Extreme Multi-label Text Classification with Multi-layer Experts

Anonymous ACL submission

Abstract

Extreme multi-label text classification (XMTC) is the task of tagging each document with the relevant labels from a very large space of predefined categories, which presents an open challenge in the recent development of neural classifiers. Popular Transformer-based XMTC methods typically use the last-layer features to represent the document and to match it against candidate labels. We argue that the last-layer features may not be sufficient for predicting labels at different levels of semantic granularity, and that multi-layer features may offer a better choice instead. Based on this insight we propose a novel multi-expert model, namely ME-XML (Multiple Experts for XMTC), which combines multi-layer embeddings in Transformer for improving the prediction power of the model. Our experiments show that ME-XML outperforms the state-of-the-art methods on two out of three datasets, in the predictions over both head (common) labels and tail (rare) labels.

1 Introduction

Extreme multi-label text classification(XMTC) is the task to assign a set of relevant labels to each input text instance, where the number of candidate labels can reach tens of thousands or over a million. With such an enormous label space, severe data sparse issue is one of the main challenges as most of the label has a very few training instances. For example, in the Wiki10-31k dataset, more than 90% of labels have less than 20 training instances. XMTC has many real-world applications, such as topic spotting for Wikipedia articles, news stories, and academic publications, and tagging products for advertising.

With the resent success of large-scale pre-trained language models (Peters et al., 2018; Devlin et al., 2018; Yang et al., 2019; Liu et al., 2019a; Raffel et al., 2020) in neural network research, modern XMTC models employs pre-trained Transformers to extract latent features for input documents and then train the classification models based on the extracted features. Typically, the features in the last layer of a Transformer are used to represent the whole input document because they tend to capture the richest abstract semantic information of the text.

However, we argue that the different labels in XMTC can reflect the semantic contents of a document at various granularity levels, and that using only the last-layer features may not be the best design choice for classification modeling. For some labels, a keyword or key phrase may be sufficient to classify the concept, and such keywords or key phrases may be best captured by the latent features in some early or middle layers of the Transformer, instead of the last layer. For example, a model can easily classify the "electronic" category by simply detecting the keyword "computer" or "hardware" in the text without knowing a higher-level abstraction of the content in the whole document.

In this paper, we investigate how to improve the XMTC prediction power by using features from multiple Transformer layers, which may represent the semantic information at various granularity levels. A simple approach is just to concatenate or pool over the embeddings from several layers (Sun et al., 2019; Jiang et al., 2021). However, as observed by prior work (Sun et al., 2019) and by our own analysis in Section. 4.4, such a simplistic treatment has a limited success as it does not have the capability to discriminate which layers are more important for different labels, and cannot dynamically decide which layers should be relied on more than other layers given an input instance.

As a remedy, we propose ME-XML, a Multiple Experts model for XMTC, where each expert is forced to utilize the embeddings from only one assigned Transformer layer. To better leverage word or phrase level information, the experts on the early layers focus on label-to-word attention (You et al., 2018), where each label directly selects the most important words to form its label-specific document representation. On the other hand, the expert on the last layer simply uses [CLS] embedding to represent the whole document. With different model architecture for different experts, it encourages experts to specialize for different tasks, which can potentially improve the performance of our multiple experts models. Furthermore, we propose a distribution-aware diversity loss which is tailored for XMTC, to encourage the experts to be proficient for labels at different levels of rareness. This is particularly important for effective modeling in XMTC as the label distributions in XMTC are often highly skewed.

Our experiments demonstrate the effectiveness of our proposed method. As an ensemble approach, it enhances the diversity of expert predictions, which is correlated to the model performance improvement. Compared to other Transformer-based ensemble models, such as LightXML (Jiang et al., 2021), which ensembles the predictions by finetuning BERT, RoBERTa, and XL-Net separately, our method is more efficient because the experts share the same Transformer backbone and can be jointly finetuned. With such a lightweight multiple experts ensemble method, we achieve state-of-the-art results on two out of three datasets, in both head labels (common) and tail (rare) labels.

2 Related Work

Description based Text Classification Recently, utilizing keywords information or label description for text classification has attracted the attention of many researchers. This is especially useful for few shot or zero shot (Zhang et al., 2019) text classification where this extra information regularizes the behavior of the models. Prompt-based few shot text classification (Schick and Schütze, 2021; Gao et al., 2021) utilizes label description to extract knowledge from Transformers. Bao et al. (2020) computes the word importance by incorporating word statistic information into neural network training.

