extreme scale datasets becomes an extremely difficult NLP task. The difficulties of XMTC mainly comes from two parts. First, the training phrase of XMTC for extreme datasets requires significant computational challenges, which brings many troubles in developing effective deep learning model for developers. Besides, millions of labels (tail labels) experience the lack of positive samples, thus decreasing the training effectiveness on these tail labels.

According to the representation of the input datasets and labels, we can classify the existing works into two categories: (1) Utilize the semantic features of the labels or input datasets as the training datasets. This type of strategy requires extra effort to extra semantic features such as bag-of-words (BOW) [5] from the input texts. (2) Directly use the raw input text as the training dataset. This method directly feed the training model with sparse vectors of the texts.

In order to improve the training accuracy for tail labels, many works [3], [6]–[8] have been presented by using label partitions or probabilistic label trees (PLTs) strategies that can exploit the relations within labels. Among which, Parabel [6] utilizes the BOW information of the labels, which ignores the context information of long-distance dependency of words. Besides, Parabel's tree-based methods for labels classifies a lot of dissimilar labels into a single cluster, which further decreases the classification accuracy for tail labels. AttentionXML [3] addresses the above problems in Parabel by using raw text information and a more shallow and wide PLT model for training. Using raw text information together with a Bidirectional-LSTM (BiLSTM) model takes more features, especially the context information among words, into consideration, thus capturing the most relavent parts of text and utilizing them into the follow-up classification phrase. They also represent the labels with different representation, which is claimed to be helpful for tail labels. Moreover, using a shallow and wide PLT results in faster training efficiency, which alleviates the computational pressure of XMTC task by avoiding a deep tree structure. Despite all these improvements and advantages, there is still one major shortcoming in AttentionXML. They used a static negative sampling strategy for labels, hence most of the tail labels were trained on a small amount of samples. Therefore, the efficiency and accuracy of the XMTC on tail labels will be significantly affected.

This project is greatly inspired by the AttentionXML and aims to address the major shortcoming in AttentionXML. To do this, we propose a generative coopreative networks that can dynamically generate negative labels. With this dynamic negative label sampling function, we are able to generate quite a series of both possitive and negative labels for tail labels, thus increasing the overall training accuracy on tail labels. In all, we present a new deep learning model, called LightAttention, by integrating the dynamic negative label sampling network into the existing AttentionXML model. The overall structure of LightAttention mainly consists of three components: BiLSTM, multi-label attention layer, and dynamic negative label sampling network. Here, the BiLSTM helps to capture the long-distance context information from texts, thus increasing the matching possibility from text to labels. The multi-label attention layer encode each given text into a specific representation for different label, which is helpful for training on tail labels. By combing the multi-label attention with the proposed dynamic label sampling network, we are able to further improve the overall accuracy on tail labels by dynamically generating more negative label samples. Experiment results show that the LightAttention slightly outperform the AttentionXML on three datasets: Eur-Lex, Wiki10-31K, and AmazonCat-13K.

II. Related works

Many works have achieved better performance on Extreme Multi-label. Generally, the approaches can be broadly categorized into five types, one-vs-all OVA approach, tree-based approach, embedding-based approach, deep learning approach and transformer approach.

a) One-vs-all OVA Approach: One-vs-all (OVA) Approach is a heuristic technique for multi-class classification utilizing binary classification algorithms that categorizes multi-class dataset into multiple binary problem. The meaningful contribution is prediction accuracy improvement. But the framework computational size and speed are the major challenges. Several works have tried overcome above disadvantages. The prior research DiSoEMC(Distributed Sparse Machines) [9] is the first work to attempts scaling up one-to-one paradigm in extreme multi-label classification problems by parallel training speed-up. ProXML [10] modified DiSEMC and focus on long-tail label prediction. Similarity, PPDSparse [11] and PD-Sparse [12] use dual sparsity to accelerate training and prediction.

