On Evaluation and Improvement of Tail Label Performance for Multi-label Text Classification

Anonymous ACL submission

Abstract

Extreme multi-label text classification (XMTC) is a task for tagging each document with the most relevant subset of labels from an extremely large label set. The most challenging part for machine learning methods is the skewed label distribution in which a majority of labels receive very few training instances (named as the tail labels). Benchmark evaluations so far have focused on micro-averaging metrics, where the performance on tail labels can be easily overshadowed by high-frequency labels (named as head labels), and hence they are insufficient for evaluating the true success of methods in XMTC. This paper presents a re-evaluation of state-of-the-art (SOTA) methods based on the *binned macro-averaging F1* instead, which reveals new insights into the strengths and weaknesses of representative methods. Based on the evaluation, we conduct in-depth analysis and experiments on Transformer models with various depths and attention mechanisms to improve the tail label performance. We show that a shallow Transformer model with word-label attentions can effectively leverage word-level features and outperforms previous Transformers on tails labels.

1 Introduction

Extreme multi-label text classification (XMTC) is a task for tagging each document with the most relevant subset of labels from an extremely large label set, in which the number of labels can be from a few thousands to more than a million. It has a wide range of potential applications, such as tagging keywords for advertising, recommendation system, or product category classification.

In the enormous label space, the skewed distribution is the main challenge because the label frequency follows the Zipf's Law. As a result, a majority of low-frequency labels (named as tail labels) cover only a small subset of training instances, while a minority of high-frequency labels (named as head labels) cover a large subset of training instances. In the Wiki-31K dataset shown in Figure 1, only 1% of labels has more than 100 training instances, but they cover more than 40% of all the training instances. On the other hand, 85% of labels only cover less than 30% of training instances. Other benchmark datasets show similar distribution (Appendix A).

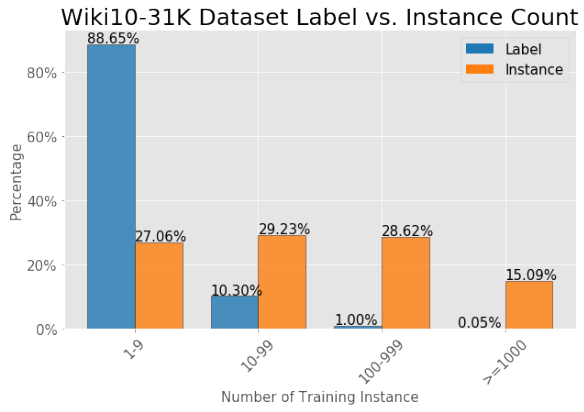

Figure 1: The percentage of label vs. training instance in Wiki10-31K dataset. 1% of head labels cover more than 40% of training instances, while 88% of tail labels cover less than 30% of training instances.

It is worth pointing out that the performance of current SOTA models has been evaluated using the micro-averaging precision scores as the dominating metrics (Liu et al., 2017; You et al., 2018; Ye et al., 2020; Chang et al., 2020; Jiang et al., 2021). Those metrics assign an equal weight to the score of each instance and hence are dominated by the head labels that appear more frequently. In other words, if a model performs well on a few head labels, it will be scored highly in micro-based metrics but that cannot be counted as the evidence for the model to perform well on tail labels.

In order to measure the true success of existing methods on massive tail labels in XMTC, a re-examination based on macro-based metrics is necessary, where the performance score on each label is given equal weight in calculating the average.

This paper presents such a re-examination. Specifically, by comparing the relative improvement of SOTA neural models with respect to the binary Support Vector Machines (SVM) in *binned macro-averaging F1 scores*, where each bin is a group of labels with similar frequencies, our experiments reveal new insights into the methods.

In our evaluation, we found that the deep pre-trained Transformer-based XMTC models (Ye et al., 2020; Jiang et al., 2021; Chang et al., 2020) perform worse than the simple SVM baseline on the tail labels, while a shallow word-label attention-based RNN model (You et al., 2018) can outperform the SVM baseline on two of the benchmark datasets. We speculate that the deep-layered architectures in those Transformer models may not be optimal for modeling low-frequency word-level features which are informative for tail label prediction. We found that a simple shallow Transformer with label-word attention that can better leverage word embedding with a more desirable inductive bias towards the low-frequency cases. Our experimental results demonstrate the effectiveness of the shallow Transformer model in tail label prediction.

2 Background

We will introduce the notation and formulation in Section 2.1, the SOTA methods in Section 2.2 and the commonly used evaluation metrics in Section 2.3.

2.1 Task Introduction

We denote a training dataset with N instances and L labels as $\mathcal{D} = \{(\mathbf{x}_{i=1}^{N}, \mathbf{y}_{i=1}^{N})\}$ such that $\mathbf{x}_i$ is the input text and $\mathbf{y}_i \in \{0,1\}^L$ is the ground truth label list.

The goal of the XMTC task is to learn a scoring function $s_l = f(\mathbf{x}, l) \in \mathbb{R}$ which maps an input $\mathbf{x}$ and a candidate label l to a score s_l, such that for a relevant label, $s_l = 1$, otherwise $s_l = 0$.

The simplest approach to optimize $f(\mathbf{x}, l)$ is the one-vs-all classifiers, which trains an independent binary classifier for each label:

$$\min_f \quad \frac{1}{NL} \sum_{i=1}^{N} \sum_{l=1}^{L} \mathcal{L}(y_{il}, f(\mathbf{x}_i, l))$$

where $\mathcal{L}(\cdot,\cdot)$ is a point-wise loss function, such as a hinge loss for Linear SVM or a logistic regression as the final layer of a neural network:

$$\begin{aligned}\mathcal{L}_{\text{hinge}} &= \max(0, 1 - \dot{y}_{il} f(\mathbf{x}_i, l)) \\ \mathcal{L}_{\text{logistic}} &= \log(1 + exp(-\dot{y}_{il} f(\mathbf{x}_i, l)))\end{aligned}$$

where $\dot{y}_{il} = 2y_{il} - 1 \in \{-1, 1\}$.

In the evaluation, since multi-labels can be relevant for an input document $\mathbf{x}$, a typical way is to measure the quality of a top-k ranked list of the predicted label, denoted as $\mathbf{p}^k$:

$$\mathbf{p}^k = \text{Top-k}([f(\mathbf{x}, 1), \ldots, f(\mathbf{x}, L)])$$

2.2 Methods under Investigation

Based on the design of the classification scoring function f, we discuss three types of methods: the *linear SVM*, the *document-vector representation-based method* and the *word-label attention-based method*. Specifically, the SVM is a non-neural baseline and the other two methods are used the SOTA deep learning models.

2.2.1 Linear SVM Model

The one-vs-all linear SVM model is a simple realization of $f(\mathbf{x}, l)$:

$$f(\mathbf{x}, l; \mathbf{w}) = \mathbf{w}_l^T \phi_{\text{tf-idf}}(\mathbf{x})$$

where $\phi_{\text{tf-idf}}(\mathbf{x})$ converts each text input into d dimensional tf-idf feature, and $\mathbf{W} = [\mathbf{w}_{l=1}^{L}] \in \mathbb{R}^{L \times d}$ is the parameter for linear classifier. The tf-idf feature uses the human defined heuristics to reflect the distributed importance of each word in the training corpus.