For example, the method by Wang et al. (2018) selects the most relevant words for each class and learns label-to-word attention, where the label embeddings are generated based on label descriptions. Chai et al. (2020) generates the label descriptions in an unsupervised manner, and concatenate them with the input text for selecting the most salient part of the text.

Extreme Multi-Label Text Classification Extreme multi-label text classification (XMTC) is different from typical text classification in its enormous label space. The main difficulties in XMTC are regarding how to improve the performance with feasible computational complexity, and how to handle highly skewed label distributions and the associated severe data sparse issues. Before the deep learning era, traditional classifiers use bag-of-words (BoW) features with frequency based weights (such as TF-IDF) as the input features. To reduce the label search space, tree-based methods (Prabhu and Varma, 2014; Khandagale et al., 2019; Prabhu et al., 2018) construct hierarchical label trees that partition the label space into clusters of labels, and then train local classifiers models within the scope of each cluster.

Deep learning methods employ various deep neural networks such as CNN (Kim, 2014), LSTM (Hochreiter and Schmidhuber, 1997), or Transformers (Vaswani et al., 2017) to learn generic features for all labels, which can be further incorporated with tree-based methods by replacing the node classifiers with neural networks. Similar to Wang et al. (2018), Attentional-XML (You et al., 2018) employs LSTM with label-to-word attention along a system-induced hierarchy of labels. X-Transformer (Chang et al., 2020) is a two stages method, which uses XLNet (Yang et al., 2019) to predict the relevant cluster given an input document in the first stage, and then predicts labels within the selected clusterin the second stage. APLC-XLNet (Ye et al., 2020) separates the labels into several groups based on their frequencies, and each group has its own classifier with the dimension of label embeddings proportional to the label frequencies. LightXML (Jiang et al., 2021) is a method similar to X-Transformer and Attentional-XML in terms of using a label hierarchy to decouple the problem into sub-problems, but finetunes Transformers in an end2end manner instead of a two-stage training.

Ensemble Learning Different models can specialize in different types of tasks. The goal of ensemble learning is to get a more accurate prediction by combining the predictions from many models. Mixtures of experts models (Jacobs et al., 1991) modularize a huge cumbersome network by decomposing it into several expert networks, where each network is trained on a subset of a training corpus. The experts are encouraged to acquire diverse ex-

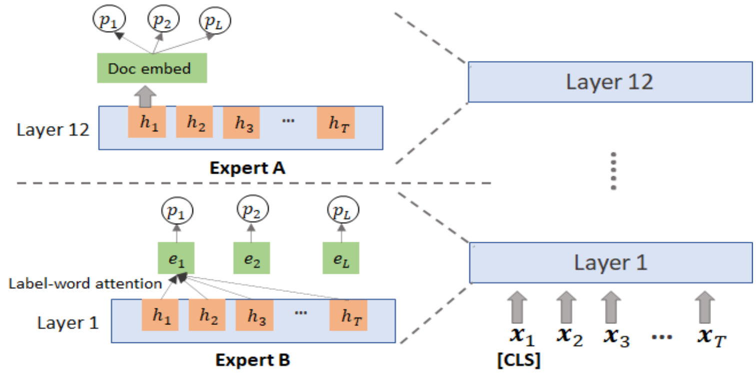


Figure 1: The framework of the proposed method. Here, we consider two experts setting in which the experts share the same Transformer model with 12 layers. Expert 1 extracts the embedding of [CLS] tag as its document representation. Expert 2 employs a label to word attention module on the first layer of the Transformer.

pertise to make a more robust prediction (Shazeer et al., 2017). Knowledge distillation (Hinton et al., 2015) is widely applied to transfer knowledge from multiple networks to a smaller network.

Ensemble learning demonstrates its effectiveness for tail label prediction, which is an important task in XMTC due to the label sparsity issue. BBN (Zhou et al., 2020) splits a network into bilateral branches, one for learning tail label features and another one for head label, of which predictions are ensembled into a single prediction. LFME (Xiang et al., 2020) transfers knowledge from multiple expert networks to a student network with curriculum learning. RIDE (Wang et al., 2021) is trained with a distribution-aware diversity loss to encourage the diversity of experts on tail label prediction.

3 Proposed Method

3.1 Overall framework

In Figure 1, we only consider a scenario with two experts, named Expert 1 and expert 2. Expert 1 has a typical architecture for text classification, which uses the representation of a "[CLS]" tag from the last layer to represent a whole document. Expert 2 uses a label to word attention on an early Transformer layer, which selects the most important words to form a label-specific document representation.

The experts are trained on binary cross entropy loss individually without collaborating with each other. A distribution-aware diversity loss is applied to encourage the experts to predict diverse tail labels. During inference, each expert predicts a probability of each class, and the probabilities of a class from different experts are combined together with equal weights to produce the final prediction.

