HAXMLNet: Hierarchical Attention Network for Extreme Multi-Label Text Classification

Ronghui You  
School of Computer Science  
Fudan University  
rhyou18@fudan.edu.cn

Zihan Zhang  
School of Computer Science  
Fudan University  
17210240027@fudan.edu.cn

Suyang Dai  
School of Computer Science  
Fudan University  
16210240005@fudan.edu.cn

Shanfeng Zhu  
School of Computer Science  
Fudan University  
zhusf@fudan.edu.cn

Abstract

Extreme multi-label text classification (XMTC) addresses the problem of tagging each text with the most relevant labels from an extreme-scale label set. Traditional methods use bag-of-words (BOW) representations without context information as their features. The state-of-the-art deep learning-based method, AttentionXML, which uses a recurrent neural network (RNN) and the multi-label attention, can hardly deal with extreme-scale (hundreds of thousands labels) problem. To address this, we propose our HAXMLNet, which uses an efficient and effective hierarchical structure with the multi-label attention. Experimental results show that HAXMLNet reaches a competitive performance with other state-of-the-art methods.

1 Introduction

Extreme multi-label text classification (XMTC) addresses the problem of tagging each document with the most relevant labels from an extreme-scale label set. For capturing context information, long-term dependency and the most relevant part for each label, the state-of-the-art deep learning-based method, AttentionXML [7] is proposed. AttentionXML used a recurrent neural network (RNN) with a multi-label attention. For given context inputs $h_i \in \mathbb{R}^n$, multi-label attention gets output as follows:

$$\alpha_{ij} = \frac{e^{h_i \cdot w_j}}{\sum_{t=1}^T e^{h_t \cdot w_j}},$$

$$m_j = \sum_{i=1}^T \alpha_{ij} h_i,$$

(1)

where $\alpha_{ij}$ is the normalized coefficient of $h_i$ and $w_j \in \mathbb{R}^n$ is the so-called attention parameters. However, AttentionXML cannot handle extreme-scale datasets.

In this paper, we proposed HAXMLNet, keeping label-wise attention and using a probabilistic label tree (PLT) [2] for solving extreme-scale datasets. Experimental results show that HAXMLNet reaches a competitive performance on two extreme-scale datasets with other state-of-the-art methods.

Preprint. Work in progress.
2 Method: HAXMLNet

Directly using AttentionXML[7] can hardly deal with solve extreme-scale multi-label classification (with hundreds of thousands labels) due to its model scale (can’t put in GPU memory) and computational complexity (slow training and prediction). For solving classification with extreme-scale labels, we proposed an efficient and effective method, HAXMLNet, which also uses label-wise attention with a PLT to reduce the model scale and computational complexity during training and prediction as follows:

1. We partitioned all labels into $g$ groups. The difference of size among each group is less than one. Each label $j$ belongs to only one group $G(j)$ as its father node, which is called group label for label $j$. We add a root node as father of all group label and construct a PLT with height of 3 (including root node).

2. For each training sample, we generate its group labels, which are all group labels of its raw labels. Using group labels as target labels, it’s still a multi-label classification problem, but notice that the size of group labels are $|L_g|$ times smaller than the original raw labels. We train an AttentionXML called HAXMLNet-G with these group labels.

3. We use all labels in groups of each training sample as its candidate labels. Obviously, candidate labels of each sample includes its all raw (positive) labels and some negative labels. We train another AttentionXML named HAXMLNet-L by using these candidate labels only. During training process, we calculate attention and loss scores only with these candidate labels. The loss function of HAXMLNet-L is given as follows:

$$J(\theta) = -\frac{1}{N|C(i)|} \sum_{i=1}^{N} \sum_{j \in C(i)} y_{ij} \log(y_{ij}) + (1 - y_{ij}) \log(1 - \hat{y}_{ij}),$$  \hspace{1cm} (2)

where $C(i)$ is the candidate labels of $i$th sample, $y_{ij}$ and $\hat{y}_{ij}$ are the ground truth and predicted probability, respectively. Note that $\hat{y}_{ij}$ is a conditional probability because we used a PLT and we already know that $G(j)$ is a truth group label of sample $i$. The expectation number of candidate labels for each sample is the average number of group labels per sample (no more than the average number of raw labels) timing the group size, which is much smaller than the number of all labels. We also use a number $c=1000$, if candidate number of this sample is larger than $c$, we only keep $c$ candidate labels randomly.