b) Tree-based Approach: Tree based method is a solution in computational issue. In addition, the well known advantages are less training and prediction time by recursively splitting the labels or features. For instance, FastXML is a classifier for extreme multi-label classification that uses a nDCG-based ranking loss function [13]. It learns a hierarchy structure over the feature space rather than the label space. Similarity, the label partition can use Gini index to evaluate the performance [14]. The Parabel [6] recursively splitting the labels into two balanced groups produces each label tree. If nodes have less than M labels, they become leaves and are not partitioned further. The leaf nodes have linear 1-vsAll classifiers, one for each label in the leaf, that have only been trained on samples that have at least one leaf node label. Negative examples for training a label's classifier come from other labels in the same leaf as the provided label.

c) Embedding-based Approach: Embedding models employ a low-rank representation for the label matrix in order to a low-dimensional search for label similarity. Embedding-based approaches suppose the label matrix space can be represented by a low-dimensional latent space with comparable latent representations for related labels. For example, SLEEC (Sparse Local Embedding for Extreme Classification) [15] reduce the number of labels by embedding labels onto low dimensional space. Typically, SLEEC uses k-nearest neighbor clusters labels into small groups. In addition, The [16] introduces a standard empirical risk minimization framework by using various loss functions and regularization. However, embedding-based models generally perform worse than sparse one-vs-all techniques like PPDSparse [11] to achieve equivalent computational speedups, which might be attributed to the inefficiency of the label representation structure.

d) Deep Learning Approach: With the development of neural network architecture, many deep learning model have shown better improvement in XMC problems. XML-CNN [2] is the first work to implement deep neural network in XMC. It learns text representation by forwarding training to CNN networks. XML-CNN also contains a hidden layer to project text features onto low dimensional space in order to reduce the model computational size. However, unlike basic multi-label classification, XML-CNN only utilizes a simple fully connected layer to score all labels with binary entropy loss, making it difficult to deal with big label sets. AttentionXML [3] uses a probabilistic label tree (PLT) that can deal millions of labels instead of a simple fully connected layer for label scoring. For a single dataset, it must train on many models. AttentionXML solves this problem by multiplying the weight of the current layer model by the weight of its upper layer model, allowing the model to converge fast.

e) Transformer Approach: The NLP proposes a new concept: pre-training then fine-tuning. BERT [17] is one of the pioneering efforts whose pre-training targets include token prediction and following sentence prediction tasks. On the other hand, Transformer model outperforms existing state-of-the-art after pre-training on large-scale unsupervised corpora like Wikipedia and BookCorpus. X-transformer [18] is the first pre-trained model implementation on XMC problems. Compared with AttentionXML, X-transformer has higher accuracy. But there are two major shortcomings, firstly, the computational size of model. Secondly, negative labels sampling reduces the prediction accuracy. X-BERT [19] learns the label representations from the label and the input text. The process for fine-tuning BERT models to capture the contextual link between input text and the generated label clusters is the important component of X-BERT. An ensemble of multiple BERT models trained on heterogeneous label clusters yields best final model.