3.2 Model Architecture

In this section, we elaborate on the model architecture of experts.

3.2.1 Expert using high-level features

The Expert 1 in Figure 1 utilizes the high-level abstract features of Transformer, which is the classic architecture for text classification. The hidden representations from the m-th Transformer layer is denoted as:

$$\phi^n_{transformer}(x) = \{h_1^{(m)}, h_2^{(m)}, h_3^{(m)}, ..., h_T^{(m)}\}$$

, where T denotes the input sequence length, and $h_1^{(m)}$ denotes the contextualized word embedding of [CLS] tag. Here, as the expert leverages the high-level features, we set $m = 12$, which is the last layer of RoBERTa-base (Liu et al., 2019a) model. Then a multi-layer perceptron (MLP) network $\phi^{(k)}_{MLP} : \mathbb{R}^d \rightarrow \mathbb{R}^L$ projects $h_1^{(m)}$ to L class logits $l_i^{(k)}$:

$$\phi^{(k)}_{MLP}(h_1^{(m)}) = \{l_1^{(k)}, l_2^{(k)}, ..., l_L^{(k)}\}, \qquad (1)$$

, where d is the dimension of contextualized word embedding, k denotes the expert index and L denotes the label number.

3.2.2 Expert using low-level features

In Figure 1, Expert 2 uses the low-level features, thus we set $m = 1$. We found that using contextualized word embedding of [CLS] tag as document representation on early layers yields poor results. Therefore, we explore a model architecture that can directly use these keywords information on low-level features. Label to word attention (Wang et al., 2018; You et al., 2018) is a straightforward option for utilizing such low-level features, which selects the most salient words to form a label-specific document representation.

Specifically, the label to word attention allows each label to interact with each word by an attention mechanism. The attention score α_{ij} of label j to word i can be calculated as:

$$\alpha_{ij} = \frac{e^{\mathbf{h}_i \mathbf{w}_j}}{\sum_{t=1}^{T} e^{\mathbf{h}_t \mathbf{w}_j}} \tag{2}$$

, where w_j denotes the label embedding for j-th label. The attention score α_{ij} can be interpreted as the importance of word i to label j. Then, the label-specific document embedding e_j can be calculated as:

$$e_j = \sum_{i=1}^{T} \alpha_{ij} \mathbf{h}_i \tag{3}$$

Finally, the logit $l_j^{(k)}$ for label j is calculated by:

$$l_j^{(k)} = \phi_{MLP}^{(k)}(e_j) \tag{4}$$

where $\phi_{MLP}^{(k)}(e_j) : \mathbb{R}^d \rightarrow \mathbb{R}$ is the multi-layer perceptron function that summarizes a label specific document feature into a real-valued logtit. The *MLP* function is shared by all labels.

3.3 Training

We use individual expert loss and distribution-aware diversity loss to train our multiple experts model. In the following sections, we elaborate on these two losses respectively.

3.3.1 Individual Loss

As multiple experts models aim at improving the performance of the final prediction combined from individual experts, one intuitive method is making experts collaborate with each other to make a better final prediction. Collaborative loss (Zhou et al., 2020; Xiang et al., 2020) aggregates the logits from multiple experts, and then the aggregated logits are used to optimize the objective function. Specifically, the collaborative loss for an input x and its ground truth labels y can be written as:

$$\mathcal{L}_{collaborative}(x, y) = \mathcal{L}(\frac{1}{K}\sum_{i=1}^{K}(f_{\theta_i}(x)), y) \tag{5}$$

, where $f_{\theta_i}(x) = \{l_1^i, l_2^i, ..., l_C^i\}$ denotes the output logits of expert i and K denotes the number of experts. We use binary cross entropy loss with logits as $\mathcal{L}$. However, Wang et al. (2021) found that collaborative loss hiders the experts from making complementary or diverse predictions. This phenomenon is also found in our experiment that the collaborative loss makes the final prediction relies on the expert using high-level features because the high-level features can better fit the objective. Therefore, we use the individual loss calculated as:

$$\mathcal{L}_{individual}(x, y) = \frac{1}{K}\sum_{i=1}^{K}\mathcal{L}((f_{\theta_i}(x)), y)$$

With this loss, it forces each expert to make a good prediction independently, and thus enhance the diversity of expert predictions.