4. During prediction, for a given sample, the predicted score $\hat{y}_j$ for $j$th label based on probability chain rule is as follows:

$$\hat{y}_j = \hat{y}_{G(j)}^{\text{group}} \times \hat{y}_j^{\text{label}}$$  \hspace{1cm} (3)

where $\hat{y}_{G(j)}^{\text{group}}$ is the predicted score for the group $G(j)$ by AttentionXML-G, and $\hat{y}_j^{\text{label}}$ is the predicted score for the $j$th label by AttentionXML-L. For the efficiency of our model, we only predicted the scores for labels in top $k=10$ groups.

5. If complexity of AttentionXML-G is still too large, we can use the above algorithm on this multi-label classification problem recursively.

Here we train two AttentionXML models (HAXMLNet-G and HAXMLNet-L), but notice that the complexity of both HAXMLNet-G and HAXMLNet-L are much smaller than AttentionXML against all raw labels directly. **If the scale of HAXMLNet-G is still too large, we use this algorithm recursively because training HAXMLNet-G is also a multi-label classification task.**

Inspired by Parabel [5], we use a similar top-down hierarchical clustering to constructing a PLT rather than random construction. This clustering approach divided a label clustering into two clusters with a balanced k-means and repeated recursively until the size of all clusters smaller than a given leaf size. Different from Parabel, we only keep the leaves of this tree as its clustering result. Now we can solve extreme-scale datasets by label-wise attention efficiently.
Table 1: Datasets we used in our experiments.

| Dataset    | N_{train} | N_{test} | D | L | \bar{L} | L | W_{train} | W_{test} |
|------------|-----------|----------|---|---|---------|---|-----------|----------|
| Amazon-670K| 490,449   | 153,025  | 670,091 | 5.45 | 3.99    | 247.33 | 241.22    |
| Wiki-500K  | 1,779,881 | 76,942   | 501,008 | 4.75 | 16.86   | 808.66 | 808.56    |

N_{train}: #training instances, N_{test}: #test instances, D: #features, L: #labels, \bar{L}: average #labels per instance, \bar{L}: the average #instances per label, W_{train}: the average #words per training instance and W_{test}: the average #words per test instance. The partition of training and test is from the data source.

3 Experiments

3.1 Dataset

We used two extreme-scale multi-label datasets benchmarks: Amazon-670K and Wiki-500K (Table 1). All these datasets are downloaded from XMLRepository.

3.2 Evaluation Metric

We chose \( P@k \) (Precision at \( k \)) and \( nDCG@k \) (normalized Discounted Cumulative Gain at \( k \)) as our evaluation metrics for performance comparison, since both \( P@k \) and \( nDCG@k \) are widely used for evaluation methods for multi-label classification problems. \( P@k \) is defined as follows:

\[
P@k = \frac{1}{k} \sum_{l=1}^{k} y_{\text{rank}(l)}
\]

where \( y \in \{0, 1\}^L \) is the true binary vector, and \( \text{rank}(l) \) is the index of the \( l \)-th highest predicted label. \( nDCG@k \) is defined as follows:

\[
\begin{align*}
DCG@k &= \sum_{l=1}^{k} y_{\text{rank}(l)} \log(l + 1) \\
iDCG@k &= \frac{1}{\min(k, ||y||_0)} \sum_{l=1}^{\min(k, ||y||_0)} 1 \\
nDCG@k &= \frac{DCG@k}{iDCG@k}
\end{align*}
\]

\( nDCG@k \) is a metric for ranking, meaning that the order of top \( k \) prediction was considered in \( nDCG@k \) but not in \( P@k \). Note that \( P@1 \) and \( nDCG@1 \) are the same.

