Does Tail Label Help for Large-Scale Multi-Label Learning?

Tong Wei and Yu-Feng Li, *Member, IEEE*

***Abstract*— Large-scale multi-label learning (LMLL) annotates relevant labels for unseen data from a huge number of candidate labels. It is perceived that labels exhibit a long tail distribution in which a significant number of labels are tail labels. Most previous studies consider that the performance would benefit from incorporating tail labels. Nonetheless, it is not quantified how tail labels impact the performance. In this article, we disclose that whatever labels are randomly missing or misclassified, the impact of labels on commonly used LMLL evaluation metrics (Propensity Score Precision (PSP)@k and Propensity Score nDCG (PSnDCG)@k) is directly related to the product of the label weights and the label frequencies. In particular, when labels share equal weights, tail labels impact much less than common labels due to the scarcity of relevant examples. Based on such observation, we propose to develop low-complexity LMLL methods with the goal of facilitating fast prediction time and compact model size by restraining less performance-influential labels. With the consideration that discarding labels may cause the loss of predictive capability, we further propose to preserve dominant model parameters for the less performance-influential labels. Experiments clearly justify that both the prediction time and the model size are significantly reduced without sacrificing much predictive performance.**

***Index Terms*— Large-scale multi-label learning (LMLL), performance metric, scalability, tail label.**

I. Introduction

LARGE-SCALE multi-label learning (LMLL) [1], [2] aims to annotate objects with the relevant labels from an extremely large number of candidate labels. LMLL recently owns many real-world applications. For example, in webpage categorization [3], millions of labels (categories) are collected in Wikipedia, and one needs to annotate a new webpage with relevant labels from such a big candidate set; in image annotation [4], millions of people tags are in the repository and one wishes to tag each individual picture from such a big candidate tags; in recommendation system [5], millions of items are presented and one hopes to make informative personalized recommendations from the big candidate items; Because of the high dimensionality of label space, traditional multi-label learning approaches, such as [6]–[8], become infeasible, and new algorithms are required. Recently, LMLL has been drawing increasing attention and many approaches have been proposed [9]–[25].

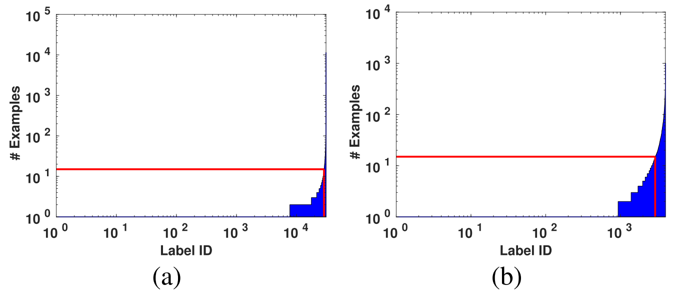

Fig. 1. Number of examples for each label is presented on (a) Wiki10 and (b) EUR-Lex data sets. The horizontal axis indicates the indices of labels, while the vertical axis indicates the number of associated examples in the training data. The vertical red line indicates that labels to the left of it (more than 70%) occur in at most 15 examples on each data set.

In LMLL, one important statistical characteristic is that labels follow a power-law distribution as illustrated in Fig. 1. Infrequently occurring labels (referred to as *tail labels*) possess limited training examples and are harder to predict than frequently occurring ones (referred to as *common labels*). How does the tail label impact the performance? It turns out that this intrinsic question has less discussion in most LMLL studies although tail labels have recently attracted increasing attention [20], [26]. Existing approaches believe that the ultimate performance would benefit from leveraging tail labels [13], [15], [16]. Conventional approaches take all labels into account to train models and make predictions over the entire label set in which a significant fraction of labels are tail labels.

To answer this question, in this article, we compute the impact of labels on commonly used LMLL evaluation metrics (Propensity Score Precision (PSP)@k and Propensity Score nDCG (PSnDCG)@k). Through analyzing the scenarios that labels are missing and misclassified, our analyses consistently show that the impact on commonly used evaluation metrics is directly related to the product of the label weights and the label frequencies. This explicitly discloses that, when labels share equal weights, such as in evaluation metrics P@k and nDCG@k, tail labels impact much less than common labels. Intuitively, as illustrated in Fig. 2, by trimming off 50% labels with fewest relevant training examples, the performance is not sacrificed, while both the prediction time and the model size are clearly reduced.

Manuscript received November 2, 2018; revised April 26, 2019 and July 8, 2019; accepted August 2, 2019. This work was supported in part by the National Key R&D Program of China under Grant 2017YFB1002201, in part by the National Natural Science Foundation of China under Grant 61772262, and in part by the Fundamental Research Funds for the Central Universities under Grant 020214380053. *(Corresponding author: Yu-Feng Li.)*

The authors are with National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China (e-mail: weit@lamda.nju.edu.cn; liyf@lamda.nju.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2935143

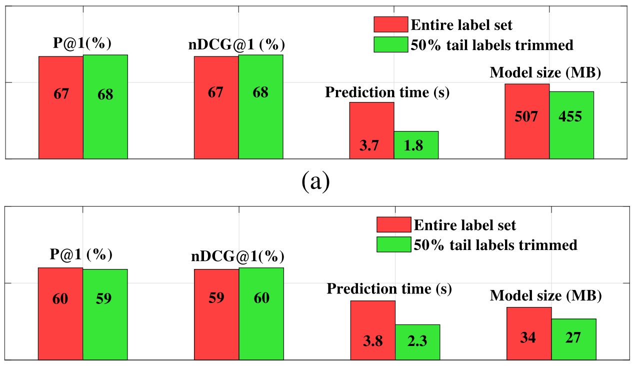

Fig. 2. Performance of state-of-the-art LMLL method (LEML [11]) on (a) Wiki10 and (b) EUR-Lex data sets with entire label set and label set after trimming off 50% tail labels. Performance do not deteriorate while the prediction time and the model size are reduced.