3.3.2 Distribution-aware Diversity Loss

As shown in the Table 1, in XMTC datasets, the average training instances for most of the labels are very few, especially for Wiki10-31K dataset. Thus, predicting tail labels is an important direction for improving XMTC. To tackle tail label prediction, Wang et al. (2021) proposed distribution-aware diversity loss which maximizes the KL-divergence between experts' classification probability distributions with a focus on tail labels.

However, their diversity loss is designed for single label classification task rather than multi-label classification task, and thus we cannot directly apply their loss to our task. Also, we argue that directly maximizes the KL divergence between experts' predictions is not reasonable. Considering a scenario that all the experts can perfectly predict the correct labels for an input text, then the experts' output label distributions will be close to each other and thus the KL-divergence will be small. In this scenario, maximizing KL divergence discourages the experts to make correct prediction.

To fix this, we propose a distribution-aware diversity loss tailored for the XMTC. For an input x, the diversity loss between predictions from an expert j and an expert k is calculated as:

$$\mathcal{L}_{diversity}(x, y) = -\sum_{i=1}^{C}\lambda_i D_{KL}(p_i^{(j)}||p_i^{(k)}) \tag{6}$$

$$D_{KL}(p_i^{(j)}||p_i^{(k)}) = p_i^{(j)}\log(\frac{p_i^{(j)}}{p_i^{(k)}}) \tag{7}$$

$$+ (1 - p_i^{(j)})\log(\frac{1 - p_i^{(j)}}{1 - p_i^{(k)}}) \tag{8}$$

Dataset	N_{train}	N_{test}	$\lvert \bar{y_n} \rvert$	L	$\bar{n_l}$	$\lvert L_{few} \rvert$	$\lvert L_{med} \rvert$	$\lvert L_{many} \rvert$
EURLex-4K	15,539	3,809	5.30	3,956	20.84	3,133	2,978	183
Wiki10-31K	14,146	6,616	18.64	30,938	18.64	29,309	1,321	308
AmazonCat-13K	1,186,239	306,782	5.04	13,330	448.57	5,875	3,889	3,566

Table 1: Datasets statistics. N_{train} and N_{test} refer to the number of training and testing instances respectively. $|\bar{y_n}|$ is the average number of labels per instance. L is the number of labels. $|L_{few}|$, $|L_{med}|$ and $|L_{many}|$ denote the number of labels belongs to few-shot(≤ 20) /medium-shot(≤ 100 & > 20) /many-shot(> 100) classes respectively.

, where $p_i^{(j)} = sigmoid(l_i)$ is the probability for label i from expert j. The KL-divergence is in this form because the distributions for each label are independent Bernoulli distributions. The term $\lambda_i \in [0, 1]$ controls the extent to which the KL divergence of the label i is maximized :

$$\lambda_i = \begin{cases} 0 & \text{if } i \notin y \\ \frac{1 - SG(max(p_i^{(j)}, p_i^{(k)}))}{n_i} & \text{else} \end{cases} \quad (9)$$

, where $SG(\cdot)$ means the gradient stop operator and n_i is the number of training instances for label i. For tail labels, λ_i is larger, which encourages the experts to make diverse predictions on tail labels. If the label i is not in the ground truth label set y for an input x, then its KL-divergence will not be maximized because maximizing the KL-divergence makes the model predict irrelevant labels. If any expert can successfully predict the ground truth label $i \in y$, then the λ_i will be smaller because we don't want the loss to hinder experts from making a correct prediction. The diversity loss is large only when both experts cannot correctly predict the label.

Finally, the whole training loss $\mathcal{L}$ is:

$$\mathcal{L} = \mathcal{L}_{individual}(x, y) + \alpha \mathcal{L}_{diversity}(x, y) \quad (10)$$

, where α controls the weight of diversity term. We only apply $\mathcal{L}_{diversity}$ after using $\mathcal{L}_{individual}$ to train a few epochs.

3.4 Inference

During inference, we have tried to train a neural network such as expert routing network (Wang et al., 2021) to assign suitable experts to predict the input text. However, we found that a simple combination that assigns each expert with an equal weight yields better results. The final prediction $g(x)$ of an input x can be written as:

$$g(x) = \frac{1}{K} \sum_{i=1}^{K} sigmoid(f_{\theta_i}(x))$$

, where K denotes the expert number.

4 Experiments

4.1 Experimental Settings

Datasets We conduct experiments on three datasets, which are EUR-LEX (Loza Mencía and Fürnkranz, 2008), Wiki10-31k (Zubiaga, 2012) and AmazonCat-13k (McAuley and Leskovec, 2013). The processed data is obtained from Jiang et al. (2021)[1].