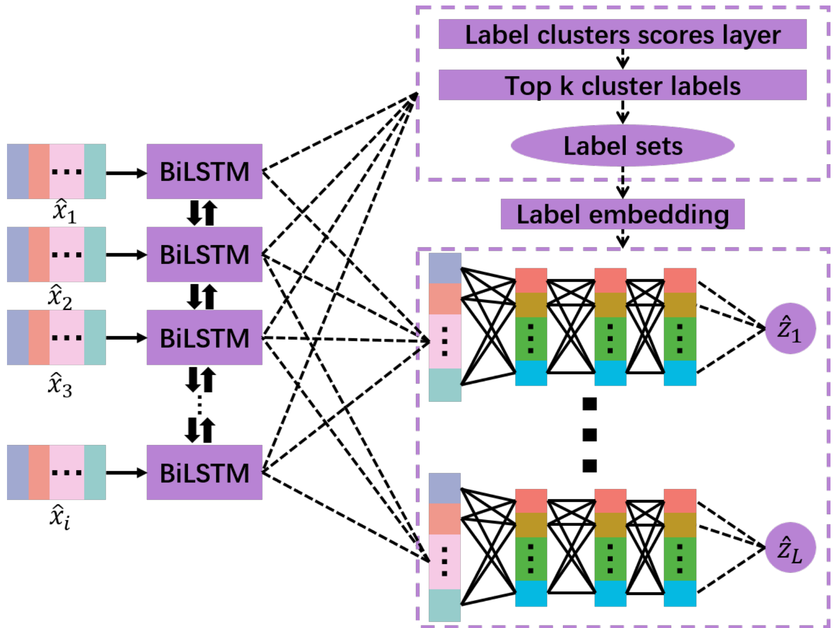

Fig. 1. The overall structure of LightAttention

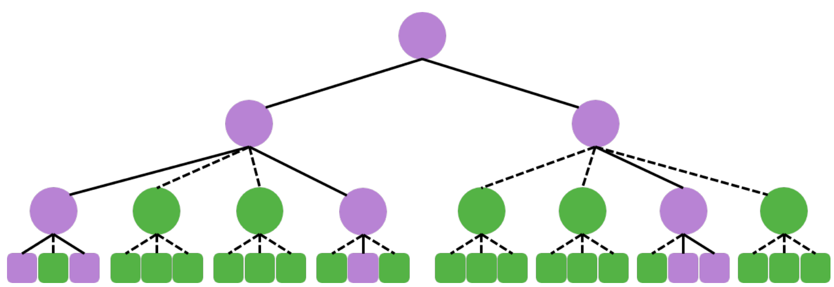

Fig. 2. Shallow and wide probabilistic label tree for label clustering

III. Methodology

A. Problem Formulation and Overview

The XMTC task is to find multiple relevant labels for each given raw text. For a training set $\{x_i, y_i\}_{i=1}^{N}$, where x_i is a raw text, $y_i \in \{0,1\}^L$ is the relevant label of x_i and y_i is a multi-hot L-dimensional vectors. The goal of XMTC is to train a model that can better simulate the function $f(x_i) \in R^L$, such that f output a high score for the l-th label y_{il} if y_{il} is relevant to the text x_i. However, one significant problem in XMTC is that L can be up to millions, which makes it impossible to directly train a model on L-dimensional vectors due to the high workload. Therefore, in order to perform multi-label classification over tens of thousands or millions of label set, we first need to construct a probabilistic label tree (PLT) to divide massive number of labels into smaller label clusters, thus accelerating the process of label classification. After we build a PLT for label clustering, we then train an attention-aware deep learning model LightAttention by combining the BiLSTM, multi-label attention layer and generative cooperative networks for negative label sampling. The overall structure of the model is shown in Figure 1.

B. Building a PLT for label clustering

For the PLT construction, we follow the method shown in AttentionXML [3], which constructs a shallow and wide PLT from the original deep PLT presented in Parabel [6]. Their PLT construction algorithm first utilizes the KMeans (K=2) algorithm to generate a hierarchical deep PLT. After that, they presents an algorithm to split down the layer of the deep PLT due to the reason that a deep PLT would result in slower performance.

The so-called PLT is to construct a tree with L leaves. Here, each leaf represent a unique label. Suppose that there exists a text x, for this given text, we assign each node in the PLT a value $z_n \in \{0,1\}$. $z_n = 0$ means that the children node of n doesn't have any relations with the given text x. Otherwise, $z_n = 0$ indicates that there exists at least one children node of n is relevant to x. An example of the shallow and wide PLT can be found in Figure 2, during which the nodes in purple represent $z_n = 1$ and the nodes in green indicate $z_n = 0$. PLT evaluates the conditional probability that each node n's relevance with x by computing $P\left(z_n \mid z_{Pa(n)} = 1, x\right)$, where $Pa(n)$ is the parent node of n. Then, the probability that how each node n is relevant to x can be simply computed with Equation 1.

$$P\left(z_n=1 \mid x\right)=\prod_{i \in Path(n)} P\left(z_i=1 \mid z_{Pa(i)}=1, x\right) \quad (1)$$

The $Path(n)$ refers to the nodes appeared between node n and root.

