Exploiting Local and Global Features in Transformer-based Extreme Multi-label Text Classification

Anonymous ACL submission

Abstract

Pre-trained Transformer models have been successfully applied to the extreme multi-label text classification (XMTC) task, which aims to tag each document with the relevant labels from a very large output space. When applying Transformer models to text classification, a typical usage is to adopt the special CLS token embedding as the document feature. Intuitively, the CLS embedding is a summarization of a Transformer layer to reflect the global semantic of a document. While this may be sufficient for smaller scale classification tasks, we find that the global feature itself is not sufficient to reflect fine-grained semantics in a document under extreme classification. As a remedy, we propose to leverage all the token embeddings in a Transformer layer to represent the local semantics. Our approach combines both the local and global features produced by a Transformer model to represent different granularity of document semantics, which outperforms the state-of-the-art methods on benchmark datasets.

1 Introduction

Extreme multi-label text classification (XMTC) is the task of tagging each document with relevant labels where the target space may contain up to a million categories. A central problem in XMTC is to extract the semantic information of a document to support the prediction over multiple relevant labels at various granularity levels. As an example, the Amazon product "Falling in Love Is Wonderful" is a collection of Broadway love duets. The category "music" reflects a high-level summarization of the product content, while "vocalist" and "soundtracks" are finer-grained categories which are associated with the keywords "singer" and "recording" in the text description. A successful XMTC model should capture the general concept of a document and, at the same time, identify the meaningful keywords in order to make accurate predictions for more distinguished labels.

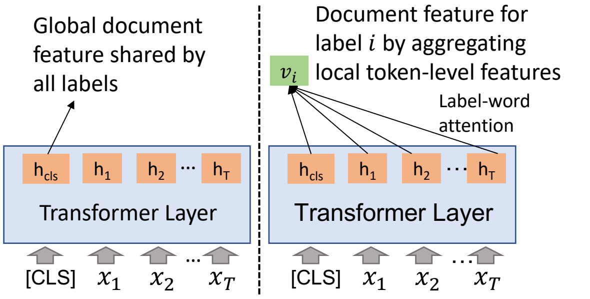

Global document feature shared by all labels
Document feature for label i by aggregating local token-level features
v_i
Label-word attention
h_{cls}
h_1
h_2
···
h_T
Transformer Layer
[CLS]
x_1
x_2
x_T

Figure 1: Document representation from global feature v.s. local features from a Transformer layer.

When applying Transformer-based models for XMTC, previous works use the embeddings of [CLS] token at the last (Chang et al., 2020; Ye et al., 2020) or last few layers (Jiang et al., 2021) as the document representation. Since the [CLS] embeddings at each Transformer layer summarizes the content of the input document into a single vector, we call them the *global features* in our paper. However, only using global features is too coarse to reflect the semantics of such a large label space. As shown in figure 1, the embeddings of all word tokens, which we call the *local features*, can directly participate in label prediction. Specifically, we apply the label-word attention that lets each label attentively select the keywords in the document text, which emphasizes the semantic matching between each label and document words.

In this paper, we propose an integrating framework that combines both global and local features in pre-trained Transformer models, namely GloCalXML. Our contributions are : 1) we reveal the effectiveness of token-level features in Transformer models for XMTC, 2) we conduct extensive experiments to investigate the optimal choice of global and local features to be combined, and 3) we dig the reason of the performance gain to the level of contextualization of Transformer layers (in section 4 for more details).

2 Proposed Method

2.1 Preliminaries

Let $\boldsymbol{x} = \{x_1, x_2, \ldots, x_T\}$ be the input document with length T, and the set of associated ground truth labels is $\boldsymbol{y} \in \mathbb{R}^C$ with $y_c \in \{0, 1\}$, where C is the label size. A classifier calculates a probability p_c of the label being relevant. The binary cross entropy (BCE) loss between $\boldsymbol{p} = \{p_1, p_2, \ldots, p_C\}$ and $\boldsymbol{y}$ is calculated as:

$$\mathcal{L}_{\text{BCE}}(\boldsymbol{p}, \boldsymbol{y}) = -\frac{1}{L}\sum_{l \in L}\Big[y_c \log p_c + (1 - y_c)\log(1 - p_c)\Big] .$$