From the data statistics in Table 1, we can find that the average number of training instances $\bar{n_l}$ for the labels in EURLex-4K and Wiki10 are very few, which is less than 21. While in AmazonCat-13K, each labels has 448 training instances in average. To better understand the data statistics, we follow the literature of long tail classification (Liu et al., 2019b), dividing the labels into three classes, which are few-shot, medium-shot, and many-shot. In the few-shot class, all the labels have less or equal to 20 training instances ($n_l \leq 20$). The medium-shot class has $20 < n_l \leq 100$, and the many-shot class has $100 < n_l$. More than 90% of labels in Wiki10-31K belongs to the few-shot class, which highlights the severe data sparsity issue in this dataset.

4.2 Comparison with Prior Methods

Evaluation Metric In this section, we use the *micro-averaging* P@k (precision@k), the most widely used metric in XMTC, to evaluate our results. P@k evaluates the accuracy on the top k ranked labels. Note that micro-averaging P@k is averaged over instances, and thus its performance is dominated by the high-frequency labels. Specifically, P@k is calculated as:

$$\frac{1}{N} \sum_{n=1}^{N} \frac{1}{k} \sum_{l=1}^{k} y_{rank(l)} \quad (11)$$

, where N is the number of testing instances, and $y_{rank(l^{(n)})} \in \{0, 1\}^C$ is 1 if $rank(l^{(n)}) \in y^{(n)}$ otherwise 0, and $rank(l^{(n)})$ is the index of the l-th highest predicted label of the n-th instance.

[1]https://github.com/kongds/LightXML

	EUR-LEX			Wiki10-31k			AmazonCat-13k		
Methods	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
SVM	86.43	73.94	62.43	85.16	75.34	66.87	94.21	79.80	64.78
Parabel	82.12	68.91	57.89	84.19	72.46	63.37	93.02	79.14	64.51
Bonsai	82.30	69.55	58.35	84.52	73.76	64.69	92.98	79.13	64.46
XML-CNN	76.38	62.81	51.41	81.41	66.23	56.11	93.26	77.06	61.40
AttnXML	85.49	73.08	61.10	87.05	77.78	68.78	95.65	81.93	66.90
X-Transformer-E	87.22	75.12	62.90	88.51	78.71	69.62	96.70	83.85	68.58
X-Transformer	85.46	72.87	60.79	87.12	76.51	66.69	95.75	82.46	67.22
APLC-XLNet	87.72	74.56	62.28	89.44	78.93	69.73	94.56	79.82	64.60
LightXML-E	87.63	75.89	63.36	89.45	78.96	69.85	**96.77**	**84.02**	**68.70**
LightXML	87.56	74.31	62.14	88.55	78.48	68.87	95.21	81.01	65.93
ME-XML (1 expert)	87.11	74.53	62.33	88.61	78.53	68.92	95.82	82.68	67.42
ME-XML (2 experts)	**89.43**	**77.12**	**64.38**	**90.32**	**80.10**	**71.51**	**96.01**	**82.71**	**67.45**
ME-XML (3 experts)	89.38	**77.63**	**64.85**	90.35	80.22	71.58	95.98	82.78	67.55

Table 2: Comparison with different methods using micro-averaging P@k. The method ending with "-E" denotes the model as an ensemble model. Compared with Transformer-based methods, ME-XML improves the results on two out of three datasets.

Settings The implementation details are in Appendix A. In Table 2, the ME-XML (1 expert) refers to the expert using the last layer features described in Section 3.2.1, which is a vanilla Transformer-based classifier. The ME-XML (2 experts) refers to the two experts model described in Section 3, in which one expert uses the last layer features and another expert employs label to word attention on the features of the first layer. ME-XML(3 experts) is based on ME-XML (2 experts) with an extra expert using label to word attention on the sixth layer of Transformer.

Baselines In Table 2, the non-neural baselines include SVM (Chang and Lin, 2011), Bonsai (Khandagale et al., 2019) and Parabel (Prabhu et al., 2018). For the neural network baselines, there are XML-CNN (Liu et al., 2017) and AttentionXML (You et al., 2018). Our main baselines are Transformer based methods, including X-Transformer (Chang et al., 2020), APLC-XLNet (Ye et al., 2020) and LightXML (Jiang et al., 2021). X-Transformer-E and LightXML-E denote they are ensemble models rather than a single model.

Results ME-XML outperforms all other single model baselines. ME-XML shows significant improvements on EUR-LEX and Wiki10-31k datasets, and only slightly improve the results on AmazonCat-13k. Compared with Transformer-based ensemble methods, ME-XML still performs better on EUR-LEX and Wiki10-31k datasets, but it doesn't improve the performance on the AmazonCat-13k dataset. Note that although ME-XML is an ensemble method, the Transformer backbone of all the experts can be jointly finetuned rather than finetuning on several Transformer models, like LightXML-E ensembles the predictions by finetuning BERT, RoBERTa and XL-Net separately.

