LightAttention: Extreme Multi-Label text classification

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 [2], has received extensive research attention due to the rapid development of machine learning methods recently. Compared to the traditional statistical NLP [3], deep neural network [4] 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 [5] [6], [7]. Due to recent development of the Internet, the application scenarios of MTC are prevalent in our daily life, such as item categorization [8] in Taobao, news annotation [9], recommendation system [10], website tagging [11], and so on. The objective of XMTC is to give suitable labels to a give texts or words from an extremely large-scale label set [12]. It is more complicated than multi-class classification [13], 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 [14]. This type of strategy requires extra effort to extra semantic features such as bag-of-words (BOW) [15] from the input texts. (2) Directly use the raw input text as the training dataset [16]. 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 [7], [17]–[19] have been presented by using label partitions or probabilistic label trees (PLTs) strategies that can exploit the relations within labels. Among which, Parabel [17] 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 [20], which further decreases the classification accuracy for tail labels. AttentionXML [7] 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 relevant 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 [21]. 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 cooperative networks that can dynamically generate negative labels. With this dynamic negative label sampling method, we are able to generate quite a series of both positive 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 [22], multi-label attention layer [?], [23], 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 [24], and AmazonCat-13K [25].

Overall, the contributions of this project are as follows:

1. A improvement deep learning framework combines generative cooperative networks and BiLSTM.

2. The generative cooperative networks(GCN) contains label generator and discriminator. The two parts enhances the model's ability to distinguish the negative labels and positive labels. In addition, the GCN avoids overfitting problems.

3. LightAttention experiments show that it improves the accuracy compared with LightXML.

II. Related works

Many works have achieved better performance on XMTC tasks. 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) [26] is the first work to attempts scaling up one-to-one paradigm in extreme multi-label classification problems by parallel training speed-up. ProXML [27] modified DiSEMC and focus on long-tail label prediction. Similarity, PPDSparse [28] and PD-Sparse [29] 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 uses ranking loss function to modify the classifier [30]. Particularly,it constructs a hierarchy structure over the feature space. Similarity, the label partition can use Gini index to evaluate the performance [31]. The Parabel [17] 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 classifiers.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 matrix for the label representation in order to a low-dimensional search for label similarity.For example, SLEEC (Sparse Local Embedding for Extreme Classification) [32] embeds labels onto low dimensional space in order to get low dimensional and dense martix. Typically, SLEEC uses k-nearest neighbor clusters labels into small groups. In addition, The [33] 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 [28]. The reason for this problem should be the embedding representation structure.

d) Deep Learning Approach: As neural network architectures have evolved, many deep learning models have shown better improvements in XMC problems. XML-CNN [6] is the first work to implement deep neural networks on XMC. It gets text representations by forwarding training with CNN network. XML-CNN has s a hidden layer for projecting text features into a low-dimensional space to reduce the computational size. However, unlike the basic multi-label classification, XML-CNN utilizes a simple fully-connected layer to score all labels with binary entropy loss, making it difficult to handle large label sets. AttentionXML [7] uses a probabilistic label tree (PLT) that can handle millions of labels, rather than a simple fully-connected layer, to perform label scoring.

e) Transformer Approach: The NLP proposes a new concept: pre-training then fine-tuning. BERT [34] is the pioneer work 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 [35] 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 [36] learns the label representations from the label and the input text. The process for fine-tuning BERT to exploit the contextual relations between input text. Besides, the generated label clusters is the important component of X-BERT. The best final model is trained on multiple BERT heterogeneous clusters.

III. Methodology

A. Problem Formulation and Overview

The XMTC task is to find multiple relevant labels for each given raw text. The training set $\{x_i, y_i\}_{i=1}^{N}$, where x_i is the origin source, $y_i \in \{0,1\}^L$ 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 whole structure of the LightAttention is listed in Figure 1.