Base on such observation, we propose to develop low-complexity LMLL methods with the goal of facilitating fast prediction and compact models through restraining less performance-influential labels. First, as tail labels have only a few relevant instances, it is challenging to learn accurate models based on such biased label distribution. Second, we prove that trimming off less performance-influential labels has little impact on performance in terms of the commonly used LMLL metrics (PSP@k and PSnDCG@k). Third, after trimming off less performance-influential labels, both prediction time and model size can be effectively reduced, which benefits to many applications especially when fast prediction speed and compact model size are desired. With further consideration that discarding labels may cause the loss of predictive capability, we propose low-complexity methods by preserving the most significant model parameters for less performance-influential labels. Experiments clearly demonstrate the promising performance of our proposed algorithms. Simultaneously, both the prediction time and the model size are evidently reduced.

The rest of this article is arranged as follows. Section II briefly introduces some related works in LMLL. Section III introduces commonly used LMLL performance metrics. Section IV studies the usefulness of tail labels in LMLL and presents the proposed methods, which restrain less performance-influential labels to facilitate fast prediction and compact models. Section V compares the performance of our proposal with state-of-the-art approaches. Section VI discusses more metrics, which might be more suitable to measure the performance on tail labels. Section VII concludes this article.

II. Related Work

This article is mostly related to three branches of studies.

A. Tail Label of LMLL

Recently, there are some discussions on the power-law distribution in LMLL. Bhatia *et al.* [13] learned embeddings for tail labels which captures nonlinear label correlations by preserving the pairwise distances between label vectors. Jain *et al.* [16] explained that infrequently occurring tail labels are harder to predict than frequently occurring ones since they have little training examples. Xu *et al.* [15] treated tail labels as outliers and decomposed the label matrix into a low-rank matrix which depicts label correlations and a sparse one capturing the influence of tail labels. Wang and Hebert [27] and Wang [28] cast the tail label problem as transfer learning by transferring knowledge from the data-rich head classes to the data-poor tail classes. Li *et al.* [29] handled the long-tail recommendation problem. They decomposed the recommendations into two parts, a low-rank part to address short-head items and a sparse part to handle long-tail items. Liu *et al.* [30] proposed easy-to-hard learning paradigms for multi-label classification to automatically identify easy and hard labels. Tail labels are supposed to be identified as hard labels because of the scarcity of positive training examples. Babbar and Schölkopf [20] regarded this phenomenon as a setup in which an adversary is generating test examples such that the features of the test instances are quite different from those in the training set. Opposed to [16], they claimed that hamming loss is a more suitable loss function to optimize in LMLL scenario since it treats tail labels equally with common labels. Most of these studies believe that the ultimate performance would benefit from leveraging tail labels.

B. Prediction Time of LMLL

In many real-world applications such as recommender systems and search engines, LMLL algorithms are supposed to respond in an extremely limited time for good user experience.

The vast amount of effort has been made to reduce prediction time. For example, embedding-based methods aim to project label vectors onto a low-dimensional space based on the assumption that label matrix is low-rank [2], [17], [31]–[36]. In addition to being able to capture the label correlation, these methods also exhibit strong generalization guarantees. Sparse Local Embeddings for Extreme multi-label Classification (SLEEC) [13] takes a further step to scale up to more labels. It first clusters the data into smaller regions. It subsequently performs local embeddings of label vectors by preserving distances to nearest label vectors learned using a k-nearest neighbor classifier. Another recent line of research is tree-based methods which recursively divide the space of labels or features to achieve fast prediction speed [12], [37]–[40]. FastXML [12] is one of leading tree-based classifiers. It optimizes an nDCG-based ranking loss function. It recursively partitions the feature space instead of the label space and uses the observation that only a small number of labels are active in each region of feature space. Zhang *et al.* [41] learned low-dimensional representations for both instances and labels with deep neural networks. Similar to [13], they use k-nearest neighbor classifier to determine resultant labels for testing instance in the embedding space and a clustering method to speed up the prediction phase. In addition, various strategies have been adopted to tackle the scaling issue, such as parallelization strategy and subset column selection. Reference [18] is based on a regularized one-vs-rest large margin linear model. By exploiting the independence between the submodels associated with each label, computations are accelerated with parallelization of the training stage.

Boutsidis *et al.* [42] proposed to find approximate solutions of the Column Subset Selection Problem (CSSP) more efficiently. Later, Bi and Kwok [35] addressed this problem by selecting a small subset of class labels that can approximately span the original label space. This is performed by an efficient randomized sampling procedure where the sampling probability of each class label reflects its importance among all the labels. Recently, Niculescu-Mizil and Abbasnejad [43] proposed a label filters method. These label filters preselect a fairly modest set of candidate labels before the base multi-class or multi-label classifier is applied. Liu *et al.* [44] incorporated approximate nearest neighbor (ANN) methods to speed up the testing time.

C. Model Size of LMLL

Model size is an important consideration for the practicality of LMLL methods in order to be deployed on portable devices with small memory. For example, in order to enable LMLL applications such as image annotation software on the cell phone, a compact model is an essential prerequisite because of limited storage space.

Several methods were proposed to yield sparse solutions explicitly or prune spurious weights to achieve sparse models. Yen *et al.* [14] maximized the margin loss with ℓ_1 penalty and yielded an extremely sparse solution without sacrificing the expressive power of the predictor. Babbar and Schölkopf [18] proposed a framework which controls the model size by filtering out the billions of stored spurious parameters by *ad hoc* usage of the off-the-shelf solvers. By weeding out ambiguous parameters, one can obtain model size which is three orders of magnitude smaller. Liu and Tsang [45] developed a tree-based algorithm which learns a sparse hyperplane classifier in each decision node. Additionally, data distribution observations are leveraged for the early stopping of the tree model.