Although more complicated kernels such as the RBF kernel are introduced in the SVM model for the classification tasks, (Chang and Lin, 2011) shows that a Linear SVM achieves the same performance with a RBF kernel SVM when the feature space is large (refer to Table 1 for the number of features in the benchmark datasets). Therefore, we use the Linear SVM to lower computational cost without affecting the performance.

2.2.2 Document-vector Representation

The document-vector representation-based methods (*doc-vec*) use neural networks as $\phi(\mathbf{x})$ to replace the human defined $\phi_{\text{tf-idf}}(\mathbf{x})$, shown in Fig 2 left. Specifically, Liu et al. (2017) are the first to apply CNN model (Kim, 2014) to encode the input text $\mathbf{x}$ into a single bottleneck vector as the document-level representation in the XMTC task. Recently, the deep Transformer-based (Vaswani

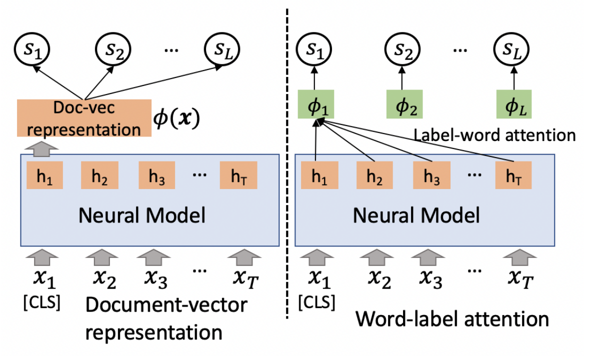
s_1
s_2
s_L
Doc-vec representation
$\phi(x)$
h_1 h_2 h_3 ... h_T
Neural Model
x_1 x_2 x_3 ... x_T
[CLS]
Document-vector representation
ϕ_1 ϕ_2 ϕ_L
Label-word attention
Word-label attention

Figure 2: Document-vector representation (doc-vec) vs. word-label attention (word-label). The doc-vec method encodes the input text into a fixed vector representation to calculate score s_i, while the word-label attention extracts local word features to form label specific document embeddings ϕ_l.

et al., 2017) models quickly exceed the performance of CNN and achieve the state-of-the-art. We choose the deep Transformer-based models and write score function as follows:

$$f(\mathbf{x}, l) = \mathbf{w}_l^T \phi_{\text{transformer}}(\mathbf{x}) \qquad (1)$$

where $\phi_{\text{transformer}}(\mathbf{x})$ is the deep transformer feature extractor such as BERT (Devlin et al., 2018) or XLNet (Yang et al., 2019). Again, $\mathbf{w}$ represents a binary classifier applied on top of the document vector to compute the score for each label, which is optimized by the logistic regression objective $\mathcal{L}_{\text{logistic}}$.

The most representative deep Transformer-based models tailored for the XMTC task are X-transformer (Chang et al., 2020), APLC-XLNet (Ye et al., 2020) and LightXML (Jiang et al., 2021), which are discussed in Appendix C.

2.2.3 Word-label Attention

The word-label attention-based method (*word-label*) extracts local word features to form label specific embeddings, which avoids using a fixed doc-vec representation, as shown in Fig 2 right.

Specifically, the word-label attention allows each label to interact with each word by the attention mechanism. Let T denote the length of input $\mathbf{x}$, $\mathbf{h}_{i=1}^T \in \mathbb{R}^d$ denote the contextualized word embedding and $\mathbf{w}_{j=1}^L \in \mathbb{R}^d$ denote the label embedding. The label-word attention is calculated by

$$\alpha_{ij} = \frac{e^{\mathbf{h}_i \mathbf{w}_j}}{\sum_{i=1}^T e^{\mathbf{h}_t \mathbf{w}_j}} \qquad (2)$$

The attention score is then used to calculate the label specific feature vector

$$\phi_{\text{word-label}}(\mathbf{x}, l) = \sum_{i=1}^T \alpha_{ij} \mathbf{h}_i \qquad (3)$$

Finally, the score function is calculated by:

$$f(\mathbf{x}, l) = \text{MLP}(\phi_{\text{word-label}}(\mathbf{x}, l)) \qquad (4)$$

where $\text{MLP} : \mathbb{R}^d \rightarrow \mathbb{R}$ is the multi-layer perceptron function that summarizes the label specific feature into a real-valued score. Although in this method there is no explicit classifier, the parameter $\mathbf{w}_l$ serves a similar role of the classifier as in the previous methods.

This method can be related to a word-level retrieval, where a label measures the word importance via the interaction term $\mathbf{h}_i \mathbf{w}_j$, and then the word-level information is combined by the a weighted sum $\sum_{i=1}^T \alpha_{ij} \mathbf{h}_i$.

Although this method was originally proposed in AttentionXML (You et al., 2018), we rename it to the word-label attention-based method as a counterpart to the document vector-based method to signify the distinction between the two types of methods. Also, the contextualized word embedding $\mathbf{h}$ is obtained from a one layer shallow LSTM in the original paper, but in our paper, we will also generalize this method to the shallow or deep Transformer-based models.

2.3 Evaluation Metrics

In XMTC, there are multiple relevant labels for each document, so a typical way for evaluation is to measure the quality of the label ranked list produced by the model.

The micro-based metrics for a ranked list are the most commonly used in previous works, where the performance score is averaged over all the test instances. Since the test instances are weighted equally, the instance level metrics are usually dominated by the head labels which have more training instances. These metrics can be formulated as:

$$\text{Micro@k} = \frac{1}{N} \sum_{i=1}^N \text{Metric}(\mathbf{p}_i^k, \mathbf{y}_i) \qquad (5)$$

where N is the number of test instances and $\mathbf{p}_i^k, \mathbf{y}_i$ are the predicted top k ranked list and the ground truth labels respectively. Metric is a function to score the quality of the ranked list such as P@k (Precision at k) (Liu et al., 2017; You et al., 2018;

Dataset	N_{train}	N_{test}	F	$\lvert\bar{L_d}\rvert$	$\lvert L\rvert$	$\lvert L_{b1}\rvert$	$\lvert L_{b2}\rvert$	$\lvert L_3\rvert$	$\lvert L_{b4}\rvert$	$\lvert\bar{W}\rvert$
EURLex-4K	15,539	3,809	186,104	5.30	3,956	2,413	1,205	182	1	1,248
Wiki10-31K	14,146	6,616	101,938	18.64	30,938	26,545	3,084	300	16	2,484
AmazonCat-13K	1,186,239	306,782	203,882	5.04	13,330	3,936	5,813	2,862	719	246

Table 1: Data statistics. N_{train} and N_{test} denote the number of training and testing instances respectively. F is the tf-idf feature size. $|\bar{L_d}|$ is the average number of labels per document. $|L|$ is the number of labels. L_{bk} refers to the number of labels in bin k, in which the labels in bin $[1, 2, 3, 4]$ have $[1 \sim 9, 10 \sim 99, 100 \sim 999, 1000 \sim]$ instances respectively. This bin partition is used in our binned macro-based F1. $|\bar{W}|$ refers to the average word number per training document.