Analysis LightXML also leverages the representations from multiple Transformer layers by concatenating the embeddings of [CLS] from several layers. The superior single model performance of ME-XML shows that the multiple experts training is a better method to utilize representations from multiple layers. In addition, we can observe that ME-XML is more effective on EUR-LEX and Wiki10-31k than on AmazonCat-13k. We speculate the main reason is that EUR-LEX and Wiki10-31k have more serious data sparsity issues as shown in Table 1. We speculate that leveraging keyword information or low-level features is more effective when the number of training instances is not sufficient. To support this speculation, we examine the performance on tail labels in the next section.

	EUR-Lex								
	Few-shot			Medium-shot			Many-shot		
Methods	P@5	R@5	F@5	P@5	R@5	F@5	P@5	R@5	F@5
ME-XML (1 expert)	22.08	27.98	22.97	42.96	72.81	52.02	46.06	82.84	58.07
ME-XML (2 experts)	22.57	28.64	23.66	44.08	76.51	54.01	46.15	85.02	58.83
	Wiki10-31k								
	Few-shot			Medium-shot			Many-shot		
Methods	P@5	R@5	F@5	P@5	R@5	F@5	P@5	R@5	F@5
ME-XML (1 expert)	6.32	4.13	4.55	37.94	22.51	26.34	44.65	32.03	35.57
ME-XML (2 experts)	5.68	3.69	4.05	38.94	24.97	27.86	45.87	33.84	36.65
	AmazonCat-13k								
	Few-shot			Medium-shot			Many-shot		
Methods	P@5	R@5	F@5	P@5	R@5	F@5	P@5	R@5	F@5
ME-XML (1 expert)	22.05	26.19	21.02	40.39	69.81	45.89	43.13	72.76	49.95
ME-XML (2 experts)	28.84	31.65	27.24	43.78	69.23	49.21	42.90	72.51	49.91

Table 3: Performance analysis on different frequency classes using macro-averaging metrics.

4.3 Performance on Different Frequency Classes

Evaluation Metrics In this section, we examine the performance of ME-XML on different label frequency classes. Here, we follow paradigm in tail label classification literature (Liu et al., 2019b), which divides the labels into few-shot, medium-shot and many-shot classes. The boundary for few-shot is $(n \leq 20)$, medium-shot is $(20 < n \leq 100)$ and many-shot is $(100 < n)$. The label number for each class on different datasets is shown in Table 1.

Following the tail label evaluation literature, we use *macro-averaging* evaluation metrics to evaluate our results, which is averaged over *labels* rather than *instances*. Specifically, it is calculated as:

$$Metric_{macro} = \frac{1}{|L_{class}|} \sum_{l \in L_{class}} Metric(l) \tag{12}$$

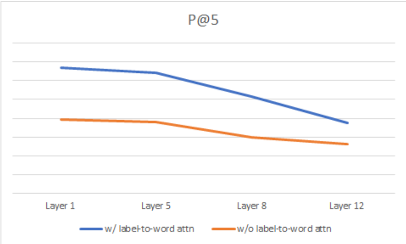
, where L_{class} can be few-shot class L_{few}, medium-shot class L_{medium}, or many-shot class L_{many}. $Metric(l)$ is a metric to evaluate the performance of label l. We use macro- P@5 (precision@5), R@5 (Recall@5) and F@5 (F1-score@5) as our metrics. The details of these metrics are in Appendix B.

Performance Analysis In Table 2 AmazonCat-13k, when using micro-averaging metric, the P@k scores of ME-XML with 2 experts and 1 expert are almost the same. However, in Table 3, by splitting the labels into different frequency classes, we can find that their behavior is actually different. The two experts model performs much better on few-shot and medium-shot classes, which demonstrates its efficacy on tail label prediction. The slightly higher R@5 scores of the single expert model on the Medium-shot and Many-shot classes implies it predicts more head labels in the top 5 ranking lists.

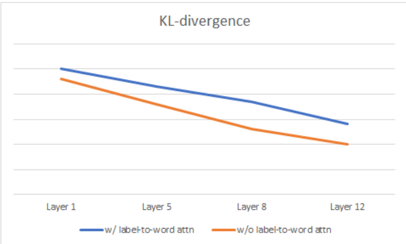
In the EUR-LEX dataset, the two experts model outperforms the single expert model on all classes. In the Wiki10-31k dataset, both methods perform equally poorly on the few-shot class, which means the labels in the few-shot category are almost not possible to be predicted and thus the models tend to not put these tail labels in the top 5 ranking lists. In conclusion, the multiple experts model can improve the performance on few-shot and medium-shot classes except for the almost unpredictable few-shot class of Wiki10-31k.