To the best of our knowledge, this article is the first proposal on connecting the three above aspects of LMLL together, i.e., studies on the impact of tail labels that help facilitate fast prediction time and compact model size of LMLL.

III. Common Performance Metrics in LMLL

Prior to presenting our proposed methods, we introduce two commonly used LMLL performance metrics, PSP@k and PSnDCG@k [16], [20], in this section.

A. PSP@k

The first one is Propensity Scored Precision@k (PSP@k) proposed in [16]. PSP@k is popularly used in LMLL applications, especially for ranking tasks such as information retrieval. In PSP@k, only a few top predictions of an instance will be considered. For instance, $\mathbf{x} \in \mathbb{R}^d$ where d represents the feature dimensionality, PSP@k is defined for a predicted score vector $\hat{\mathbf{y}} \in \mathbb{R}^L$ and ground truth label vector $\mathbf{y} \in \{0,1\}^L$ as

$$\text{PSP@}k(\mathbf{y}, \hat{\mathbf{y}}) := \frac{1}{k} \sum_{l \in \text{rank}_k(\hat{\mathbf{y}})} \frac{\mathbf{y}_l}{p_l}$$

where L represents the size of label set and $\text{rank}_k(\hat{\mathbf{y}})$ returns the indices of k largest value in $\hat{\mathbf{y}}$ ranked in descending order. $p_l = (1/(1 + C(N_l + B)^{-A}))$ is the propensity score for the lth label, where A, B, C are set in a heuristic manner and N_l is the number of the positive training instances. The propensities are modeled as a sigmoidal function of $\log N_l$ proposed in [16] based on empirical observations. By assigning larger rewards for accurately predicted tail labels, it is expected to remove the popularity bias.

B. PSnDCG@k

Another frequently used ranking-based performance measure in LMLL literature is propensity scored nDCG@k (PSnDCG@k) which is defined as

$$\text{PSnDCG@}k(\mathbf{y}, \hat{\mathbf{y}}) := \frac{\text{PSDCG@}k}{\sum_{l=1}^{\min(k, ||\mathbf{y}||_0)} \frac{1}{\log(l+1)}}$$

where $\text{PSDCG@}k(\mathbf{y}, \hat{\mathbf{y}}) := \sum_{l \in \text{rank}_k(\hat{\mathbf{y}})} (\mathbf{y}_l/(p_l \log(l+1)))$. In particular, when setting $p_l = 1, \forall 1 \le l \le L$, PSP@$k$ and PSnDCG@k reduce to another two popular LMLL performance metrics P@k and nDCG@k, respectively.

Notably, unlike PSP@k, PSnDCG@k takes into account the ranking of the correctly predicted labels. For instance, if there are only one of the five labels that are correctly predicted, then PSP@5 gives the same score if the correctly predicted label is at rank 1 or rank 5. On the other hand, PSnDCG@5 gives it a higher score if it is predicted as rank 1, and the lowest nonzero score if it is predicted at rank 5.

IV. Propensity Score-Based Label Restraining

In this section, we first present the proposed propensity score-based label restraining (PoLaR) through identifying and pruning less performance-influential labels. With further consideration that discarding labels may cause the loss of predictive capability, we propose an extension method PoLaR$_{\text{var}}$ by preserving the most significant model parameters for less performance-influential labels.

A. Preliminaries

Let $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1) \ldots (\mathbf{x}_N, \mathbf{y}_N)\}$ be the given training set, where $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature of the ith example and $\mathbf{y}_i \in \{0,1\}^L$ is the corresponding label vector. Let $\mathbf{Y} = [\mathbf{y}_1; \ldots, \mathbf{y}_N]$ denote the label matrix of training examples. $\mathbf{Y}_{ij} = 1$ if example $\mathbf{x}_i$ is relevant with the jth label, and 0 otherwise. LMLL aims to learn a classifier $f : \mathbb{R}^d \to \{0,1\}^L$ that predicts the label vector for unseen data. Unlike traditional multi-label learning, the label set size in LMLL is extremely large and labels usually follow a long-tail distribution.

As aforementioned, the main idea is to identify and filter out less performance-influential labels in terms of LMLL metrics until the performance degeneration constraint is violated. After that, we train LMLL models using examples of remaining labels.

To identify less performance-influential labels, we consider two widespread scenarios in LMLL, i.e., missing labels and misclassified labels. We investigate how labels impact the LMLL evaluation metrics under these two scenarios. The analyses consistently show that the performance impact of the label is directly proportional to its weight in LMLL metrics and its frequency in the training data. The analyses provide a guideline to restrain the least influential labels. In the following, we first present our analyses under *missing labels* and *misclassified labels* scenarios.

B. Usefulness of Tail Labels in LMLL

1) Randomly Missing Labels: Missing labels commonly occur in LMLL [15], [35], [36]. To quantify the impact of labels on evaluation metrics, we compute the impact of labels under the scenario that *relevant labels are randomly missing with a probability* π [46]. Without loss of generality, we use $u_j = ||\mathbf{Y}_{:,j}||_0$ $(j = 1, \ldots, L)$ to denote the number of examples which are associated with the jth label. Let $w_j = (1/p_j)$ denote the weight for the jth label defined in evaluation metrics. c_i indicates the number of relevant labels of example $\mathbf{x}_i$. We provide the result in Theorem 1.

Theorem 1: Suppose that relevant labels are randomly missing with probability π, the impact of the jth label in terms of PSP@k and PSnDCG@k is upper bounded by $(1-\pi)w_j u_j$.