A PLT has two important parameters: tree height H and cluster size M. If these two parameters are too big, then the overall performance of the PLT would be very slow. Therefore, we follow the method in AttentionXML and build a shallow and wide PLT T_H, reducing both the tree height H and wide M. The overall procedure of this algorithm is shown in algorithm 1. This algorithm takes T_0, which is built with the Parabel method [6], as input, and it performs the compression function H times over the parents of leaves S_0. This algorithm first select c-th ancestor nodes as S_l. Then, remove the nodes between S_{l-1} and S_l to reduce the overall number of nodes. Finally, reset the tree based on the new nodes. After these steps, a shallow and wide tree T_H with smaller height H and smaller wide M can be obtained.

ALGORITHM 1: The shallow and wide PLT construction

Input: Labels of training texts $\{y_i\}_{i=1}^N$; Initial PLT T_0; $K=2^c, H$
Output: A shallow and wide PLT T

```
Initialize parent nodes of leaves S_0
for l ∈ [1, H] do
    if l < H then
        S_l ← {c-th ancestor node n of nodes in S_{l-1}}
    else
        S_l ← {the root of T_0}
    end if
    T_l ← T_{l-1}
    for nodes n ∈ S_l do
        for nodes n' ∈ S_{l-1} and node n is the ancestor of n' in T do
            Pa(n') ← n
        end for
    end for
end for
return T_H
```

C. Learning LightAttention and Generative Cooperative Networks

After we construct a PLT, we need to train a deep model among the PLT. For a deep PLT, the nodes near the bottom layer is very difficult since the labels. Instead of training the model for all nodes together, AttentionXML, i.e., a level-wise and top-down manner for each level of nodes. We define LightAttention$_d$ as the training procedure of training the d-level candidate nodes $g(x)$ for given sample x. The candidate nodes is selected by first sortting the $(d-1)$-level nodes by z_n (from positive to negative) and the scores of each nodes obtained in LightAttention$_{d-1}$. Then, the children nodes of the top k nodes at the $(d-1)$-level is the training candidate $g(x)$. LightAttention$_1$ can be directly computed for the root nodes.

During the original AttentionXML, the candidate nodes $g(x)$ obtained by the top k function may consist of both positive and negative nodes. The negative nodes obtained here are all static negative label samples and sometimes, there may be no negative label samples at all. The trained model would overfit due to the lack of negative samples. The convergence of the model with static negative samples would also be difficult due to the high similarity between the positive and negative labels in the original AttentionXML. To address this problem in AttentionXML, we adopt the strategy presented in [3] and deploy Generative Cooperative Networks (GCN) to generate negative label samples dynamically so that the training procedure can distinguish more negative samples from positive samples, reducing the overfitting issue in AttentionXML.

Specifically, the GCN for LightAttention consists of the two components: label generator, discriminator. These two components work as a cooperative networks. The label generator first generates negative labels and delivers the the negative samples into the discriminator to learn the difference between negative and positive label samples. Here, the discriminator can achieve better label representation with the help of large dynamic negative samples.

The generator is a fully connected layer $G(e) = \sigma\left(W_g e+b_g\right)$ that output a K-dimensional vectors representing the scores of all K label clusters. Top b label clusters with highest scores are choosen at the generation seed of labels. Then, all these labels together with positive labels are added as the training label sets so that the deep model can be trained to recognize the postive and negative labels. The generator loss is computed as Equation 2, where $y_g \in 0,1^K$ represents the multi-hot label representation of label cluster for x.

$$\begin{aligned}\mathcal{L}_g\left(G(e), y_g\right)=\sum_{i=0}^K\left(1-y_g^i\right)\left(-\log \left(1-G(e)^i\right)\right)+ \\ y_g^i\left(-\log \left(G(e)^i\right)\right)\end{aligned} \quad (2)$$

D. Attention-Aware deep model

As shown in Figure 1,the deep model of LightAttention can be briefly divided into fix layers: text representation layer, BiLSTM layer, generative cooperative networks, multi-label attention layer, fully connected layer and output layer.