The document x is usually prepended with a special [CLS] token as the input to Transformer. For a Transformer model with L layers, the hidden representations from the n-th layer is denoted as:

$$\phi^{(n)}_{\text{transformer}}(\boldsymbol{x}) = \{\boldsymbol{h}^{(n)}_{cls}, \boldsymbol{h}^{(n)}_1, \boldsymbol{h}^{(n)}_2, ..., \boldsymbol{h}^{(n)}_T\} . \tag{1}$$

We will introduce our classification system with global and local features respectively.

2.2 Classification with Global Features

Global features are summarized in the [CLS] embedding. Since the [CLS] embedding from the last layer L contains the richest information after multiple layers of self-attention, we use $\boldsymbol{h}^{(L)}_{cls}$ (or optionally passed to a linear pooler) as the global feature. The probability of predicting a label c is calculated by:

$$p_c^{\text{global}} = \sigma(\langle \boldsymbol{h}^{(L)}_{cls}, \boldsymbol{e}_c^{\text{global}} \rangle) , \tag{2}$$

where $\boldsymbol{e}_c^{\text{global}}$ is the label embedding for global features, σ is the sigmoid activation and $\langle \cdot, \cdot \rangle$ is the dot product.

2.3 Classification with Local Features

The local features denote all the token embeddings at a certain layer of Transformer, which preserve the diverse and fine-grained token level information. As the first layer token embeddings from a Transformer model mostly pertain to the token surface-level meaning (while being contextualized), they are used as the local features.

Similar to the label-word attention (You et al., 2018), each label attentively select the key tokens from the document. The labels are treated as queries to retrieve the salient tokens in the documents ($\boldsymbol{h}^{(1)}_{cls}$ is written as $\boldsymbol{h}^{(1)}_0$):

$$\psi_{\text{K}}(\phi^{(1)}_{\text{transformer}}(\boldsymbol{x})) = \{\boldsymbol{w}^k_1, \boldsymbol{w}^k_2, \ldots, \boldsymbol{w}^k_T\} , \tag{3}$$

$$\alpha_{ij} = \frac{\exp(\langle \boldsymbol{w}^k_i, \boldsymbol{e}^{local}_j \rangle / \tau)}{\sum_{t=0}^{T} \exp(\langle \boldsymbol{w}^k_t, \boldsymbol{e}^{local}_j \rangle / \tau)} , \tag{4}$$

where ψ_K is a linear function, $\boldsymbol{e}^{local}_j$ is the label embedding for local features and τ is the temperature. τ controls the smoothness of the attention distribution over the words. With a smaller $\tau < 1$, the attention is peaked on the most salient key tokens.

The retrieved key tokens are aggregated according to the relevance score α_{ij}:

$$\psi_{\text{V}}(\phi^{(1)}_{\text{transformer}}(\boldsymbol{x})) = \{\boldsymbol{w}^v_1, \boldsymbol{w}^v_2, \ldots, \boldsymbol{w}^v_T\} , \tag{5}$$

$$\boldsymbol{v}_j = \sum_{i=0}^{T} \alpha_{ij} \boldsymbol{w}^v_i , \quad p_j^{\text{local}} = \sigma(\phi_{MLP}(\boldsymbol{v}_j)) , \tag{6}$$

where ψ_V is a linear function and $\phi_{MLP}(\boldsymbol{v}_j)$ is a multi-layer perceptron that maps $\boldsymbol{v}_j$ into a real-valued score.

Equation 2 and 4 highlight the difference between using the global and local features: for the global features, relevance scores are computed between the label embeddings and the document embedding; for the local features, relevance scores are directly computed between the label embeddings and the token embeddings inside a document.