Proof: Because k out of c_i relevant labels are selected in the calculation of PSP@k, which has $\binom{c_i}{k}$ distinct choices. The expected impact of the jth label is computed as follows:

$$\frac{w_j}{k}\sum_{i=1\wedge \mathbf{Y}_{ij}=1}^{N}\frac{\binom{c_i-1}{k-1}}{\binom{c_i}{k}} = (1-\epsilon)\sum_{i=1\wedge \mathbf{Y}_{ij}=1}^{N}\frac{w_j}{c_i} \le (1-\pi)w_j u_j.$$

It can be perceived that the impact of the jth label is upper bounded by the product of label weight w_j, its frequency u_j and a constant.

For PSnDCG@k, it is noteworthy that every observed label has the same rank, hence $r = (1/(\log(l+1)))$ remains a constant and we get

$$\begin{aligned}\text{PSnDCG@}k &= \frac{\text{PSDCG@}k}{\sum_{l=1}^{\min(k,\|\mathbf{y}\|_0)}\frac{1}{\log(l+1)}}\\ &= \frac{r}{\sum_{l=1}^{\min(k,\|\mathbf{y}\|_0)} r}\sum_{l\in \text{rank}_k(\hat{\mathbf{y}})}\mathbf{y}_l.\end{aligned}$$

Since L is extremely large in LMLL and $k \le 5$, we usually have $||\mathbf{y}||_0 \ge k$ and PSnDCG@k is cast as follows:

$$\begin{aligned}\text{PSnDCG@}k &= \frac{r}{\sum_{l=1}^{k} r}\sum_{l\in \text{rank}_k(\hat{\mathbf{y}})}\mathbf{y}_l\\ &= \frac{1}{k}\sum_{l\in \text{rank}_k(\hat{\mathbf{y}})}\mathbf{y}_l = \text{PSP@}k.\end{aligned}$$

As a result, the analysis for PSnDCG@k is reduced to the one for PSP@k. □

2) Randomly Misclassified Labels: Another common scenario is label misclassification [47], [48]. We compute the impact of labels under the scenario that *labels are randomly misclassified with probability* π, and similarly, reveal the impact of labels on the LMLL evaluation metrics in Theorem 2.

Theorem 2: Suppose that labels are randomly misclassified with probability π, the impact of the jth label on PSP@k and PSnDCG@k is upper bounded by $((1-\pi)/((L-2)\pi+1))w_j u_j$.

Proof: For PSP@k, there are $v_i = (1-\pi)c_i + \pi(L-c_i)$ relevant labels in the predicted label vector. By choosing a random subset of size k from v_i labels, the influence of the jth label to PSP@k can be computed as

$$\frac{w_j}{k}\sum_{i=1\wedge \mathbf{Y}_{ij}=1}^{N}\frac{\binom{v_i-1}{k-1}}{\binom{v_i}{k}} = \sum_{i=1\wedge \mathbf{Y}_{ij}=1}^{N}\frac{w_j}{v_i} \le \frac{(1-\pi)w_j u_j}{(L-2)\pi+1}.$$

This depicts that the impact of the jth label is upper bounded by $((1-\pi)/((L-2)\pi+1))w_j u_j$. It is intriguing to observe that the upper bound of misclassified labels is smaller than that of missing labels. The reason is owing to the fact that missing labels in the missing label scenario are assumed to be relevant labels, while misclassified labels might be irrelevant. Therefore misclassified labels have little impact to PSP@k.

For PSnDCG@k, similar to the proof in Theorem 1, the proof for PSnDCG@k is reduced to the one in PSP@k and we obtain same conclusive remark for PSnDCG@k. □

With the analyses above, in both label-missing and label-misclassified scenarios, we conclude that the impact of labels in terms of frequently used LMLL metrics (PSP@k and PSnDCG@k) is proportional to the product of the label weights and the label frequencies.

C. Less Performance-Influential Labels Removing

Based on the analyses, we rank the labels according to the value of $w_j u_j$, $j = \{1, \ldots, L\}$ in ascending order and filter out labels with little performance influence. Because of the significantly large number of labels, it is very expensive to discard labels one by one until the performance deterioration constraint is violated. To this end, a binary search is developed to efficiently determine the cutoff threshold based on the observation that the performance is monotonically decreasing as the number of removed labels increases. By performing this, the computational cost is reduced from $O(L)$ to $O(\log L)$. Algorithm 1 summarizes the elaborate procedure of binary search for label removing.

In the following, we analyze the prediction time and model size complexities of the resultant model after employing the proposed strategy. We first list the prediction time complexity and memory complexity of LMLL methods in Table I. For embedding-based methods, r denotes the low-rank parameter. For tree-based methods, T denotes the size of the ensemble and M_t is the average number of nodes in the trees. $\hat{L}$ denotes the number of nonzero elements in the label vector of each

TABLE I
SUMMARY OF THE PREDICTION TIME AND MEMORY COMPLEXITIES OF LMLL METHODS

Method	Prediction time complexity	Memory complexity
Embedding-based	$r(d+L)+L\log L$	$r(d+L)$
Tree-based	$O(TdH+T\hat{L}+\hat{L}\log\hat{L})$	$O(TM_t\hat{L}+TdM_t)$
Binary Relevance	$O(dL)$	$O(dL)$

Algorithm 1 Less Performance-Influential Labels Removing

Input: propensity score w; data $\mathcal{D}$; performance of model trained on the entire label set $p^\star$; performance reduction tolerance ϵ

Output: model $\mathbf{M}$ trained on the most performance-influential labels

```
1: sort labels according to $w_l \times N_l$ in ascending order
2: $L'_{\text{left}} = 1$, $L'_{\text{right}} = L$
3: while $L'_{\text{right}} - L'_{\text{left}} > 1$ do
4:     $L'_{\text{curr}} = (L'_{\text{left}} + L'_{\text{right}})/2$
5:     $\mathbf{M} = \text{train}(\mathcal{D}, L'_{\text{curr}})$
6:     if $perf(\mathbf{M}) \geq p^\star - \epsilon$ then
7:         $L'_{\text{left}} = L'_{\text{curr}}$
8:     else
9:         $L'_{\text{right}} = L'_{\text{curr}}$
10:    end if
11: end while
12: return $\mathbf{M} = \text{train}(\mathcal{D}, L'_{\text{left}})$
```

leaf node on average and H represents the average length of the path traversed by instance $\mathbf{x}$ to reach the leaf node in the T trees. It is effortless to see that as the number of labels L decreases, both prediction time and memory complexities reduce. To minimize the performance loss, we propose to identify less performance-influential labels by computing label impact on commonly used LMLL performance metrics.