B. Building a PLT for label clustering

For the PLT construction, we follow the method shown in AttentionXML [7], which constructs a shallow and wide PLT from the original deep PLT presented in Parabel [17]. 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 [17], 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 $l \in [1, H]$ do
    if $l < H$ then
        $S_l \leftarrow \{c$-th ancestor node $n$ of nodes in $S_{l-1}\}$
    else
        $S_l \leftarrow \{$the root of $T_0\}$
    end if
    $T_l \leftarrow T_{l-1}$
    for nodes $n \in S_l$ do
        for nodes $n' \in S_{l-1}$ and node $n$ is the ancestor
        of $n'$ in T do
            $Pa(n') \leftarrow n$
        end for
    end for
end for
return $T_H$
```

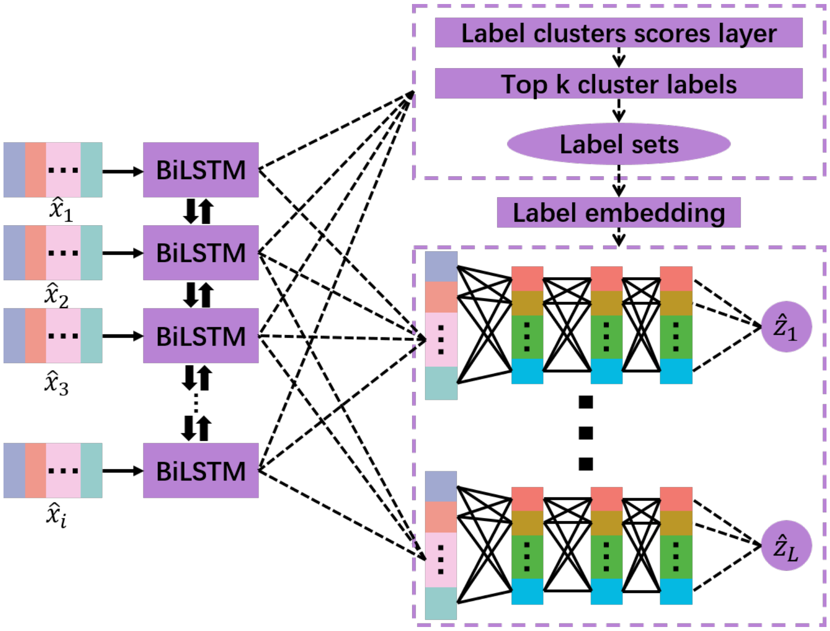

Fig. 1. The overall structure of LightAttention

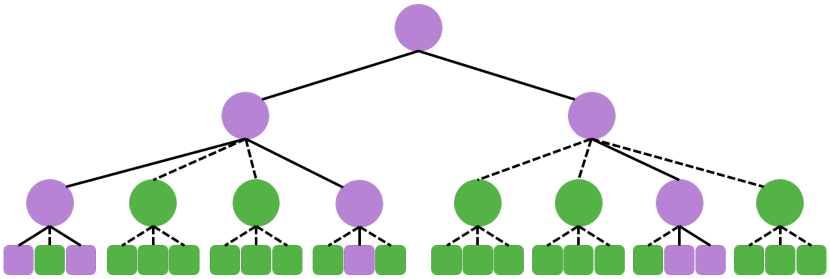

Fig. 2. Shallow and wide probabilistic label tree for label clustering

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 presents a level-wise and top-down manner for each level of nodes. They define AttentionXML$_d$ 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 AttentionXML$_{d-1}$. Then, the children nodes of the top k nodes at the $(d-1)$-level is the training candidate $g(x)$. AttentionXML$_1$ 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 [7] 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.

Generator. 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, during which $y_g \in 0, 1^K$ represents the multi-hot label representation of label cluster for x.

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

Instead of training each level of nodes separately as AttentionXML, we treat the whole label clusters as the labels we sample. The generator needs to regenerate negative labels for the same training texts, so that the discriminator can better distinguish positive and negative labels. The label candidates generated by the generator in LightAttention is generated as Equation 3:

$$S_g = \{l_i : i \in \{i : g_c(l_i) \in G(e)\}\} \tag{3}$$

where g_c helps to map labels to its clusters and l_i denotes the i-th label. During the training phrase, all positive labels are added to S_g as the whole label clusters.

Discriminator. After we obtain the label candidates S_g containing both positive and negative labels, we have to convert the label candidates into label embedding M for each label in S_g. This is denoted as Equation 4.

$$M = [E_i : i \in \{i : S_g\}] \tag{4}$$

where $E_i \in R^b$ is the modifiable b dimension embedding of i-th label and $E \in R^{L \times b}$ is the embedding matrix storing all label embedding. This matrix is initialized randomly and will be updated repeatedly during the training procedure.

$$D(e, M) = \sigma\left(M\sigma\left(W_h e + b_h\right)\right) \tag{5}$$

The discriminator, as shown in Equation 5, is deployed into a hidden bottleneck layer in the deep model. During Equation 5, $W_h \in R^{b \times 5k}$ and $b_h \in R^b$ are the weight of the hidden bottleneck layer. The goal of the discriminator $D(e, M)$ is to detect the difference between positive and negative labels generated by the generator. The training target of $D(e, M)$ is y_d where $y_d^i = 0$ if S_g^i is a positive label and $y_d^i = 1$ if S_g^i is a negative label. Therefore, the loss for discriminator phrase can be computed as Equation 6.

$$\mathcal{L}_d \left(D(e, M), y_d\right) = \sum_{i=0}^{K} \left(1 - y_d^i\right) \left(-\log\left(1 - D(e, M)^i\right)\right) + y_d^i \left(-\log\left(D(e, M)^i\right)\right) \tag{6}$$

The total loss function of the whole training model is the combination of the generator and discriminator, as shown in Equation 7.

$$\mathcal{L} = \mathcal{L}_g + \mathcal{L}_d \tag{7}$$

The whole procedures of generative cooperative netowrks are shown in algorithm 2.

ALGORITHM 2: Dynamic negative labels generator 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 $C$ generation 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 $M$
```