2.4 Inference

Our framework integrates the classification with local and global features, and the final prediction for a given input text and label l is:

$$p_c^{\text{final}} = \frac{1}{2}(p_c^{\text{local}} + p_c^{\text{global}}) . \tag{7}$$

2.5 Training Objective

The training objective optimizes a summation of global and local losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}}(\boldsymbol{p}^{\text{global}}, \boldsymbol{y}) + \mathcal{L}_{\text{BCE}}(\boldsymbol{p}^{\text{local}}, \boldsymbol{y}) . \tag{8}$$

Optimizing the $\mathcal{L}_{\text{BCE}}$ losses for the global and local classifiers encourages each of them to focus on its own specialities. As the backbone of Transformer is shared, the number of parameters is not increased, but the features induced by Transformer (both latent token-level embeddings and [CLS] embedding) are used more efficiently.

Dataset	N_{train}	N_{test}	$\bar{C}_d$	C
EURLex-4K	15,539	3,809	5.30	3,956
Wiki10-31K	14,146	6,616	18.64	30,938
AmazonCat-13K	1,186,239	306,782	5.04	13,330
Wiki-500K	1,779,881	769,421	4.75	501,070

Table 1: Corpus Statistics: N_{train} and N_{test} are the number of training and testing instances respectively; $\bar{C}_d$ is the average number of labels per document, and C is the number of unique labels. An unstemmed version of EURLex-4K is obtained from the APLC-XLNet github and the rest are from the Extreme classification Repository. Details are included in appendix section B.

	EURLex-4K			Wiki10-31K			AmazonCat-13K			Wiki-500K		
Methods	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5	P@1	P@3	P@5
DisMEC	83.21	70.39	58.73	84.13	74.72	65.94	93.81	79.08	64.06	70.21	50.57	39.68
PfastreXML	73.14	60.16	50.54	83.57	68.61	59.10	91.75	77.97	63.68	56.25	37.32	28.16
eXtremeText	79.17	66.80	56.09	83.66	73.28	64.51	92.50	78.12	63.51	65.17	46.32	36.15
Parabel	82.12	68.91	57.89	84.19	72.46	63.37	93.02	79.14	64.51	68.70	49.57	38.64
Bonsai	82.30	69.55	58.35	84.52	73.76	64.69	92.98	79.13	64.46	69.26	46.72	36.46
XML-CNN	75.32	60.14	49.21	81.41	66.23	56.11	93.26	77.06	61.40	59.85	39.28	29.81
AttentionXML	87.12	73.99	61.92	87.47	78.48	69.37	95.92	82.41	67.31	76.95	58.42	46.14
X-Transformer	87.22	75.12	62.90	88.51	78.71	69.62	96.70	83.85	68.58	77.28	57.47	45.31
APLC-XLNet	87.72	74.56	62.28	89.44	78.93	69.73	94.56	79.82	64.60	72.83	50.50	38.55
LightXML	87.63	75.89	63.36	89.45	78.96	69.85	96.77	**84.02**	68.70	77.78	58.85	45.57
XR-Transformer	88.41	75.97	63.18	88.69	80.17	70.91	**96.79**	83.66	68.04	79.40	59.02	46.25
GlocalXML	**90.32**	**78.90**	**66.20**	**90.11**	**80.95**	**71.97**	96.60	83.97	**68.78**	**80.52**	**61.30**	**47.72**

Table 2: The evaluation of representative classification systems with P@k metric. The bold phase and underscore highlight the best and second best model performance.

3 Experiments

3.1 Datasets

We conduct our experiments on 4 benchmark datasets: EURLex-4K, Wiki10-31K and AmazonCat-13K, and Wiki-500K. The statistics of the datasets are shown in Table 1.

3.2 Evaluation Metrics and Settings

We evaluate our models with the widely used precision at k (P@k) metric:

$$P@k = \frac{1}{k} \sum_{i \in \text{top-k}(\hat{\mathbf{y}})} y_i \tag{9}$$

where $\text{top-k}(\hat{\mathbf{y}})$ are the top k predicted labels.