1) Text representation layer: The input text of our model is raw text with length $\hat{T}$. The famous 300-dimensional GloVe [20] word embedding representation is used in our model.

2) BiLSTM layer: BiLSTM, abbreviation of Bidirectional long short-term memory, is one of the recursive neural networks (RNN). The overall structure of BiLSTM is shown in Figure 3. It is equipped with two LSTM layers: one forward LSTM and one backward LSTM. Each LSTM is used to capture either forward or backward context information in a raw text, thus providing much more adequate context

ALGORITHM 2: GCN for LightAttention

Input: Labels of training texts $\{X, Y\} = \{(x_i, y_i)\}_{i=1}^{N}$, semantic features of the training text $\hat{X}$
Output: Label embedding M

Label clusters C using $\hat{X}, Y$
Discriminator D initialization over cluster C
Label embedding E, generator G initialization over cluster C
Get m samples from X, Y: X_{batch}, Y_{batch}
Get text embedding $\hat{h}$ from BiLSTM
for $l \in [1, m]$ **do**
    Generate label clusters $S_{generated}$ using $G(\hat{h_l})$
    Select negative labels S_{neg} using $S_{generated}, C$
    Delete positive labels from S_{neg}
end for
Generate positive labels S_{pos} using Y_{batch}
for $l \in [1, m]$ **do**
    Generate label embedding M using S_{pos}, S_{neg}
end for
return M

information for later usage. The use of BiLSTM is the major reason that we can accept raw text as input in our model. The output $\hat{h_t}$ (t is the time step) of the BiLSTM in our model is the combination of both forward and backward outputs.

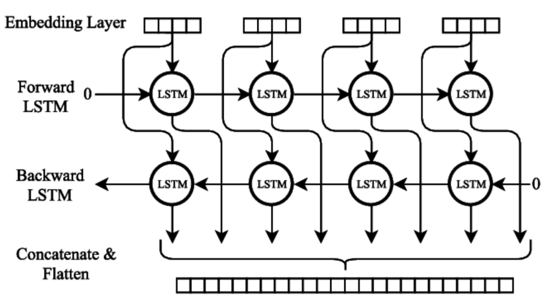

Fig. 3. Structure of BiLSTM

3) Multi-label attention: Attention mechanism [21] is to permit the decoder to utilize the most relevant parts of the input sequence in a flexible manner, by a weighted combination of all of the encoded input vectors, with the most relevant vectors being attributed the highest weights. In XMTC, the most relevant context that each label captures would be totally different. We adopts the multi-label attention shown in AttentionXML, which detect the intensive parts of text for multiple labels by computing:

$$\hat{\mathbf{m}}_j = \sum_{i=1}^{\hat{T}} \alpha_{ij} \hat{\mathbf{h}}_i, \quad \alpha_{ij} = \frac{e^{\hat{\mathbf{h}}_i \hat{\mathbf{w}}_j}}{\sum_{t=1}^{\hat{T}} e^{\hat{\mathbf{h}}_t \hat{\mathbf{w}}_j}} \tag{3}$$

where $\hat{m_j} \in R^{2\hat{N}}$ is the output of multi-label attention, α_{ij} is the normalized coefficient of $\hat{h_i}$ and $\hat{w_j} \in R^{2\hat{N}}$ is the attention parameters.

4) Fully connected layer and output layer: LightAttention is equipped with 3 or 4 layers of fully connected layers and an output layer.

IV. Experiments

In this section, we mainly introduce the experimental datasets, experimental evaluation measures, platform configuration and experimental settings and performance comparison results.