Prabhu et al., 2018; Khandagale et al., 2019; Ye et al., 2020; Chang et al., 2020; Jiang et al., 2021), R@k (Recall at k), N@k (Normalized Discounted Cumulative Gain at k) (Prabhu and Varma, 2014) , or PSP@k (Propensity-scored Performance at k) (Jain et al., 2016), which are discussed in Section B.1.

In this paper, we report the micro F1@k (refer to Sec 3 for details) as the evaluation metric, which is calculated by:

$$\text{F1@k} = 2\frac{P@k \cdot R@k}{P@k + R@k} \tag{6}$$

3 Issues in Existing Evaluation

Recently, various deep learning models are proposed to solve the data sparse issue in the skewed distribution in the XMTC task. However, we found that due to the issues of the evaluation metrics, the significance on the improvement is not evident, especially on the tail labels. We list the 3 main issues of the evaluation in the previous work as follows:

Missing Macro-averaging Metrics The current evaluation only focuses on the micro-averaging metrics such as P@5, which only measure how well a model performs on the instance level, but not on the label level. Instead, the macro-averaging metrics measure the model performance on the label level, but they are ignored in recent works.

Missing Label Recall Evaluation The widely used precision metric only focuses on the accuracy of prediction, but not on how many labels are predicted. Especially in tail label evaluation, measuring the precision of a model which hardly predicts any tail label is not meaningful. Actually, the recall metric is important to measure how likely the tail labels are predicted. We propose to use F1 as our metric since it can balance the precision and recall.

Missing Simple SVM Baseline The recent work only has in depth comparison with other deep learning models, but the simple one-vs-all linear SVM is missing. We acknowledge that several previous works include tree-based models in which part of the modules are built upon SVM (Prabhu et al., 2018; Khandagale et al., 2019), but they don't directly compare with the simple one-vs-all SVM because they may assume it as just a weaker baseline. In our paper, we conducted in-depth comparisons with other deep learning models to show that one-vs-all SVM is not without advantages.

4 Re-evaluation of SOTA Models

4.1 Datasets

In this paper, we use three benchmark datasets: EURLex-4K (Loza Mencía and Fürnkranz, 2008), Wiki10-31K (Zubiaga, 2012) and AmazonCat-13K (McAuley and Leskovec, 2013). The statistics of the datasets are shown in Table 1. For EURLex-4K and Wiki10-31K, the bin with $1 \sim 9$ training instances covers 63.48% and 88.65% of the labels respectively. The AmazonCat-13K dataset contains more instances, where the bin with $1 \sim 9$ instances covers 30% of the labels and the first two bins together with $1 \sim 99$ instances cover more than 70% of labels. If the model cannot perform well on those bins, it means that the predictions of most labels are inaccurate. The data statistics shed light on the importance of tail label prediction in this task.

The details of the experiment settings, descriptions of SOTA models and the training hyperparameters are discussed in Appendix C.

4.2 Macro-averaging Metrics

For label level evaluation, we proposed to apply the *macro-averaging F1* metric (refer to Sec B.2 for more details), where the performance score on each label is given an equal weight when calculating the average. Moreover, the macro-averaging F1 metric can be applied to groups of labels according to their training frequency to clearly reflect the performance on head labels and tail labels. We call this metric the *binned macro-averaging F1* metric. Let L_b be a group of labels, then the macro-based F1 on L_b is calculated by:

$$\text{Macro@k} = \frac{1}{|L_b|} \sum_{l \in L_b} \text{F1}_l^k \tag{7}$$

where L_b is the label set for bin b and F1_l^k is the F1 metric for label l evaluated on top k ranked list $\mathbf{p}^k$.

Specifically, the F1@k for label l over predicted ranked list $\mathbf{p}^k$ is calculated according to the confusion matrix at Table 2:

$$\text{P}_l^k = \frac{\text{TP}_l^k}{\text{TP}_l^k + \text{FP}_l^k}, \quad \text{R}_l^k = \frac{\text{TP}_l^k}{\text{TP}_l^k + \text{FN}_l^k}$$

$$\text{F1}_l^k = 2\frac{\text{P}_l^k \cdot \text{R}_l^k}{\text{P}_l^k + \text{R}_l^k}$$

where $\text{P}_l^k, \text{R}_l^k, \text{F1}_l^k$ are the precision, recall, F1 at k for label l. As a designed choice, we set $\text{F1}_l^k = 0$ if a label l is never predicted.

	l in $\mathbf{y}$	l not in $\mathbf{y}$
l in $\mathbf{p}^k$	True Positive(TP_l^k)	False Positive(FP_l^k)
l not in $\mathbf{p}^k$	False Negative(FN_l^k)	True Negative(TN_l^k)

Table 2: Confusion Matrix for label l and top k ranked list $\mathbf{p}^k$.

4.3 Performance in Micro-averaging Metrics

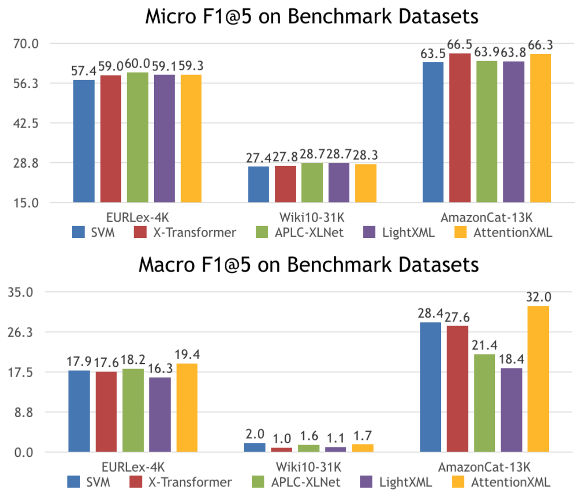

Figure 3: The result of micro and macro-averaging F1@5 metrics. The two metrics gives different conclusion on the best performing model.

In Figure 3, we report the micro-averaging F1@5 for the SOTA models on the benchmark datasets. More evaluation results on micro P@k, micro F1@k and macro F1@k (k=1, 3, 5) are reported in Table 3 in Appendix. The other micro-averaging metrics such as N@k and PSP@k are reported and discussed in Appendix C.

In the micro-based evaluation, all the SOTA neural models outperform the SVM baseline on the 3 benchmark datasets. We found that the deep Transformer model achieves the best performs across the datasets. Specifically, APLC-XLNet is the best on the EUR-Lex and Wiki10-31K datasets, and the X-Transformer is the best on the AmazonCat-13K dataset.