Following the experimental settings in previous XMTC evaluation (Jiang et al., 2021; Chang et al., 2020; Zhang et al., 2021), we report the ensemble score of three Transformer-base models (except for Wiki-500K): BERT (Devlin et al., 2018), Roberta (Liu et al., 2019) and XLNet (Yang et al., 2019). For Wiki-500K, we leverage the label clustering algorithm in LightXML for scalability and similarly report the ensemble of three different label clusters with Roberta. The implementation details are included in appendix B.

3.3 Baselines

We compare our model with the statistical and neural baselines. The statistical models include one-vs-all DisMEC (Babbar and Schölkopf, 2017), PfastreXML (Jain et al., 2016); tree-based Parabel (Prabhu et al., 2018), eXtremeText (Wydmuch et al., 2018). The deep learning approaches include XML-CNN (Liu et al., 2017), AttentionXML (You et al., 2018); SOTA pre-trained Transformer models X-Transformer (Chang et al., 2020), APLC-XLNet (Ye et al., 2020) and LightXML (Jiang et al., 2021), and XR-Transformer (Zhang et al., 2021).

3.4 Main Result

The performance of model evaluated with the P@k metric is reported in table 2. Our model is compared against the representative statistical and neural models, with the best performance highlighted in bold phase and the second best in underline.

The most competitive baselines are the pre-trained transformer-based models. Our model outperforms those SOTA models on EURLex-4K and Wiki10-31K, Wiki-500K measured with P@1, P@3 and P@5, and achieves competitive performance on the AmazonCat-13K dataset. We attribute the performance gains to the additional utilization of local features in Transformers. Labels with more specific categorization may directly benefit from attending to token embeddings in the Transformer, when the [CLS] embedding fails to capture the fine-grained details. Although we don't observe significant improvement in AmazonCat-13K dataset, our model still performs the best or second best compared with the SOTA models.

4 Analysis on Global & Local Features

We further analyze the effectiveness of global and local features. Experiments are performed on EURLex-4K (more in appendix C) with single

Roberta model of sequence length 256. The global feature uses the [CLS] embedding at the final layer, while the local features are the token embeddings from layers 0-12. The layer 0 corresponds to the original token embedding before being passed into any Transformer layer.

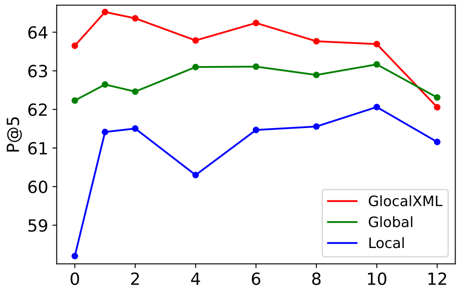


Figure 2: (EURLex-4k) Analysis on the effectiveness of the global feature ([CLS] embedding at the final layer) and local feature from different layers. GlocalXML is our proposed method. The horizontal axis is the local layer number, the vertical axis is the P@5 performance. Layer 0 corresponds to the original token embeddings.

Q1: How effective are local features alone?

Figure 2 compares the effectiveness of classifier with global and local features. The classifier with global feature (layer 12) has a slight variation due to the joint optimization of local and global features. The classifiers with local features from any layers inevitably underperform that with the global feature, with layer 0 performing significantly worse since its word embeddings are not contextualized.

Q2: Can local features complement global feature with additional information?

First of all, combining the global and local features can boost performance in most cases. Surprisingly, the combination of global feature with local feature at layer 0 achieves competitive results, even if the local feature at layer 0 performs the worst by itself. On the other hand, even local features from higher layers tend to perform better by themselves, we observe that combining global feature with a higher layer of local feature, especially at layer 12, is less effective. The performance of GlocalXML is peaked when the local feature is at layer 1, providing an empirical evidence of our design choice. We attribute the gains to the complementary information in local and global feature.

Q3: Underlying reasons why lower layer local features are better?