Next, we explain the complexities of the proposed approach. Let L' denote the number of less performance-influential labels and $L' \ll L$ is consistently observed in the experiments. For embedding-based methods, the prediction time and model size complexities reduce to $O(r(d + L') + L' \log L')$ and $O(r(d + L'))$, respectively. For tree-based methods, both $\hat{L}$, M_t and H decrease as the label set becomes smaller. If the trees are balanced, then $H = \log N \approx \log L$. The average cost of prediction becomes $O(Td \log L')$. Therefore, the overall prediction time and model size complexities are reduced.

D. Less Performance-Influential Labels With Discriminant Parameter Preserving

Although directly discarding less performance-influential labels can bring significant reduction on model size and prediction time; however, it is not always desirable owing to the loss of predictive capability. To this end, we propose to preserve discriminant model parameters for performance-influential labels. Concretely, given a pretrained model with model parameters arranged in a matrix $\mathbf{M} \in \mathbb{R}^{d \times L}$ (such as [18], [20], [43]) where d and L are the dimensionality of feature space and label space, respectively, we ensure that at least δ parameters are preserved for each label, i.e., $||\mathbf{M}_{:,j}||_0 \geq \delta, j = 1, \ldots, L$. Ultimately, we fix the most discriminant label parameters such that the predictive capabilities for less performance-influential labels are kept. By doing this, the prediction time and model size complexities are both reduced from $O(dL)$ to $O(dL' + \delta(L - L'))$. The elaborate procedure is summarized in Algorithm 2.

Algorithm 2 Less Performance-Influential Labels With Discriminant Parameter Preserving

Input: pretrained model $\hat{\mathbf{M}}$; inverse propensities $\mathbf{w}$; data $\mathcal{D}$; performance of model trained on the entire label set $\hat{p}$; performance reduction tolerance ϵ; number of parameters to preserve δ

Output: model $\mathbf{M}$ with discriminant parameters preserved for less performance-influential labels

```
1: sort labels according to $w_l \times N_l$ in ascending order
2: preserve $\delta$ parameters with largest absolute value in $\hat{\mathbf{M}}_{:,j}$,
   for $1 \leq j \leq L$, and store the resultant model as $\mathbf{M}'$
3: call Algorithm 1 and obtain $\tilde{\mathbf{M}}$, as well as the number of
   less performance-influential labels $L'$
4: $\mathbf{M} = [\mathbf{M}'_{:,1:L'}; \tilde{\mathbf{M}}]$
5: return $\mathbf{M}$
```

TABLE II
DATA SET STATISTICS

Data set	Train N	Features D	Labels L	Test M	Avg. labels per point	Avg. points per label
Bibtex	4,880	1,836	159	2,515	2.40	111.71
Delicious	12,920	500	983	3,185	19.03	311.61
EUR-Lex	15,539	5,000	3,993	3,809	5.31	25.73
Wiki10	14,146	101,938	30,938	6,616	18.64	8.52

V. Experiments

To validate our theoretical findings and efficacy of the proposed approach PoLaR and its variant PoLaR$_{\text{var}}$, we conduct experiments with two leading embedding-based method LargE-scale Multi-Label learning (LEML) [11], SLEEC [13], a state-of-the-art tree-based method FastXML [12], a classic multi-label learning approach Binary Relevance (BR) [49], and two model compression methods Primal and Dual Sparse approach to multiclass and multilabel classification (PD-Sparse) [14] and Distributed Sparse Machines for Extreme multi-label Classification (DiSMEC) [18]. Experiments are carried out on four LMLL benchmark data sets. The comprehensive statistics are listed in Table II. All the data sets and implementation of LEML and FastXML are publicly available and can be downloaded from the Extreme Classification Repository.[1]

Because of computational reasons, we only report results on one train/test split of each data set. Because multiple train/test splits would mean multiple times more training time and, in LMLL setting, training is expensive on large data sets. Moreover, it allows fair comparisons for different methods by training and testing on the same train/test split, which is available in the Extreme Classification Repository.

A. Propensity Score-Based Label Removing

To verify the effectiveness of our propensity score-based label-removing strategy, we compare the results in terms of PSP@k and PSnDCG@k using an embedding-based method

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

TABLE III

COMPARISON WITH LEML IN TERMS OF MODEL SIZE, PREDICTION TIME, PSP@k (%), AND PSNDCG@k (%)