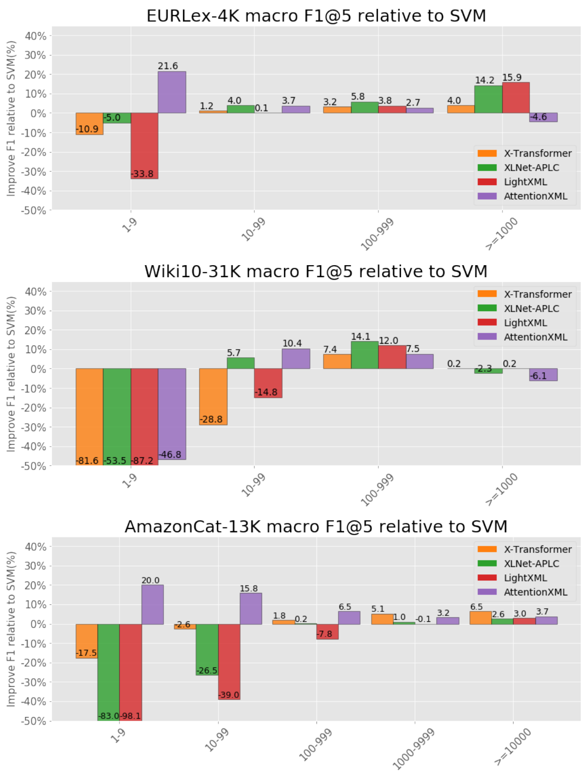

Figure 4: The relative improvement of SOTA deep learning models over one-vs-all SVM baseline on the binned macro-averaging F1@5 metrics. The labels are partitioned to bins whose labels have $[1 \sim 9, 10 \sim 99, 100 \sim 999, 1000 \sim 999, \ldots]$ instances respectively. The corresponding label number of each bin can be found in Table 1.

4.4 Performance in Macro-averaging Metrics

At the bottom of Figure 3, we report the macro-averaging F1@5 for the same models under investigation. The results of the macro-averaging metrics give different conclusions from the micro-averaging evaluation. Specifically, the SVM baselines performs the best in the Wiki10-31K, where the label space is both large and skewed. It also achieves competitive results on the other two datasets. Especially on the AmazonCat-13K dataset, the SVM beats all the deep Transformer-based models which perform better on the micro-averaging metric.

We also observe that the AttentionXML model is the best on both the EUR-Lex and the AmazonCat-13K datasets, though it is not the best in any datasets when evaluated with the micro-averaging metric.

4.5 Binned Macro-averaging F1 Metric

In order to evaluate the performance of the tail labels, we show the binned macro-averaging F1 in Figure 4. We report the improvement of the macro-averaging F1 metric on each bin relative to the SVM baseline. We have two main observations: 1) all the deep Transformer-based models underperform the SVM baseline on the tail labels. 2) The AttentionXML model tends to outperform the SVM baseline on the tail labels.

Deep Transformers vs. SVM On all the dataset, the deep Transformer-based model inevitably underperform the SVM baseline on the tail bin with $1 \sim 9$ training instances. Note that although the tail bin contains relatively few instances, it actually covers most of the label set. For instance, the tail bin of EUR-Lex and Wiki10-31K covers 63% and 89% of the labels respectively. On the other hand, the deep Transformer-based model tends to have an advantage over the SVM baseline when the number of training instance is greater than 100. This shows that the source of improvement of the SOTA deep Transformer model comes from the improvement on the few head labels which covers a large number of instances.

AttentionXML vs. SVM For EUR-Lex and AmazonCat-13K datasets, the AttentionXML outperforms the SVM baseline on the tail bin where the labels have $1 \sim 9$ training instances, while all the other deep learning models underperform the SVM baseline. The AttentionXML model also shows an advantage over the SVM baseline on the head labels, but it tends to underperform the deep neural models on the head bins. This shows that the AttentionXML with the word-label attention can be beneficial for tail label prediction, but potentially at the price of sacrificing the head label prediction.

4.6 Micro vs. Macro-averaging Metric

Compared with the micro-averaging metric in evaluation, the macro-averaging metric focuses on the label level prediction. We find that by shifting our attention from the quality of instance-level prediction to the label-level prediction especially on the tail labels, the conclusion on the best performing model is reversed. Our conclusions are two folds: 1) the SVM baseline outperforms the deep Transformer models on the tail label prediction. 2) the AttentionXML, which uses shallow one layer LSTM with word-label attention, achieves better tail label performance on two datasets.

Explanation We speculate the reason accountable for the our observation lies in the nature of the model architecture . The SOTA transformer models use document-vector representation-based method, gradually encodes global information into a contextualized embedding ([CLS] token) via self-attention. With abundant training instances, the powerful Transformer feature extractors can learn abstract semantic representations of documents, and thus enhancing the performance on head labels. However, without sufficient training instances, the deep Transformers overfit on the tail labels during training, failing to learn informative representations generalized on testing data.

Different from the representation-based methods, the AttentionXML uses the word-label attention-based method, which avoids encoding all the textual information into a single vector. Rather, the shallow RNN keeps the local word level information and the word-label attention allows the classifier to better aggregate them for different label predictions. For tail-label prediction, as there are not enough training instances to provide high-level meaningful semantic information for neural models, it is critical to rely on the word level feature for better prediction.

5 Improving Tail Label Prediction

From the evaluations in the previous sections, we observe that the deep Transformer-based models achieves the best on the micro-based metrics, while the AttentionXML model achieves the best results on the tail label prediction on the two benchmark datasets. Our hypothesis is that the deep Transformer models are powerful at exploiting the global semantic meaning because they are initialized with the general language knowledge from large text

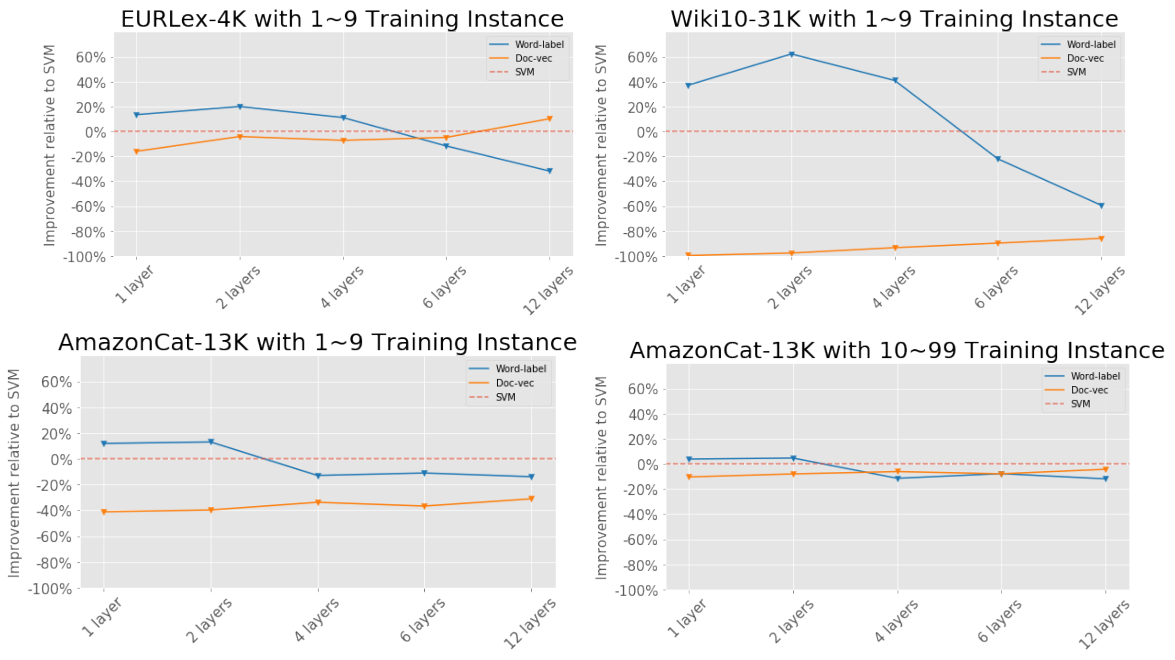

Figure 5: Comparison of the word-label attention-based method and the document vector-based method with different layers of BERT model on the benchmark datasets. The relative improvement to SVM on macro-averaging F1 is plotted.

corpus via unsupervised pretraining. On the other hand, the word-label attention method in the AttentionXML exploits the less abstract semantics such as the word embedding features that may benefit tail label prediction when training instances are scarce. A natural question is that whether we can combine the two to use the word-label attention to leverage the word knowledge in the pretrained model, and thus achieving consistent improvement on tail label prediction over all the benchmark datasets.