4.4 Ablation Study

Model Architecture In Table 4, to utiliz multi-layer features, the 1E (D-(12+1)) concatenates the [CLS] embeddings from layer 12 and layer 1. Comparing 1E (D-(12+1)) with 2E(D-12+A-1), we can find that multiple experts training is a more effective way to leverage features from multiple layers.



(a) The micro-averaging P@5



(b) The KL-divergence of the two experts.

Figure 2: Comparison of different model architectures for the second expert on EUR-LEX dataset.

Method	P@1	P@3	P@5
1E (D-12)	87.11	74.53	62.33
1E (A-1)	86.03	71.10	57.75
1E (A-12)	86.14	72.39	59.05
1E (D-(12+1))	87.24	74.67	62.35
2E (A-1+D-12)+$\mathcal{L}_{col}$	88.35	76.12	63.22
2E (A-1+D-12)+$\mathcal{L}_{ind}$	89.13	77.01	64.14
2E (A-1+D-12)+$\mathcal{L}_{ind}$+$\mathcal{L}_{div}$	**89.43**	**77.12**	**64.38**
2E (D-1+D-12)+$\mathcal{L}_{ind}$+$\mathcal{L}_{div}$	87.69	75.48	62.96
2E (A-12+D-12) +$\mathcal{L}_{ind}$+$\mathcal{L}_{div}$	87.77	75.94	63.76
3E(A1+A6+D12)+$\mathcal{L}_{ind}$+$\mathcal{L}_{div}$	89.38	**77.63**	**64.85**
3E(A1+D6+D12)+$\mathcal{L}_{ind}$+$\mathcal{L}_{div}$	88.54	76.15	63.51

Table 4: Ablation Study on EUR-LEX dataset with micro-averaging P@k. #E denotes the number of experts (# is a number). A# denotes one expert uses label to word attention on layer #. D# denotes one expert uses [CLS] as document representation on layer #. $\mathcal{L}_{col}$, $\mathcal{L}_{ind}$, and $\mathcal{L}_{div}$ denote the collaborative loss, individual loss, and diversity loss respectively.

Then, we examine whether using low-level features and word-to-label attention can benefit the performance of multi-experts model. In Figure 2, there are two experts. Expert 1 is fixed, which uses the [CLS] embedding on the last layer as described in Section 3.2.1. We adjust the model architecture of expert 2, and evaluate its performance using micro-based P@5. As shown in Figure 2(a), using label to word architecture on early layer features as the second expert can greatly enhance the performance. Similar observation can also be found in Table 4, by comparing 2E(A-1+D-12) with 2E(D-1+D-12) and 2E(A-12+D-12).

We speculate the main reason is that using the label to word attention on the first layer as expert 2 can generate a complementary prediction for expert 1 to the most extent because their model architectures are the most different. To verify this, we plot the KL-divergence in Eq(6) and set $\lambda_i = 1$ if $i \in y_n$. Greater KL-divergence implies the predictions are more different. Putting Figure 2(a) and Figure 2(b) together, we find that KL-divergence is correlated with the model performance, which supports our motivation to maximize the KL-divergence. The complementary prediction implies that the early layer can capture different levels of semantic granularity.

Training Loss As shown in the Table 4, compared with the collaborative loss $\mathcal{L}_{col}$, the individual loss $\mathcal{L}_{ind}$ performs better. The distribution-aware diversity loss $\mathcal{L}_{div}$ also improves the performance. The improvement of using these two losses highlights the importance of boosting the complementary and diverse predictions from multiple experts.

Expert Number In Table 4, comapred with using 1 expert, using 2 experts greatly enhance the performance. The performance of using 3 experts depends on the model architecture of the third expert. When using label to word attention on the sixth layer of Transformer, it improves the P@3 and P@5 scores. Considering the extra computation of using more experts, using 2 experts is sufficient to get good performance.

5 Conclusion

In this paper, we propose a multiple experts model (ME-XML) for XMTC, in which different experts leverage features from different Transformer layers. By leveraging features from different layers, each expert specializes in different levels of semantic granularity, thus proficient in different types of labels. With label to word attention on the early layer, one expert captures the salient part of texts. To encourage diversification, individual loss and diversity loss are applied to train the experts. With extensive experiments, ME-XML demonstrates its superior performance over other SOTA methods on two out of three datasets. Comparing ME-XML with vanilla Transfermer on labels in different frequency intervals, our method is particularly stronger in tail label prediction, which is harder part of XMTC due to the severe data sparse issue.