D. Attention-Aware deep model

As shown in Figure 1,the deep model of LightAttention can be briefly divided into six layers: text representation section, BiLSTM section, multi attention setction, fully connected layer, hidden bottleneck 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 [37] 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 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.

3) Multi-label attention: Attention mechanism [38] assigns the most highest weights by a weighted combination of all encoded input vectors. In XMTC, the most relevant context that each label captures would be totally different. We adopts the multi-label attention shown in

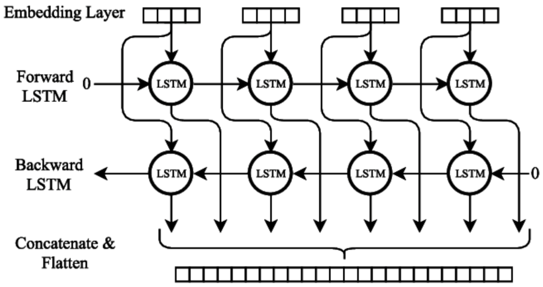

Fig. 3. Structure of BiLSTM

AttentionXML, which detect the intensive parts of text for multiple labels by computing:

$$\hat{\mathbf{m}}_j = \sum_{i=1}^{\hat{T}} \alpha_{ij} \hat{\mathbf{h}}_i, \quad \alpha_{ij} = \frac{e^{\hat{\mathbf{h}}_i \hat{\mathbf{w}}_j}}{\sum_{t=1}^{\hat{T}} e^{\hat{\mathbf{h}}_t \hat{\mathbf{w}}_j}} \tag{8}$$

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 contains two fully connected layers and one output layer.

5) Hidden bottleneck layer: We conduct the hidden bottleneck layer between the fully connected layer and output layer as in [6]. The hidden bottleneck layer also enables the generator and discriminator to concentrate variance of text representations. The generator and the discriminator focuses on fine-grained textual information coarse-grained textual information respectively.

IV. Implementation and Results

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

A. Dataset

Extreme multi-label task is a changeling NLP problem that aims to learn a classifier that can predict text label in an extremely set of labels set. 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 open source datasets (range from 4K to 30K): EUR-Lex, Amazon-670K and Wiki10-31K (shown in I).In this table, E stands for the number of epoch; B presents the batch size; $\bar{N}_{fc}$ shows the hidden unit size of the LSTM; $\bar{N}$ stand for the hidden unit size of fully connected layers; H stands for The height of PLT (excluding the root and leaves);M shows the maximum cluster size; K: The parameters of the compress process, and here we set $M = K = 2c$; And finally C demostrates the number of parents of candidate nodes.

a) Amazon-670K: Amazon-670K [25] 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 [39] proposed EUR-Lex dataset. The Eur-Lex dataset is suitable for LMTC task and zero-short/few-short learning, this allowed user to bypass BERT's 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 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

TABLE I
Hyperparameters used in experiments, practical computation time and model size

Datasets	E	B	$\bar{N}$	$\bar{N}_{fc}$	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

TABLE II
Result comparisons of LightAttention and AttentionXML over three benchmarks.