To this end, we propose to apply the word-label attention on the BERT model to integrate the word knowledge into the word-label matching process. We show that this method can achieve consistent improvement on tail label predictions.

5.1 Transformers with Word-label Attention

In the word-label attention, each label attends to the contextualized word embedding to select the most relevant information from each word, which more efficiently utilizes the local word features. Specifically, the contextualized word embeddings **h** in Eq.(2) are the outputs from a neural model and each label integrates the most relevant **h** to form the feature embedding in Eq.(3). The final score function is calculated by Eq.(4).

To obtain the contextualized word embedding, the AttentionXML uses a 1 layer bi-directional RNN model. In a pretrained Transformer model, there are 12 layers for BERT-Base and 24 layers for BERT-Large, and different layers may encode different level syntactic or semantic features (Jawahar et al., 2019). Therefore, we explore the effect of using different layers of Transformer models with either word-label attention or typical the document-vector representation-based methods in the next section.

5.2 Experiments with Shallow and Deep BERT Models

In this section, we show the performance of tail label prediction by controlling the number of layers for BERT by $[1, 2, 4, 6, 12]$ for both the word-label attention method (*word-label*) and the document-vector representation-based method (*doc-vec*). For "k layer(s) BERT", we simply use the first k layer(s) of BERT, discarding the remaining top layers. The models are trained on the full training data, and the relative improvement over the SVM baseline is reported on the tail labels ($[1 \sim 9]$ for EURLex-4K and Wiki10-31K, $[1 \sim 9]$, $[10 \sim 99]$ for AmazonCat-13K). The results of model performance are shown in Figure 5 and more evaluations are shown in Appendix 5.

In the results, we have two key conclusions 1) the word-label methods perform the better with shallower BERT models and the doc-vec methods perform better with deeper BERT models, and 2) the shallow BERT model with 1 or 2 layers show consistently improvement over the SVM baseline on tail label predictions.

The results also align with our hypothesis. For deeper Transformer models, the input tokens interact with each others via self-attention, gradually forming more complex and abstract semantic representations from layer to layer. Hence, the representations of the top layer encode the global document semantic. On the other hand, the shallower Transformer models reflect more of the token level meaning with only a little context information. The stronger word-level feature can help the word-label attention better leverage the local information for enhanced tail label prediction.

Our findings raise an interesting point that most of the pretrained models use the [CLS] embedding as a default and the local feature of Transformer models in bottom layers are often ignored, but they can be potentially beneficial for different tasks.

In addition to the improvement in tail label prediction, the shallower Transformer model has fewer number of parameters, which makes the training and inference much faster.

5.3 Comparison with SOTA Baselines

We compare our shallow word-label BERT models (1 and 2 layers) with the other SOTA neural models. We report the relative improvement of macro-averaging F1 metric relative to the SVM baseline shown in Figure 6. Specifically, we focus on the performance of tail bins (bin with $1 \sim 9$ instances for EURLex-4K and Wiki10-31K, and bins with $1 \sim 9, 10 \sim 99$ instances for AmazonCat-13K), where the deep Transformer models underperform the SVM baseline.

Our word-label BERT models outperform the SOTA deep Transformer models consitently. Compared to AttentionXML, our proposed method has better performance on Wiki10-31K dataset and the performance is on par on EUR-Lex dataset. The results confirm that the word-label attention on a shallow Transformer can improve the tail-label performance.

Similar to the AttentionXML, our word-label BERT has relatively weaker performance on the head bins with training instance greater than 100 on the Wiki10-31K and AmazonCat-13K datasets. This is also reflected in the micro-averaging metrics in Table 3 in Appendix D. This is probably because the two datasets have much skewer label distribution where in Wiki10-31K, 1.05% head labels covers 33.71% of training instances and in AmazonCat-13K, 0.68% head labels covers 51.86% of training instances, as shown in Fig 7. This demonstrates that the success on micro-based metrics is not equivalent to the success on tail label predictions.

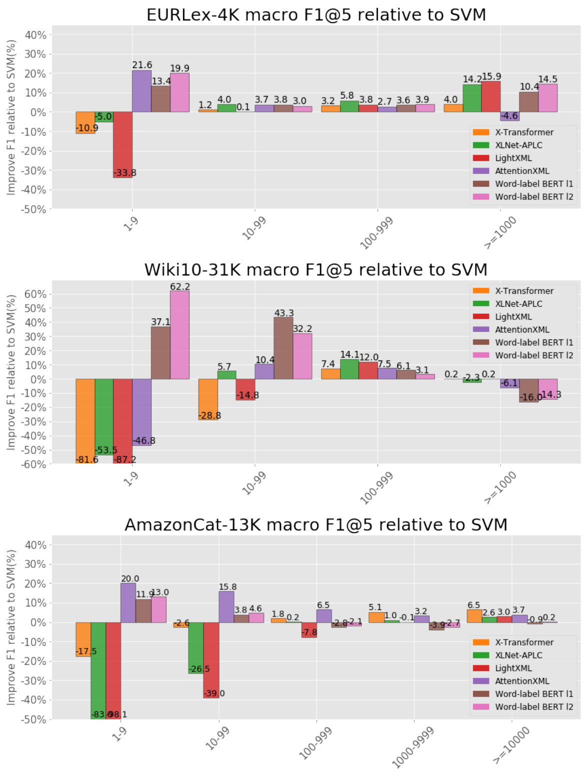

Figure 6: A comparison of our shallow word-label BERT with 1 and 2 layers vs. other baseline models on the relative improvement to SVM. Our models show consistent improvement on the tail bins.

6 Conclusion

In this paper, we show that the widely used micro-averaging evaluation metrics on XMTC are not sufficient to reflect the true performance of the tail label prediction. By a re-evaluation with macro-averaging metrics, we draw different conclusions on the best model for tail label prediction, and find that the word-label attention-based methods are potentially more suitable. Finally, we propose to combine word-label attention with shallow pre-trained BERT to obtain consistent improvement on label prediction.

References

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.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

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*.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy.

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*.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*.

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. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2021. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*.

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.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. 2020. Equalization loss for long-tailed object recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11662–11671.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

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*.