References

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. 2020. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*.

Duo Chai, Wei Wu, Qinghong Han, Fei Wu, and Jiwei Li. 2020. Description based text classification with reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1371–1382.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S Dhillon. 2020. Taming pretrained transformers for extreme multi-label text classification. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3163–3171.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87.

Ting Jiang, Deqing Wang, Leilei Sun, Huayi Yang, Zhengyang Zhao, and Fuzhen Zhuang. 2021. Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. *arXiv preprint arXiv:2101.03305*.

Sujay Khandagale, Han Xiao, and Rohit Babbar. 2019. Bonsai – diverse and shallow trees for extreme multi-label classification.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 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*, pages 115–124.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. 2019b. Large-scale long-tailed recognition in an open world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eneldo Loza Mencía and Johannes Fürnkranz. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. In *Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. page 165–172. Association for Computing Machinery.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Yashoteja Prabhu, Anil Kag, Shrutendra Harsola, Rahul Agrawal, and Manik Varma. 2018. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference*, page 993–1002. International World Wide Web Conferences Steering Committee.

Yashoteja Prabhu and Manik Varma. 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*, page 263–272, New York, NY, USA. Association for Computing Machinery.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the*

16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. 2021. Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations*.

Liuyu Xiang, Guiguang Ding, and Jungong Han. 2020. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision*, pages 247–263. Springer.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Hui Ye, Zhiyu Chen, Da-Han Wang, and Brian Davison. 2020. Pretrained generalized autoregressive model with adaptive probabilistic label clusters for extreme multi-label text classification. In *International Conference on Machine Learning*, pages 10809–10819. PMLR.

Ronghui You, Zihan Zhang, Ziye Wang, Suyang Dai, Hiroshi Mamitsuka, and Shanfeng Zhu. 2018. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *arXiv preprint arXiv:1811.01727*.

Jingqing Zhang, Piyawat Lertvittayakumjorn, and Yike Guo. 2019. Integrating semantic knowledge to tackle zero-shot text classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1031–1040, Minneapolis, Minnesota. Association for Computational Linguistics.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. 2020. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. pages 1–8.

Arkaitz Zubiaga. 2012. Enhancing navigation on wikipedia with social tags.

A Implementation Details

Module	EUR-LEX	Wiki	Amazon
Word-label	2e-4	1e-4	5e-4
Output MLP	2e-3	1e-3	2e-3
Transformer	5e-5	1e-5	5e-5

Table 5: Learning rates for different modules on different datasets.

	EUR-LEX	Wiki	Amazon
Batch size	8	5	12
Epochs	8	5	4
T	512	512	192

Table 6: Hyperparameters for different datasets. T denotes the input sequence length.

We choose RoBERTa-base (Liu et al., 2019a) as our Transformer backbone. We use different learning rates for different modules as suggested in APLC-XLNET (Ye et al., 2020). The learning rate for the pre-trained Transformer is set to be smaller because we don't want the embedded semantic information in the Transformer to change too much. The learning rate for the MLP module in Eq.(1) and the learning rate for word-to-label attention in Section 3.2.2 are set to be larger. The learning rates for different modules can be found in Table 5. Other model hyperparameters are listed in Table 6. The α in Eq(10) is set to be 0.01 because we found that if we set α to be a large value, the model only optimizes the $\mathcal{L}_{diversity}$ and neglects the $\mathcal{L}_{individual}$.

B Macro-averaging Metrics

In this section, we introduce macro-averaging P@k, R@k and F@k for a label l. The $P@k^{(l)}$ is calculated as:

$$P@k^{(l)} = \frac{t_k(l)}{p_k(l)} \tag{13}$$

, where $p_k(l)$ denotes the total number of the label l appearing in the predicting top k ranking lists, and $t_k(l)$ denotes the total number of label l is a ground truth label and appearing in the predicting top k ranking lists (also known as True Positives).

The $R@k^{(l)}$ is calculated as:

$$R@k^{(l)} = \frac{t_k(l)}{y(l)} \tag{14}$$

, where $y(l)$ denotes the total number of label l appearing in the ground truth label set.

The F@k (F1-score@k) is calculated as:

$$F@k^{(l)} = 2 \cdot \frac{P@k^{(l)} R@k^{(l)}}{P@k^{(l)} + R@k^{(l)}} \tag{15}$$

The macro-averaging scores for each metric are then averaging over labels as described in Eq(12).