Data set		LEML	POLAR	Reduction over LEML
Bibtex	Model size	0.76 MB	0.74 MB	2.71 %
	Prediction time	0.31 s	0.20 s	37.05 %
	PSnDCG@1	44.55	44.51	0.11 %
	PSnDCG@3	44.73	43.97	1.71 %
	PSnDCG@5	47.17	46.01	2.45 %
	PSP@1	44.55	44.51	0.11 %
	PSP@3	45.00	43.98	2.28 %
	PSP@5	49.54	47.80	3.51 %
Delicious	Model size	2.26 MB	1.08 MB	52.33 %
	Prediction time	0.02 s	0.01 s	50.00 %
	PSnDCG@1	30.74	29.86	2.89 %
	PSnDCG@3	32.03	31.15	2.73 %
	PSnDCG@5	32.46	31.32	3.52 %
	PSP@1	30.74	29.86	2.89 %
	PSP@3	32.44	31.55	2.72 %
	PSP@5	32.95	31.66	3.91 %
EUR-Lex	Model size	34.31 MB	20.83 MB	39.27 %
	Prediction time	3.85 s	0.26 s	93.37 %
	PSnDCG@1	21.31	20.81	2.33 %
	PSnDCG@3	23.70	22.91	3.29 %
	PSnDCG@5	24.88	23.64	4.97 %
	PSP@1	21.31	20.81	2.33 %
	PSP@3	24.56	23.65	3.69 %
	PSP@5	26.34	24.69	6.26 %
Wiki10	Model size	506.88 MB	390.04 MB	23.05 %
	Prediction time	3.67 s	0.10 s	97.28 %
	PSnDCG@1	8.83	8.68	1.70 %
	PSnDCG@3	9.37	8.98	3.13 %
	PSnDCG@5	9.61	8.86	7.80 %
	PSP@1	8.83	8.68	1.70 %
	PSP@3	9.24	8.73	5.52 %
	PSP@5	9.42	9.09	3.50 %

TABLE IV

COMPARISON WITH FASTXML IN TERMS OF MODEL SIZE, PREDICTION TIME, PSP@k (%), AND PSNDCG@k (%)

Data set		FastXML	POLAR	Reduction over FastXML
Bibtex	Model size	0.76 MB	0.74 MB	2.71 %
	Prediction time	1.76 s	1.75 s	0.36 %
	PSnDCG@1	49.78	47.63	4.31 %
	PSnDCG@3	51.82	49.60	4.29 %
	PSnDCG@5	54.94	52.20	4.99 %
	PSP@1	49.78	47.63	4.31 %
	PSP@3	52.90	50.64	4.28 %
	PSP@5	58.62	55.39	5.51 %
Delicious	Model size	2.26 MB	1.08 MB	52.33 %
	Prediction time	6.67 s	3.96 s	40.63 %
	PSnDCG@1	32.05	31.52	1.65 %
	PSnDCG@3	33.92	32.62	3.83 %
	PSnDCG@5	34.66	32.68	5.71 %
	PSP@1	32.05	31.52	1.65 %
	PSP@3	34.49	32.97	4.41 %
	PSP@5	35.42	32.94	7.00 %
EUR-Lex	Model size	34.31 MB	20.83 MB	39.27 %
	Prediction time	13.21 s	12.41 s	6.05 %
	PSnDCG@1	26.90	26.41	1.80 %
	PSnDCG@3	32.18	31.03	3.55 %
	PSnDCG@5	35.42	33.14	6.44 %
	PSP@1	26.90	26.41	1.80 %
	PSP@3	34.17	32.80	4.00 %
	PSP@5	39.14	35.89	8.30 %
Wiki10	Model size	506.88 MB	390.04 MB	23.05 %
	Prediction time	48.33 s	31.42 s	34.99 %
	PSnDCG@1	9.80	9.73	0.71 %
	PSnDCG@3	10.08	9.77	3.08 %
	PSnDCG@5	10.33	9.59	7.21 %
	PSP@1	9.80	9.73	0.71 %
	PSP@3	10.17	9.77	3.99 %
	PSP@5	10.54	9.48	10.04 %

LEML [11] and a tree-based method FastXML [12] as the base models. As suggested by Jain *et al.* [16], we set $A = (0.5 + 0.6)/2 = 0.55, B = (0.4 + 2.6)/2 = 1.5, \epsilon = 1\%$ on all data sets. From Table III, POLAR saves prediction time and model size on all data sets compared with LEML. On the aspect of prediction time, POLAR saves more than 90% prediction time on two large data sets, i.e., EUR-Lex and Wiki10. On the aspect of model size, POLAR reduces more than 50% and 39% on Delicious and EUR-Lex, respectively. This is because the success of LEML mainly depends on the low-rank assumption, which tends to be violated because of the presence of tail labels. Therefore, tail labels make an extremely limited contribution to the performance of LEML. POLAR provides an appropriate approach to preserve the validity of low-rank assumption by elegantly trimming off the less performance-influential labels.

The results for FastXML are presented in Table IV. Similarly, POLAR achieves comparable results, meanwhile saves prediction time and model size on all data sets. On the aspect of prediction time, POLAR saves more than 40% and 34% prediction time on Delicious and Wiki10. On the aspect of model size, POLAR reduces more than 50% and 39% on Delicious and EUR-Lex, respectively. The reason lies in the fact that the number of split partitions and the depth of trees during the training process are both reduced as the number of labels decreases; therefore, the size of tree model and the prediction time spent on leaf nodes are cut down consequently.