Yuzhe Yang and Zhi Xu. 2020. Rethinking the value of labels for improving class-imbalanced learning. In *Advances in Neural Information Processing Systems*.

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*.

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

A Data Distribution

In Figure 7, we show the percentage of label vs. training instance in each bin in the benchmark datasets: EURLex-4K, Wiki10-31K and AmazonCat-13K dataset. In all the datasets, a small percentage of head labels cover most of the of training instances, while a large percentage of tail labels only cover a few training instances.

B Evaluation Metrics

We include more discussion and reference for the micro-averaging metrics and our proposed macro-averaging binned F1 metric.

B.1 Micro-averaging Metrics

In XMTC, there are multiple true labels for each instance, so it is important to present a ranked list of predicted labels for evaluation. The micro-averaging metrics calculate a score for each of the ranked list and then take an average over all the test instances.

$$\text{Micro@k} = \frac{1}{N}\sum_{i=1}^{N}\text{Metric}(\mathbf{p}_i^k, \mathbf{y}_i) \tag{8}$$

where N is the number of test instances and $\mathbf{p}_i^k, \mathbf{y}_i$ are the predicted top k ranked list and ground truth labels list. Metric is a function to score the quality of the ranked list based on the ground truth labels list. In the following, we omit the index of instance i for clarity.

In most of the previous work, the Metric function is chosen to be the micro-averaging Precision at k (P@k) (Liu et al., 2017; You et al., 2018; Ye et al., 2020; Chang et al., 2020; Jiang et al., 2021). Specifically, this metric evaluates the quality of the top k of the prediction ranked list for a given test instance:

$$P@k = \frac{1}{k}\sum_{l=1}^{k}\mathbb{1}_{\mathbf{y}}(\mathbf{p}_l) \tag{9}$$

where $\mathbf{p}_l$ is the l-th label in the predicted ranked list $\mathbf{p}$ and $\mathbb{1}_{\mathbf{y}}$ is the indicator function.

Other choices of the Metric function include N@k (normalized Discounted Cumulative Gain at k) or PSP@K (Propensity-scored Performance at

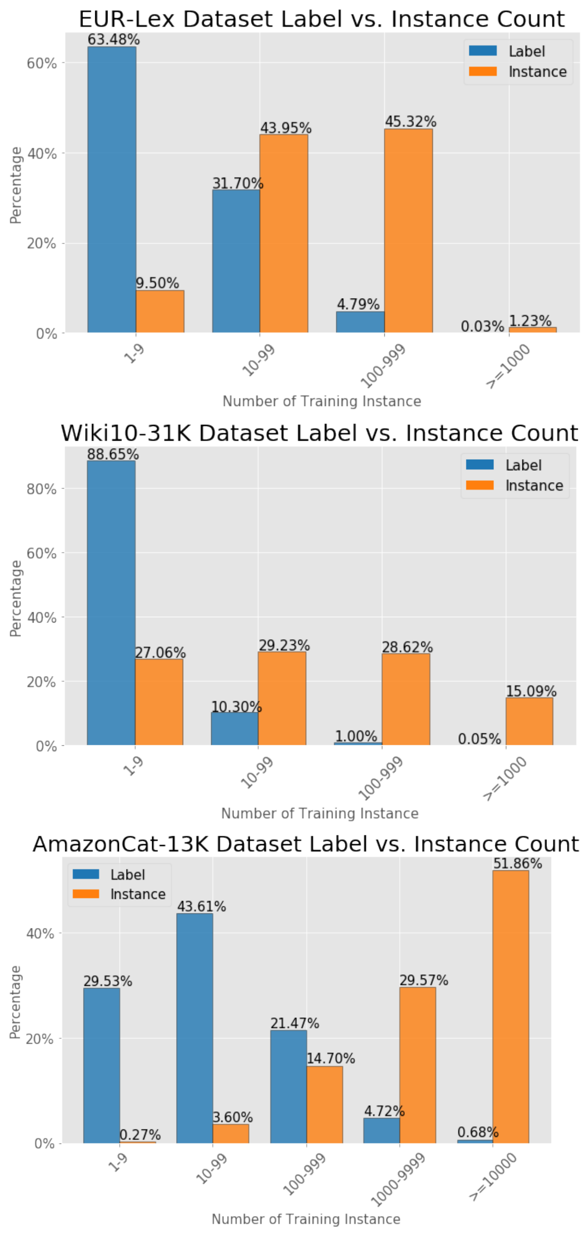

Figure 7: The percentage of label vs. training instance in EURLex-4K, Wiki10-31K and AmazonCat-13K dataset. A small percentage of head labels cover most of the of training instances, while a large percentage of tail labels only cover a few training instances.

k) (Jain et al., 2016). The N@k is defined as:

$$DCG@k = \sum_{l=1}^{k} \frac{\mathbb{1}_{\mathbf{y}}(\mathbf{p}_l)}{\log(l+1)}$$

$$iDCG@k = \sum_{l=1}^{\min(k,\|\mathbf{y}\|)} \frac{1}{\log(l+1)}$$

$$N@k = \frac{DCG@k}{iDCG@k}$$

Although N@k and P@k are calculated differently, You et al. shows that they are the same metrics when measuring the quality of ranked lists.

The PSP @ k is defined as:

$$PSP@k = \frac{1}{k} \sum_{l=1}^{k} \frac{\mathbb{1}_{\mathbf{y}}(\mathbf{p}_l)}{\mathrm{prop}(\mathbf{p}_l)}$$

where $\mathrm{prop}(\mathbf{p}_l)$ is the propensity score (Jain et al., 2016) of label $\mathbf{p}_l$, which gives higher weight for tail labels. Although PSP@k were used as evaluation metric for the long tail problem, it is still not informative enough to compare the performance of head labels vs. tail labels due to following: 1) It is still a micro-averaging metric that can be influenced by label with more instances. 2) Since it summarizes the performance into one number, we are not sure if the gain comes from the improvement of head labels or tail labels. Actually, an improvement in either type of labels will give an increment in the final score.

B.2 Macro-averaging Metrics

As we discussed above, the micro-averaging metrics are not informative enough to measure the performance of different types of labels, because they assign equal weights to each instance when taking the averaging. As a remedy, a label level evaluation should assign equal weights to each label when taking the average. Therefore, macro-averaging metrics should be applied for evaluating the label level performance, especially for the tail labels.

In computer vision, such metrics have been used in object recognition in manually sampled long-tail datasets. (Yang and Xu, 2020; Wang et al., 2021) evaluates the accuracy on each label and (Kang et al., 2020; Menon et al., 2021) linearly partition the labels into groups. However, the label size of datasets in object recognition is relatively small compared to the XMTC datasets. To create groups for large label size, we split the labels in the scale of exponential of 10 as a design choice, e.g. $\{1 \sim 9, 10 \sim 99, 100 \sim 999, \dots\}$.

We used the F1 metric to balance the precision and recall. As observe in (Tan et al., 2020), the tail labels tend to be under predicted due to the more frequent negative gradient penalty. As a result, tail labels tend to be predicted less often than the head labels, and thus the recall is low. If a model hardly predicts any tail labels, it should receive a low score for the corresponding bin. Therefore, we apply the F1 metric to balance the number of predictions (recall) and the accuracy of the predictions (prediction).