To explain this situation, we hypothesize: even if the token embeddings at a higher layer preserves the token meaning [1], it becomes more contextualized after multiple layers of self-attention. Consequently, when a label queries from more contextualized word embeddings, it is harder to pick up the salient keywords information. To verify the hypothesis, in figure 3, we study the correlation between the relative improvement (red curve) of GlocalXML over the global classifier and the Jensen–Shannon divergence (green curve) of the predicted label distributions by the local and global classifier. There are two observations from the experiment: 1) the predicted label distributions by local and global classifiers become more similar when a higher Transformer layer of word embeddings is used, and 2) a higher distribution similarity is correlated with a lower improvement of GlocalXML over the global classifier. This indicates that using more contextualized word embeddings for the local classifier leads to homogeneous features as using the global feature, which undermines the complementary of the two.

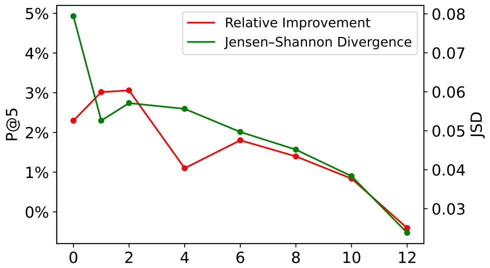


Figure 3: (EURLex-4K) the relative improvement of GlocalXML over the global feature, and the JSD between predicted label distributions with global and local feature. The global feature uses [CLS] at final layer and the local features come from different layers.

5 Conclusion

In this paper, we propose GlocalXML , a classification system integrating both the global and local features from the pre-trained Transformers. The global classifier uses [CLS] embedding as the summarization of document, and the local classifier uses the label-word attention to directly select salient part of texts for classification. Our model combines the two to capture different granularity of document semantics, which achieves superior or comparable performances over SOTA methods on the benchmark datasets.

[1] After all, the MLM loss is optimized to predict the word identity at the final layer of pre-trained Transformer.

References

Rohit Babbar and Bernhard Schölkopf. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 721–729.

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*.

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*, pages 935–944.

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*.

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*.

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*, pages 993–1002.

Marek Wydmuch, Kalina Jasinska, Mikhail Kuznetsov, Róbert Busa-Fekete, and Krzysztof Dembczynski. 2018. A no-regret generalization of hierarchical softmax to extreme multi-label classification. *Advances in neural information processing systems*, 31.

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*.

Jiong Zhang, Wei-Cheng Chang, Hsiang-Fu Yu, and Inderjit Dhillon. 2021. Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 34:7267–7280.

A Limitations

For large datasets like Wiki-500K in the extreme XMTC repo, we only experiment with simple fixed-size clusters. The recent state-of-the-art XR-Transformer leverages different granularity of tree structures, which could be adapted to our model and we leave that to the future work. Another limitation is that this work doesn't leverage the label descriptions, which could be helpful when combined with the local classifier to better retrieve keywords from the documents.

B Implementation Details

B.1 Datasets

We conduct our experiments on 4 benchmark datasets: EURLex-4K, Wiki10-31K and AmazonCat-13K, and Wiki-500K. The statistics of the datasets are shown in Table 1. An unstemmed version of EURLex-4K is obtained from the APLC-XLNet github[2] and the rest are from the Extreme classification Repository[3].

B.2 Hyperparameters

Following APLC-XLNet (Ye et al., 2020), we use different learning rate for the Transformer backbone, the pooler (optional) and the classifier, which are $1e-5, 1e-4, 1e-3$ for the Wiki10-31K and $5e-5, 2e-4, 2e-3$ for the other two datasets. For the classifier with local features, we use learning rates of $2e-4, 2e-3$ for the attention module and MLP respectively. The sequence length is set to 512 for BERT and Roberta on EURLex-4K and

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

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

Wiki10-31K, and 256 for AmazonCat-13K and the XLNet model. The fp16 training is enable to improve training efficiency.

For the large scale dataset Wiki-500K, we adopt the cluster-based algorithm from LightXML for scalability. Following their evaluation criteria, we report the ensemble of 3 different clusters using the Roberta-base model. We use a learning rate of $5e-5$ for the Roberta backbone, $1e-3$ for the classifier and the attention module. We use 256 as sequence length.