TABLE V

PERFORMANCE COMPARISON IN TERMS OF PSP@K (%) AND PSNDCG@K (%) WITH RANDOM SAMPLING (RANDS) ON DELICIOUS DATA SET. THE RANDOM SAMPLING METHOD WAS RUN TEN TIMES AND THE AVERAGE NUMBERS ARE REPORTED

Method	PSP@1	PSP@3	PSP@5	PSnDCG@1	PSnDCG@3	PSnDCG@5
RandS	22.67	25.51	27.94	22.67	26.45	29.90
POLAR	32.08	33.59	33.47	32.08	33.30	33.29

We also compare our method with random sampling method, which randomly removes an equal number of labels

TABLE VI

PERFORMANCE COMPARISONS BETWEEN THE PROPOSED POLAR$_{\text{var}}$, BR, DISMEC, AND PD-SPARSE IN TERMS OF PSP@k (%), PSNDCG@k (%), AND MODEL SIZE WITH $\delta = 5$ AND $\epsilon = 1\%$. "-" INDICATES THE RESULT IS NOT AVAILABLE

Data set		PSP@1	PSP@3	PSP@5	PSnDCG@1	PSnDCG@3	PSnDCG@5	Model size
Bibtex	BR	50.70	53.66	59.34	50.70	52.71	55.80	1.15 MB
	DiSMEC	50.20	52.20	58.60	50.20	52.00	55.70	0.71 MB
	PD-Sparse	48.34	48.77	52.93	48.34	48.49	52.93	20.00 KB
	POLAR$_{var}$	50.71	53.30	58.86	50.71	52.39	55.41	0.59 MB
Delicious	BR	32.14	33.59	33.43	32.14	33.32	33.28	7.18 MB
	DiSMEC	-	-	-	-	-	-	-
	PD-Sparse	25.22	24.63	23.85	25.22	24.80	24.25	0.25 MB
	POLAR$_{var}$	32.08	33.59	33.47	32.08	33.30	33.29	1.26 MB
EUR-Lex	BR	39.93	45.86	49.74	39.93	44.24	46.83	156.38 MB
	DiSMEC	41.20	45.40	49.30	41.20	44.30	46.90	79.86 MB
	PD-Sparse	38.28	42.00	44.89	38.28	40.96	42.84	25.00 MB
	POLAR$_{var}$	40.06	46.02	49.91	40.06	44.42	47.01	20.18 MB
Wiki10	BR	13.57	13.10	13.96	13.60	13.82	13.97	$\approx$ 870 GB
	DiSMEC	13.60	13.10	13.80	13.60	13.20	13.60	880.00 MB
	PD-Sparse	-	-	-	-	-	-	-
	POLAR$_{var}$	13.53	13.10	13.46	13.53	13.65	13.67	67.50 MB

with POLAR. Table V depicts that random sampling usually results in severe degeneration in terms of all six performance measures. The empirical results demonstrate that POLAR can accurately identify less performance-influential labels on different data sets and trades off negligible predictive performance for a significant reduction in prediction time and model size.

B. Propensity Score-Based Label Restraining

In applications where removing labels is undesirable, we show that POLAR$_{\text{var}}$ performs well. In this experiment, we set $\delta = 5$ and $\epsilon = 1\%$. In addition to state-of-the-art LMLL approaches (LEML and FastXML), we try to enable classic multi-label learning methods, such as BR, on large-scale data sets. For BR, it suffers two aspects of disadvantages. First, as BR trains a binary classifier for each label, which is computationally and memory intensive on large data sets. Second, the prediction speed is very slow because it examines each label to distinguish that label from the rest. Third, as it learns a weight vector for each label, the space consumption is extremely large. In the following, we apply the proposal on BR and resolve the above-mentioned obstacles.

As it is well known that the prediction speed of BR is proportional to the number of labels, our proposal clearly reduces the prediction time significantly since many less performance-influential labels are removed. Therefore, we only look at the effectiveness of POLAR$_{\text{var}}$ in terms of reduction in model size. Comparison results with the plain BR and two leading model compression methods DiSMEC [18], PD-Sparse [14] are depicted in Table VI. Inspired by Babbar and Schölkopf [18] that a large fraction of weights lie close to 0, we shed weight which lie in $[-0.01, 0.01]$ as suggested in the article. On relatively small data sets, i.e., Bibtex, POLAR$_{\text{var}}$ reduces above 48% model size and loses no more than 1% performance in terms of six metrics. Considering that label distribution on Bibtex is relatively balanced and hence only a few labels can be pruned, otherwise resulting in serious performance deterioration. On large data sets, we obtain a significant reduction in model size with negligible generalization performance deterioration.

We also compare POLAR$_{\text{var}}$ with state-of-the-arts on two large data sets in Figs. 3 and 4. Although POLAR$_{\text{var}}$ is built on the BR scheme, it achieves comparable or even smaller model size compared to state-of-the-art approaches. However, in terms of predictive performance, solvers relied on structural assumptions such as FastXML (tree), LEML (low-rank), and SLEEC (piecewise-low-rank) do not perform as well as POLAR$_{\text{var}}$ in most cases. This may owe to the fact that low-rank or tree assumption does not exactly hold in these data sets. On the aspect of model size, we can see that POLAR$_{\text{var}}$ gets an order of magnitude smaller model size than FastXML and SLEEC.

This not only shows that POLAR$_{\text{var}}$ retains the predictive capabilities on the less performance-influential labels very well by preserving discriminant label parameters but also brings marginal model size reduction. Notably, however, that the POLAR$_{\text{var}}$ can be used in conjunction with many multi-label classifiers, and hence better performance can be obtained by replacing BR with advanced classifiers (e.g., [18], [20]).

C. Extreme Case Study

When label weight equals 1, PSP@k and PSnDCG@k reduce to P@k and nDCG@k, respectively. We illustrate the effectiveness of POLAR in Table VII using BR as the base model. It is effortless to see that significant reduction in