Dataset	P@1=N@1	P@3	P@5	Model
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

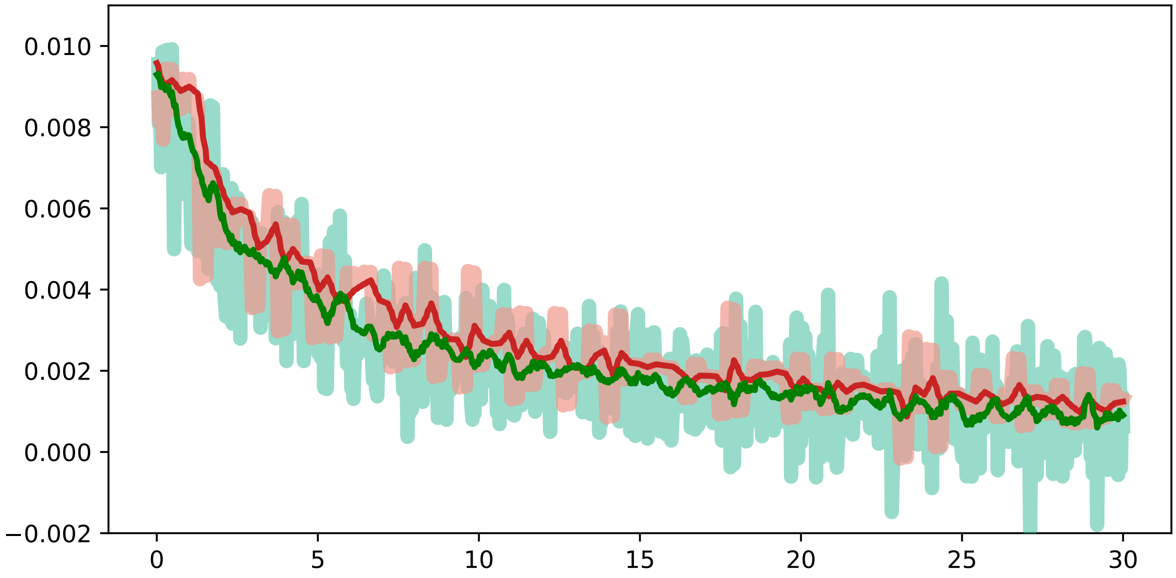

Fig. 4. Train Loss Between LightAttention (Green) and AttentionXML (Red)

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 9).

$$P@k = \frac{1}{k}\sum_{l=1}^{k} y_{rank(l)} \tag{9}$$

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

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. 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 experiments, the most frequent token 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, the fine-tune of word embeddings process is conducted in 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.

D. Performance comparison

LightAttention is based on the evolution of AttentionXML [40] and add Label Embedding Structure from LightXML [41], so here we focus on comparing the two algorithms. First we tested the experimental results on three different datasets (Amazon-670K, EUR-Lex and Wiki10-31k) and compared the loss functions of the training process of the two codes. That is, we demonstrate the superiority of the two algorithms in terms of both process and results.

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 Impact of PLTs

Since the number of PLTs can substantially affect the accuracy of the algorithm, in this section, we tested the effect of the number of PLTs on LightAttention, and the specific experimental results are shown below. We tested the influence of LightAttention with different number of PLTs on the model performance through extensive experiments, as shown in Table III 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.

F. Computation Time and Model Size

The efficiency of the model calculation depends on the performance of the GPU, and the server configuration we use is shown below:

CPU:E5-2670; GPU: 3070; Computer system: Ubuntu 18.04.

Computation time and model memory are crucial for NLP models. Low time and less memory represent a task for NLP models not only for high accuracy but also for efficiency. The following content will give detail information about the computation time and model size of LightAtteition. For AttentionXML, it uses a large neural network model and high-dimensional linear classification that causes a large model size and high computational complexity. We did not compare LightAttention with AttentionXML because AttentionXML is difficult to replicate and has poor efficiency.

As shown in Table I, with different batchsize and different dataset, will yield different models as well as time. The longest dataset is AmazonCat-13K, with a training time of 20.5 hours and a testing time of 1.53 hours, and the largest model is 0.95GB.