3.3 Performance

Table 2 and Table 3 show performance comparison of HAXMLNet and several state-of-the-art methods on Amazon-670K and Wiki-500K respectively. Note that Parabel here used 3 trees while HAXMLNet only used 1 tree. On Wiki-500K, HAXMLNet reaches the best performance on all experimental settings. Because of high sparsity of labels in Amazon-670K (the most frequent label only has less than 1,900 positive samples), performance of HAXMLNet is worse than DiSMEC and Parabel, but still have a competitive performance (better than PfastreXML).

4 Conclusion

In this paper, we propose HAXMLNet, a hierarchical label-wise attention network, which can solve extreme multi-label text classification efficiently and effectively. HAXMLNet uses a hierarchical structure with multi-label attention to reduce the scale and complexity of the state of the art method AttentionXML, and make a competitive performance with other state of the art methods like Parabel [5] and DiSMEC [1]. Future work will focus on the new algorithms to improve the performance on extreme-scale multi-label classification.

[1] http://manikvarma.org/downloads/XC/XMLRepository.html
Table 2: Performance comparison of HAXMLNet and other competing methods on Amazon-670K.

| Methods    | Prec@1 | Prec@3 | Prec@5 | nDCG@1 | nDCG@3 | nDCG@5 |
|------------|--------|--------|--------|--------|--------|--------|
| AnnexML[6] | 42.08  | 36.66  | 32.74  | 42.08  | 38.81  | 36.78  |
| PfastreXML[4] | 36.84  | 34.24  | 32.09  | 36.84  | 36.02  | 35.43  |
| DiSMEC[1]   | 45.40  | 40.42  | 36.97  | 45.40  | 42.83  | 41.37  |
| Parabel[5]  | 44.92  | 39.77  | 35.98  | 44.92  | 42.11  | 40.34  |
| XML-CNN[3]  | 33.41  | 30.00  | 27.41  | 33.41  | 31.78  | 30.67  |
| HAXMLNet    | 41.09  | 36.39  | 32.65  | 41.09  | 38.49  | 36.64  |

Table 3: Performance comparison of HAXMLNet and other competing methods on Wiki-500K.

| Methods    | Prec@1 | Prec@3 | Prec@5 | nDCG@1 | nDCG@3 | nDCG@5 |
|------------|--------|--------|--------|--------|--------|--------|
| AnnexML[6] | 64.22  | 43.15  | 32.79  | 64.22  | 54.32  | 52.25  |
| PfastreXML[4] | 56.25  | 37.32  | 28.16  | 56.25  | 47.14  | 45.05  |
| DiSMEC[1]   | 64.77  | 45.14  | 35.08  | 64.77  | 55.85  | 54.17  |
| Parabel[5]  | 68.79  | 49.57  | 38.64  | 68.79  | 60.54  | 58.60  |
| HAXMLNet    | 70.44  | 52.42  | 41.12  | 70.44  | 62.91  | 60.80  |

References

[1] R. Babbar and B. Schölkopf. DiSMEC: distributed sparse machines for extreme multi-label classification. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, pages 721–729. ACM, 2017.

[2] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hullermeier. Extreme f-measure maximization using sparse probability estimates. In International Conference on Machine Learning, pages 1435–1444, 2016.

[3] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. Deep learning for extreme multi-label text classification. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 115–124. ACM, 2017.

[4] Y. Prabhu, A. Kag, S. Gopinath, K. Dahiya, S. Harsola, R. Agrawal, and M. Varma. Extreme multi-label learning with label features for warm-start tagging, ranking & recommendation. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pages 441–449. ACM, 2018.

[5] Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In Proceedings of the 2018 World Wide Web Conference on World Wide Web, pages 993–1002. International World Wide Web Conferences Steering Committee, 2018.

[6] Y. Tagami. AnnexML: approximate nearest neighbor search for extreme multi-label classification. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 455–464. ACM, 2017.

[7] R. You, S. Dai, Z. Zhang, H. Mamitsuka, and S. Zhu. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. arXiv preprint arXiv:1811.01727, 2018.