C Experiment Settings

C.1 Re-evaluation Experiment Settings

We choose the non-neural SVM as our baseline and investigate 4 SOTA deep learning models for XMTC: AttentionXML (You et al., 2018), X-Transformer (Chang et al., 2020), APLC (Ye et al., 2020) and LightXML (Jiang et al., 2021).

In the original papers, the experiments are conducted in different settings, e.g. (Chang et al., 2020; Jiang et al., 2021) reports the ensemble of multiple models and different works have their own data processing method. For fair comparison, we run our experiments with single model with the following training and testing data. We obtain the datasets from the Extreme classification Repository[1]. However, the repository only contains the stemmed version of EURLex-4K, which hurts the performance of pretrained Transformer models whose tokenizers are applied to unstem natural text. Therefore, we obtain the unstemmed version of the EURLex-4K from the APLC-XLNet github[2].

C.2 Descriptions of SOTA models

SVM The one-vs-all SVM (Cortes and Vapnik, 1995) is a simple baseline for XMTC. The features are tf-idf word importance features.

AttentionXML The AttentionXML model uses the word-label attention mechanism on top of a bi-directional LSTM to create label specific document embedding. The word-label attention allows each label to select the most relevant information among the words in the input document. The extracted features are passed to a MLP to obtain the relevance score of the labels.

[1]`http://manikvarma.org/downloads/XC/XMLRepository.html`

[2]`https://github.com/huiyegit/APLC_XLNet.git`

X-Transformer The X-Transformer (Chang et al., 2020) was proposed to tame large Transformer models to the XMTC task. Specifically, (Chang et al., 2020) designs a two stage training procedure including a cluster-level classification and a within cluster ranking. The cluster-level classification is implement with large Transformer models such as BERT-large (Devlin et al., 2018), Roberta-large (Liu et al., 2019) and XLNet-large (Yang et al., 2019). The ranker is implemented with one-vs-all SVM with both tf-idf and Transformer features as input.

APLC Instead of a two stage training, the APLC model (Ye et al., 2020) achieves an end-to-end training by using the XLNet-base model (12 layers transformer) with decreasing sizes of document embedding to avoid the scalability issue. The decreasing size is achieved by a pooler function on top of the XLNet special [CLS] token, and the scale of decrements is determined by the frequency for training instance of each label.

LightXML (Jiang et al., 2021) learns the document vector to as feature for both the cluster-level classifier and the within cluster label ranking. The document vector is generated by concatenating multiple Transformers cls token embedding to collect more semantic information.

C.3 Training Settings and Hyperparameters

For the AttentionXML model, we use the same hyperparameter as in their code [3]. For the X-Transformer mode, the reuse the released pretrained model [4] for evaluation.

For the end-to-end Transformer-based models including APLC-XLNet and LightXML, we train the model using the same framework as APLC, including discriminative fine-tuning and slanted triangular learning rates (Ye et al., 2020). The discriminative fine-tuning decouples the learning rate of model into 3 parts: The base Transformer module, the pooler module and the linear classifier module, such that the Transformer module receives smallest learning rate and the linear module receives largest learning rate. The slanted triangular learning rates allows the model to warm up with a slowly increasing learning rate first, and then decrease the learning rate to let the model converge stably.

[3]https://github.com/yourh/AttentionXML

[4]https://github.com/OctoberChang/X-Transformer

For our experiment on shallow vs. deep Transformers, we follow the same setting as the APLC-XLnet and the LightXML. Specifically, we experiment with $[1, 2, 4, 6, 12]$ layers; the learning rate for the Transformer, the pooler and the linear classifier are $5e-5, 1e-4, 2e-3$ for EURLex-4K and AmazonCat-13K and $1e-5, 1e-4, 1e-3$ for Wiki10-31K; the batch sizes are $12, 12, 48$ for EURLex-4k, Wiki10-31K and AmazonCat-13K respectively. For the doc-vec experiment, we use the BERT-Base model which gives the best performance. For the word-label BERT experiment, we found that BERT-Large gives better performance for shallow models with layer $1, 2, 4$ and the results are similar for layer $6, 12$. Therefore, we report the results of the BERT-Large model for the word-label BERT experiments.

D More Experimental Results

In Table 3, we report the micro P@k, micro F1@k and the macro F1@k (k=$1, 3, 5$) for the SOTA models on the benchmark datasets. We observe that the micro P@5 and the F1@5 shows similar conclusions on which models perform the best, but the macro F1@5 gives different conclusions, as discussed in the Sec. 4.4.

In Table 4, we report the N@k and PSP@k (k=$1, 3, 5$) for the SOTA models on the benchmark datasets. For N@k, it shows the same conclusions with P@k on which model performs the best (they are mathematically equivalent). Although the PSP metric were used to measure the performance of tail label prediction, we found that it gives higher score for the some of deep Transformer-base models than the SVM baseline, e.g. X-Transformer, XLNet-APLC on EURLex-4K, XLNet-APLC on Wiki10-31K and X-Transformer on AmazonCat-13K. However, those models actually underperform the SVM baseline on tail labels in our binned macro-averaging F1 metric designed for tail label evaluation. This shows that the PSP metric still scarifies from the micro-averaging when applied to evaluate the tail label performance.

The complete results of the shallow vs. deep Transformer model with document-vector representation-based model and the word-label attention-based model are shown in Table 5. For each dataset, we show performance of model with layer $[1, 2, 4, 6, 12]$. The general observation is that for deeper layers, the doc-vec BERT tends to perform better, while for shallow layers, the word-label BERT tends to perform better. The reason is that the word-label attention mechanism exploits the local word feature from shallow Transformer, but the doc-vec method relies on the global information in the contextualized embedding learned from deeper models.

Although the shallow word-label BERT and the deep doc-vec BERT tend to perform similarly on the micro-averaging metrics, only the shallow word-label BERT outperform the SVM baseline on the tail labels (from our binned macro F1 evaluation). This shows that the widely used micro-averaging metrics are not informative to measure the performance of tail label prediction.