V. Conclusions

In this project, we constructed a generative cooperative network for XMC tasks. The project is inspired by the prior work, namely LightXML. LightXML uses pre-training language model, transformer, and dynamic negative label sampling. Our LightAttention framework utilize Generative Cooperative Network (GCN) to solve the negative sample problems by label generator, discriminator. The label generator aims to generates negative labels and feeds the the negative samples to the discriminator to learn the difference.Then, the discriminator can achieve better label representation with the help of large dynamic negative samples. Compared with LightXML, LightAttention improves accuracy in EUR-Lex. In addtion, the model complexity and training time are less than the LightXML. Furthermore, the lightAttention architecture can be tested on more datasets, like Amazon-670K and Wiki-500K.

References

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

[2] P. H. Winston, Artificial intelligence. Addison-Wesley Longman Publishing Co., Inc., 1992.

[3] C. Manning and H. Schutze, Foundations of statistical natural language processing. MIT press, 1999.

[4] A. Torfi, R. A. Shirvani, Y. Keneshloo, N. Tavaf, and E. A. Fox, “Natural language processing advancements by deep learning: A survey,” arXiv preprint arXiv:2003.01200, 2020.

[5] J. Nam, J. Kim, E. Loza Mencía, I. Gurevych, and J. Fürnkranz, “Large-scale multi-label text classification—revisiting neural networks,” in Joint european conference on machine learning and knowledge discovery in databases. Springer, 2014, pp. 437–452.

[6] 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.

[7] 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.

[8] I. H. Bernstein and G. Teng, “Factoring items and factoring scales are different: spurious evidence for multidimensionality due to item categorization.” Psychological bulletin, vol. 105, no. 3, p. 467, 1989.

[9] A. L. Garrido, O. Gomez, S. Ilarri, and E. Mena, “Nass: news annotation semantic system,” in 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence. IEEE, 2011, pp. 904–905.

[10] A. Da’u and N. Salim, “Recommendation system based on deep learning methods: a systematic review and new directions,” Artificial Intelligence Review, vol. 53, no. 4, pp. 2709–2748, 2020.

[11] J. Salminen, V. Yoganathan, J. Corporan, B. J. Jansen, and S.-G. Jung, “Machine learning approach to auto-tagging online content for content marketing efficiency: A comparative analysis between methods and content type,” Journal of Business Research, vol. 101, pp. 203–217, 2019.

[12] D. Ghadiyaram, D. Tran, and D. Mahajan, “Large-scale weakly-supervised pre-training for video action recognition,” in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 12 046–12 055.

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

[14] Q. Dou, D. Coelho de Castro, K. Kamnitsas, and B. Glocker, “Domain generalization via model-agnostic learning of semantic features,” Advances in Neural Information Processing Systems, vol. 32, 2019.

[15] 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.

[16] T. Boroş, Ş. D. Dumitrescu, and R. Burtica, “Nlp-cube: End-to-end raw text processing with neural networks,” in Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, 2018, pp. 171–179.

[17] 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.

[18] 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

[19] 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

[20] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T.-S. Chua, "Reinforced negative sampling over knowledge graph for recommendation," in Proceedings of The Web Conference 2020, 2020, pp. 99–109.

[21] D. W. Otter, J. R. Medina, and J. K. Kalita, "A survey of the usages of deep learning for natural language processing," IEEE transactions on neural networks and learning systems, vol. 32, no. 2, pp. 604–624, 2020.

[22] T. Chen, R. Xu, Y. He, and X. Wang, "Improving sentiment analysis via sentence type classification using bilstm-crf and cnn," Expert Systems with Applications, vol. 72, pp. 221–230, 2017.

[23] T. Baumel, J. Nassour-Kassis, R. Cohen, M. Elhadad, and N. Elhadad, "Multi-label classification of patient notes: case study on icd code assignment," in Workshops at the thirty-second AAAI conference on artificial intelligence, 2018.

[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.

[25] 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

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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

[40] 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," in Advances in Neural Information Processing Systems, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/9e6a921fbc428b5638b3986e365d4f21-Paper.pdf

[41] T. Jiang, D. Wang, L. Sun, H. Yang, Z. Zhao, and F. Zhuang, "Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification," CoRR, vol. abs/2101.03305, 2021. [Online]. Available: https://arxiv.org/abs/2101.03305