A. *Dataset*

Extreme multi-label learning is a very complex NLP problem that aims to train a classifier that can automatically label a new data point from an extremely large set of labels and provide the most relevant subset of labels.In order to obtain a classifier with good results, an efficient data structure is essential. We choose three most famous extreme multi-label text classification (XMTC) for our experiments: three large-scale datasets (range from 4K to 30K): EUR-Lex, Amazon-670K and Wiki10-31K (shown in I). E is the number of epochs, B represents the batch size.

a) Amazon-670K: Amazon-670K [22] provides a huge dataset for evaluating the performance of extreme multi-label text classification (XMTC) including evaluation code and judgment metrics.it is mainly composed of web page information and company product information, which is an excellent training dataset for XMTC tasks with complex types and large data volumes. There are 135,909 bow feature dimensionalities, 670,091 labels, 490,449 train dataset, and 153,025 test dataset. To keep balance in the distribution of data, Amazon-670K also provides over 3.99 dates per label and 5.45 labels per data. Amazon-670K is very perfect for pre-processing the data. For small-scale datasets, the authors have provided the complete data in one file. We provide separate files for the training and test splits, which contain the indexes of the points in the training and test sets. Each column of the split file contains a separate split. For large-scale datasets, the authors provide a train and a test split into two separate files.Amazon-670K also provides the option to download pre-computed (e.g., a packet of words) features or raw text, and the tokenization that can be used to create a packet of word representation may vary across datasets.

b) EUR-Lex: EUR-Lex is a famous and official website for European Union laws and other public documents, and it has published 24 official languages in the European Union. A large amount of data exists on this website for English-speaking nature languages, and based on this data,Ilias Chalkidis and Manos Fergadiotis [23] proposed EUR-Lex dataset. The Eur-Lex dataset is suitable for LMTC task and zero-short/few-short learning, this allowed user to bypass BERTs maximum text length limit and fine-tune BERT, obtaining the best results in all but zero-shot learning cases.

EUR-Lex contains 57 thousand legal instruments with a length of 750 words. Each instrument includes 4 mainly zones: the header, which imports the title and the abstract of this legal instrument body; the recitals, which are the

TABLE I
HYPERPARAMETERS USED IN EXPERIMENTS, PRACTICAL COMPUTATION TIME AND MODEL SIZE

Datasets	E	B	N	Nfc	Train	Test	Model Size
EUR-Lex	30	40	512	512	0.74	2.03	0.58
Wiki10-31K	30	40	512	512	1.68	5.65	0.68
AmazonCat-13K	10	200	512	512	20.5	1.53	0.95

background of legal references and also contain the relevant laws and legal texts; the main body, the most important part of the document, which focuses on the organization in articles; and the attachments, Attachments about the instrument, such as evidence, photos, minutes of meetings, court dates, etc. Since LMTC task requires large size documents to be input as smaller units (e.g., sentences), but in EUR-Lux they are chapters, so the author preprocessed the raw text of the dataset. they consider the header,the recitals, the main body, and the attachments as separate sections. In addition to that, authors alse split EUR-Lex into training dataset, development dataset and testing dataset, which are have 45 thousand instruments, 6 thousand instruments and 6 thousand instruments.

EUR-Lex provide 4,271 labels and divide them into frequent labels (746), few-short labels (3,362) and zero-shot labels (163).

c) Wiki10-31: Wiki10-31K [24] is also a well-known NLP dataset for XMTC task training. It was created based on the content of a Wiki, resulting in its inclusion of a lot of content, ranging from astronomy down to geography. The excessive variety of data also makes it difficult to fit and any model will have difficulty achieving good data results on it.

Wiki10-31K mainly contains instances (20,762), attributes (132,876), labels 930,938, and labelsets (20693), but the density of Wiki10-31K only is 0.0006, that's why many famous algorithms is hard to get an excellent result.Wiki10-31 was used in our experiments to test the robustness of the algorithm in extreme cases.

B. *Experimental Evaluation Measures*

In our experiments, we choose $P@k$ (k is Precision) as evaluation metrics for our method named LightAttention and AttentionXML, $P@k$ is broadly used as an evaluation indicator for NLP XMTC (shown in 4).

$$P@k = \frac{1}{k}\sum_{l=1}^{k} y_{rank(l)} \tag{4}$$

Note $y \in \{0,1\}^L$ stand for ground turth binary vector, and $rank(l)$ points out the toppest l of the predicted label. Another well-known metric is $N@k$ (stand for normalized discounted Cumulative Gain at k point). In this equasion, we know that $P@1 == N@1$. And then we evaluate algorithm's performance by $N@k$, though prove process, we can calculate that the trend of $N@k$ keep the same speed as $P@k$, so we can omit the results of N@k in the main text due to space limitation.