EURLex-4K									
Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
SVM	83.44	70.62	59.08	26.50	51.06	57.37	5.64	13.28	17.89
X-Transformer	85.46	72.87	60.79	27.14	52.69	59.03	5.12	12.86	17.56
XLNet-APLC	86.83	74.34	61.94	27.43	53.55	59.96	6.66	14.51	18.19
LightXML	86.98	73.38	61.07	27.48	52.86	59.12	5.68	12.85	16.31
AttentionXML	85.12	72.80	61.01	27.03	52.64	59.25	7.69	15.80	19.41
Word-label BERT L1	86.59	73.21	60.90	27.36	52.74	58.96	7.47	15.58	19.03
Word-label BERT L2	86.06	72.97	60.84	27.19	52.56	58.90	7.45	15.70	19.28
Wiki10-31K									
Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
SVM	84.61	74.64	65.89	8.45	20.33	27.42	0.18	0.97	1.99
X-Transformer	87.12	76.51	66.69	8.70	20.84	27.76	0.14	0.51	1.02
XLNet-APLC	88.59	78.30	68.87	8.85	21.33	28.67	0.31	0.93	1.59
LightXML	88.59	78.51	68.84	8.85	21.39	28.65	0.25	0.69	1.10
AttentionXML	86.46	77.22	67.98	8.63	21.03	28.30	0.39	1.05	1.67
Word-label BERT L1	84.18	73.67	64.37	8.41	20.07	26.80	0.86	1.95	2.77
Word-label BERT L2	82.46	72.07	63.61	8.24	19.64	26.48	0.90	2.03	2.84
AmazonCat-13K									
Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
SVM	93.20	78.89	64.14	30.54	58.42	63.49	3.86	16.29	28.36
X-Transformer	95.75	82.46	67.22	31.38	61.06	66.54	1.90	12.50	27.55
XLNet-APLC	94.56	79.78	64.59	30.99	59.07	63.93	3.95	13.32	21.38
LightXML	94.61	79.83	64.45	31.00	59.11	63.79	1.60	8.38	18.36
AttentionXML	95.53	82.03	67.00	31.31	60.74	66.31	7.12	21.94	31.98
Word-label BERT L1	93.00	77.85	62.94	30.48	57.65	62.30	8.78	21.74	29.06
Word-label BERT L2	93.32	78.29	63.33	30.58	57.97	62.68	8.51	21.91	29.30

Table 3: SVM and SOTA Deep Transformer Models evaluated on benchmark datasets: EURLex-4K, Wiki10-31K and AmazonCat-13K. The metrics are micro P@k, micro F1@k and macro F1@k for k=$1, 3, 5$. We include our Word-label BERT with layer $1, 2$ for comparison.

EURLex-4K						
Methods	N@1	N@3	N@5	PSP@1	PSP@3	PSP@5
SVM	83.44	73.58	65.70	38.76	46.71	51.17
X-Transformer	85.46	75.84	67.62	37.85	47.05	51.81
XLNet-APLC	86.83	77.29	68.79	42.21	49.83	52.88
LightXML	86.98	76.53	68.05	40.54	47.56	50.50
AttentionXML	85.12	75.74	67.71	44.20	50.85	53.87
Wiki10-31K						
Methods	N@1	N@3	N@5	PSP@1	PSP@3	PSP@5
SVM	84.61	76.99	70.34	11.89	14.23	15.96
X-Transformer	87.12	79.00	71.56	12.52	13.62	14.63
XLNet-APLC	88.59	80.72	73.58	14.43	15.38	16.47
LightXML	88.59	80.87	73.58	14.09	14.87	15.52
AttentionXML	86.46	79.41	72.47	14.49	15.65	16.54
AmazonCat-13K						
Methods	N@1	N@3	N@5	PSP@1	PSP@3	PSP@5
SVM	93.20	82.87	76.47	51.26	64.69	72.34
X-Transformer	95.75	86.26	79.80	51.42	66.14	75.57
XLNet-APLC	94.56	83.89	77.29	52.55	65.11	71.36
LightXML	94.61	83.95	77.21	50.70	63.14	70.13
AttentionXML	95.53	85.87	79.54	54.94	69.48	76.45

Table 4: SVM and SOTA Deep Transformer Models evaluated on benchmark datasets: EUR-Lex (4K), Wiki10-31K and AmazonCat-13K. The metrics are N@k and PSP@k for k=1, 3, 5.

EUR-Lex

Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
Doc-vec BERT L1	83.43	69.26	57.08	26.36	49.89	55.26	5.74	12.79	16.19
Doc-vec BERT L2	84.91	70.45	58.30	26.83	50.75	56.44	6.20	13.81	17.35
Doc-vec BERT L4	84.99	71.21	58.76	26.85	51.29	56.88	5.99	13.43	17.13
Doc-vec BERT L6	84.72	71.66	59.12	26.77	51.62	57.23	6.20	13.95	17.37
Doc-vec BERT L12	85.64	73.57	61.20	27.06	53.00	59.25	6.32	14.90	18.85
Word-label BERT L1	86.59	73.21	60.90	27.36	52.74	58.96	7.47	15.58	19.03
Word-label BERT L2	86.06	72.97	60.84	27.19	52.56	58.90	7.45	15.70	19.28
Word-label BERT L4	85.54	72.82	60.34	27.03	52.45	58.42	7.51	15.81	18.96
Word-label BERT L6	82.15	68.67	56.82	25.96	49.46	55.01	6.52	13.70	16.60
Word-label BERT L12	81.65	69.19	57.47	25.80	49.84	55.63	4.59	11.35	15.22

Wiki10-31K

Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
Doc-vec BERT L1	84.48	70.82	59.91	8.44	19.30	24.94	0.05	0.20	0.36
Doc-vec BERT L2	85.66	72.89	61.95	8.56	19.86	25.79	0.08	0.31	0.54
Doc-vec BERT L4	87.02	75.07	64.68	8.69	20.46	26.93	0.11	0.42	0.78
Doc-vec BERT L6	87.13	75.46	65.42	8.70	20.56	27.24	0.13	0.47	0.87
Doc-vec BERT L12	88.42	77.04	66.88	8.83	20.99	27.84	0.17	0.59	1.07
Word-label BERT L1	84.18	73.67	64.37	8.41	20.07	26.80	0.86	1.95	2.77
Word-label BERT L2	82.46	72.07	63.61	8.24	19.64	26.48	0.90	2.03	2.84
Word-label BERT L4	83.41	73.86	65.27	8.33	20.13	27.17	0.85	1.89	2.67
Word-label BERT L6	83.02	71.80	62.03	8.29	19.56	25.83	0.50	1.27	1.90
Word-label BERT L12	84.89	74.26	64.00	8.48	20.24	26.65	0.29	0.83	1.37

AmazonCat-13K

Methods		Micro			Micro			Macro	
	P@1	P@3	P@5	F1@1	F1@3	F1@5	F1@1	F1@3	F1@5
Doc-vec BERT L1	91.85	76.18	61.32	30.10	56.41	60.69	6.48	17.38	24.40
Doc-vec BERT L2	93.22	77.83	62.80	30.55	57.63	62.16	4.64	16.29	25.11
Doc-vec BERT L4	93.86	78.75	63.64	30.76	58.31	62.99	4.31	16.01	25.82
Doc-vec BERT L6	94.01	78.97	63.83	30.81	58.47	63.18	3.96	15.33	25.46
Doc-vec BERT L12	94.49	79.72	64.59	30.97	59.03	63.93	3.94	15.69	26.38
Word-label BERT L1	93.00	77.85	62.94	30.48	57.65	62.30	8.78	21.74	29.06
Word-label BERT L2	93.32	78.29	63.33	30.58	57.97	62.68	8.51	21.91	29.30
Word-label BERT L4	93.55	77.61	62.34	30.66	57.47	61.70	2.54	15.15	25.43
Word-label BERT L6	93.44	77.70	62.59	30.62	57.53	61.95	4.81	17.11	26.19
Word-label BERT L12	94.26	78.87	63.60	30.89	58.40	62.95	1.80	12.95	25.47

Table 5: The performance of various layers configuration and attention mechanism for BERT model evaluated on micro P@5, micro F1@5 and macro F1@5.