C More Results

C.1 Single Model Performance

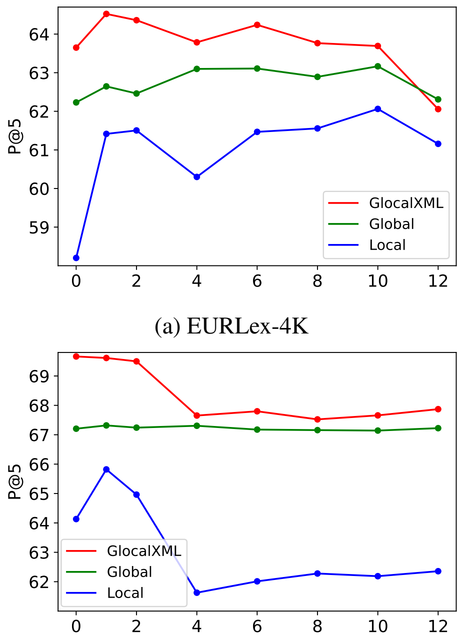
The performance of a single Roberta model (with single cluster) is reported in table 3. The classifier with local feature inevitably underperforms that with the global feature, probably because the local classifier only shares a shallow Transformer backbone which is less expressive. Despite that, when the local and global features are combined, GLOCALXML achieves the best performance, which could come from the complementary effect of local and global feature (analysed below).

Dataset		Global	Local	GLOCALXML
EURLex-4K	P@1	87.27	85.98	88.93
	P@3	75.09	73.36	76.90
	P@5	62.97	60.76	64.22
Wiki10-31K	P@1	87.62	85.31	89.60
	P@3	77.00	75.49	80.17
	P@5	68.25	66.92	70.99
AmazonCat-13K	P@1	96.27	95.08	96.27
	P@3	83.25	81.39	83.40
	P@5	68.09	66.26	68.25
Wiki-500K	P@1	76.48	73.82	78.91
	P@3	57.17	53.52	59.71
	P@5	44.05	41.81	45.57

Table 3: Ablation test results for GLOCALXML with a single model initialized with Roberta. The performance for the GLOCALXML, local and glocal classifiers is reported separately.

C.2 GlocalXML with Different Layers

In figure 4, we report more results on the combination of global and local features from different layers. We observe that the performance of classification with the global feature is relatively stable which outperforms that with the local features. Our GLOCALXML model with a combination of the two features achieves the best performance. For the Wiki10-31K dataset, GLOCALXML achieves better performance when the features comes from layers < 3, even with the original token embedding at layer 0. The reason is that since this dataset has a large label space and each document has an average of 19 labels, it is more difficult for the [CLS] embedding to summarize the text with distinctive word-level features peculiar to the labels. Therefore, the local classifier which allows labels to directly query for the keywords in the document could pick up the missing information. Combining the two leads to better results.



(a) EURLex-4K

(b) Wiki10-31K

Figure 4: Ablation test on the effectiveness of combining the global feature ([CLS] embedding at the final layer) with different layers of local feature. The horizontal axis is the local layer number, the vertical axis is the P@5 performance. Layer 0 corresponds to the original token embeddings.

C.3 Optimization for Local Features

We run the experiments on EURLex-4k and Wiki10-31k datasets using Roberta model for additional experiments by adding a stop gradient on the first layer of the transformer feature. This separates the training trajectory for local and global feature optimization.

The result shown in table 4 is slightly worse than GlocalXML in the paper, but the variance is very small. Detaching the Transformer feature for local attention module makes the optimization of local model slightly worse. For global model, the performance on Wiki10-31K is a little bit better but EURLex-4K is a little bit worse. Again, the variance is not significant.

Dataset		Global	Local	GLOCALXML
EURLex-4K	P@1	87.12	85.96	88.51
	P@3	75.04	72.68	76.73
	P@5	62.94	59.56	64.27
Wiki10-31K	P@1	88.72	85.34	89.41
	P@3	78.37	74.74	79.79
	P@5	68.89	65.56	70.69

Table 4: Analysis for Roberta model when a stop gradient is added to the first layer.