C. *Experimental Settings*

LightAttention is an NLP algorithm focused on XMTC, so we compared it with the most representative XMTC method called AttentionXML. And In order to increase the authority and feasibility of the experiments we did several experiments in three data sets Amazon-670K, EUR-Lex, and Wiki10-31K respectively, and used their average result values to obtain the final results. To ensure the accuracy of the experiments, we perform a simple pre-processing of the datasets. For each dataset, the most frequent words in the training set are extracted and formed into a new efficient training set (no more than 500,000). And for EUR-Lex and Wiki10-31K, we fine-tune the word embeddings during the training process.

For more efficient training and more accurate prediction, we slice the data into units of 500. After the embedding layer, we use dropout = 0.3 to avoid overfitting of the neural network, and after the BiLSTM layer, dropout = 0.4. In addition to this, LightAttention has a learning rate of 1e-3. We also use SWA to improve performance.

TABLE II
ADD CAPTION

Dataset	P@1=N@1	P@3	P@5	
Amazon-670K	**86.56**	**72.98**	60.21	AttentionXML
	86.53	71.58	**61.92**	LightAttention
EUR-Lex	47.5	41.56	35.32	AttentionXML
	48.6	**43.21**	**38.54**	LightAttention
Wiki10-31k	**87.4**	77.4	68.21	AttentionXML
	86.5	**77.6**	**68.35**	LightAttention

D. *Performance comparison*

Table II A stands for the performance between LightAttention and AttentionXML by $P@k$ evaluation metric, and we test those methods in EUR-Lex, Amazon-670K, and Wiki10-31K several times. Because we import subsections from LightXML, LightXML has an excellent ability to handle extremely less label text classification tasks. The experimental results are clearly evident in EUR-Lex dataset. LightAttention has better experimental results in $P@1$, $P@2$ and $P@3$, compared to AttentionXML, LightAttention improves accuracy by 3.22% in the $P@2$ branch of EUR-Lex. And the experiments in others dataset shows that LightAttention has the same dataset processing capabilities as AttentionXML when dealing with common datasets. Another point, as shown in the Figure 4, is that LightAttention has a lower loss function than AttentionXML during training, which also proves that LightAttention has an outstanding ability when dealing with some extreme data.

E. The Number of PLTs Impact

We tested the influence of LightAttention with different number of PLTs on the model performance through extensive experiments, as shown in Table **??** Through these experiments, we can clearly see that more branches can substantially improve the accuracy of prediction. However, using more trees requires more time for training and prediction data and more storage space. So it is a tradeoff between performance and time cost.

V. Conclusions

References

[1] K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[2] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 115–124.

[3] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu, "Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[4] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[5] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1, pp. 43–52, 2010.

[6] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma, "Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 993–1002.

[7] S. Khandagale, H. Xiao, and R. Babbar, "Bonsai: diverse and shallow trees for extreme multi-label classification," *Mach. Learn.*, vol. 109, no. 11, pp. 2099–2119, 2020. [Online]. Available: https://doi.org/10.1007/s10994-020-05888-2

[8] H. Yu, K. Zhong, and I. S. Dhillon, "PECOS: prediction for enormous and correlated output spaces," *CoRR*, vol. abs/2010.05878, 2020. [Online]. Available: https://arxiv.org/abs/2010.05878

[9] R. Babbar and B. Schölkopf, "Dismec: Distributed sparse machines for extreme multi-label classification," in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 721–729.

[10] ——, "Data scarcity, robustness and extreme multi-label classification," *Machine Learning*, vol. 108, no. 8, pp. 1329–1351, 2019.

[11] I. E. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing, "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*, 2017, pp. 545–553.

[12] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon, "Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification," in *International conference on machine learning*. PMLR, 2016, pp. 3069–3077.

[13] Y. Prabhu and M. Varma, "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*, 2014, pp. 263–272.

[14] J. Weston, A. Makadia, and H. Yee, "Label partitioning for sublinear ranking," in *International conference on machine learning*. PMLR, 2013, pp. 181–189.

[15] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, "Sparse local embeddings for extreme multi-label classification," *Advances in neural information processing systems*, vol. 28, 2015.

[16] H.-F. Yu, P. Jain, P. Kar, and I. Dhillon, "Large-scale multi-label learning with missing labels," in *International conference on machine learning*. PMLR, 2014, pp. 593–601.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[18] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. S. Dhillon, "Taming pretrained transformers for extreme multi-label text classification," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3163–3171.

[19] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, and I. Dhillon, "X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers," *arXiv preprint arXiv:1905.02331*, 2019.

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[21] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021.

TABLE III
PERFORMANCE OF VARIANT NUMBER OF TREES IN LIGHTATTENTION

	Amazon-670K			EUR-Lex			Wiki10-31K		
Trees	P@1	P@2	P@5	P@1	P@2	P@5	P@1	P@2	P@5
1	82.95	65.18	54.56	44.35	36.15	36.15	80.57	72.18	59.61
2	84.21	68.15	58.91	45.15	39.51	37.02	82.51	73.5	60.58
3	85.13	69.18	60.14	46.89	41.1	37.24	84.6	75.31	65.31
4	**86.53**	**71.53**	**61.92**	**48.6**	**43.21**	**38.54**	**86.5**	**77.6**	**68.35**

[22] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma, “The extreme classification repository: Multi-label datasets and code,” 2016. [Online]. Available: http://manikvarma.org/downloads/XC/XMLRepository.html

[23] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, and I. Androutsopoulos, “Large-scale multi-label text classification on EU legislation,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 6314–6322. [Online]. Available: https://www.aclweb.org/anthology/P19-1636

[24] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, “Sparse local embeddings for extreme multi-label classification,” in *Advances in neural information processing systems*, 2015, pp. 730–738.

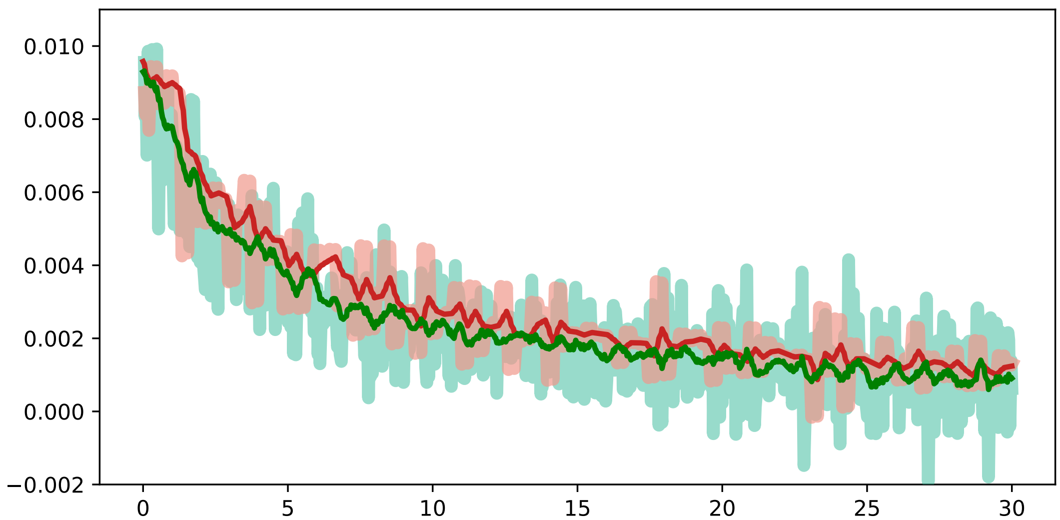

Fig. 4. Train Loss Between LightAttention (Green) and AttentionXML (Red)