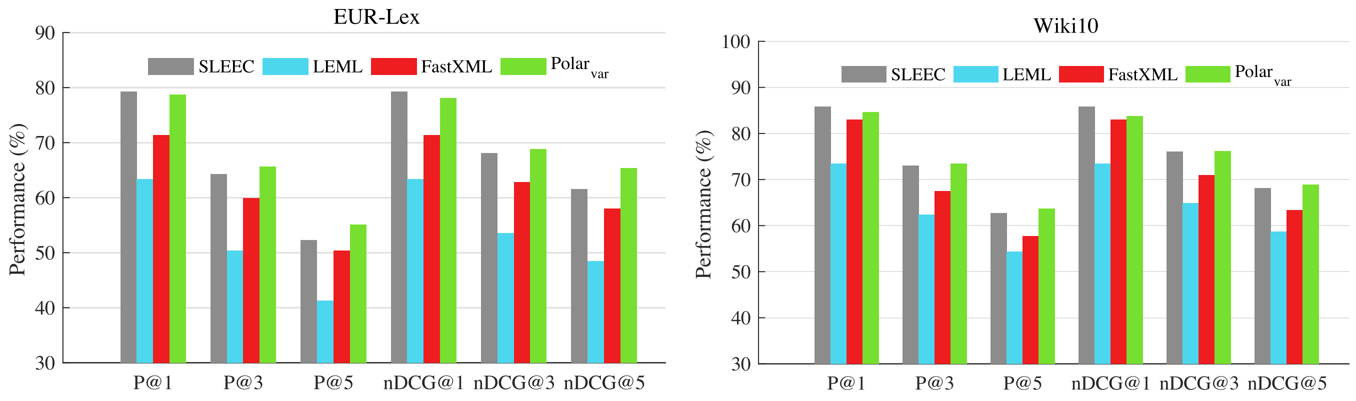

Fig. 3. Comparison with existing algorithms on EUR-Lex and Wiki10 in terms of nDCG@k (%) and P@k (%).

TABLE VII

PERFORMANCE COMPARISON BETWEEN POLAR AND BR IN TERMS OF P@k (%), NDCG@k (%), PREDICTION TIME, AND MODEL SIZE. ↓ INDICATES PREDICTION TIME OR MODEL SIZE REDUCTION OVER BR

Data set		P@1	P@3	P@5	nDCG@1	nDCG@3	nDCG@5	Prediction time	Model size
Bibtex	BR	62.94	39.40	28.98	62.94	59.68	62.19	0.13 s	1.15 MB
	POLAR	62.86	38.78	28.25	62.15	59.54	61.90	0.08 s (38.46% ↓)	0.70 MB (39.13% ↓)
Delicious	BR	61.22	56.99	52.68	61.22	58.07	54.92	2.82 s	7.18 MB
	POLAR	60.78	56.81	51.94	60.72	57.77	54.86	1.12 s (60.28% ↓)	4.70 MB (34.54% ↓)
EUR-Lex	BR	78.81	65.69	55.22	78.81	69.12	64.13	8.42 s	156.01 MB
	POLAR	78.76	65.67	55.16	78.13	68.80	65.37	4.23 s (49.76% ↓)	59.42 MB (61.91% ↓)
Wiki10	BR	85.03	74.10	64.96	84.00	77.02	70.40	$>$ 24 h	$\approx$ 870 GB
	POLAR	84.68	73.51	63.67	83.80	76.20	68.91	2.62 h (>90% ↓)	14.10 GB (98.38% ↓)

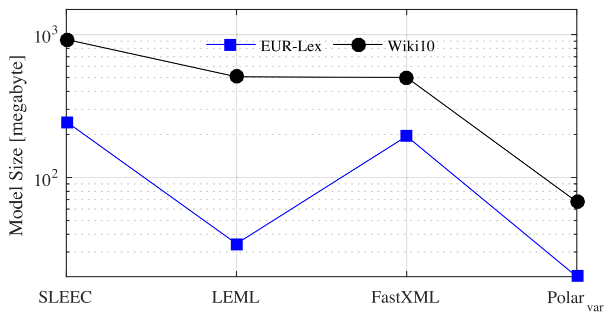

Fig. 4. Comparison with state-of-the-art approaches on EUR-Lex and Wiki10 data sets in terms of model size.

both prediction time and model size is achieved thanks to the long-tail distribution. Note that, both P@k and nDCG@k remain comparable with the plain BR. This demonstrates that POLAR is effective in applications where labels share equal importance and in such extreme cases tail labels help facilitate fast prediction speed and compact model size.

D. Parameter Study

We also investigate how different values of ϵ impact the predictive performance and the model size when using BR as the base model. Fig. 5 demonstrates that the performance deteriorates as ϵ increases because ϵ determines how many tail labels would be discarded. Therefore, it deteriorates the performance when too many informative labels are pruned. On the aspect of model size, POLAR is able to reduce 80% model size even when $\epsilon = 1\%$. Although more significant reduction can be gained with a larger value of ϵ, it comes at the cost of losing generalization performance. In practice, we set $\epsilon = 1\%$ as it was observed to yield good performance.

E. Comparison Between Reductions in Prediction Time and Model Size

From the experimental results, one intriguing observation is that the reduction on model size is not as significant as that on prediction time. To figure out the reasons, we take LEML [11] as an example of the analysis. As an embedding-based method, the label matrix is first projected onto a low-dimensional space. Without loss of generality, we assume the dimensionality of the projection space is r, and the learned projection matrices $\mathbf{W}$ and $\mathbf{H}$ have size $d \times r$ and $r \times L$, respectively, where d is the number of features, r is the low-rank parameter and L is the number of labels. Consequently, as r is very small, reducing L by trimming off less performance-influential labels will have limited reduction on model size. In terms of prediction time, it involves matrix multiplication $\mathbf{X}_t\mathbf{WH}$, where $\mathbf{X}_t$ is test data with size $N_t \times d$, where N_t is the number testing examples. Therefore, the predictive complexity is proportional to $O(N_t r(d + L))$. By removing less performance-influential labels, the prediction time reduces more effective than model size because $N_t r$ is much larger than r.

From the experimental results, we conclude that the proposed methods, POLAR and POLAR$_{\text{var}}$, are able to select the cutoff thresholds adaptively on various data sets. They can also yield fast prediction speed as well as compact model size without losing predictive capability on less performance-influential

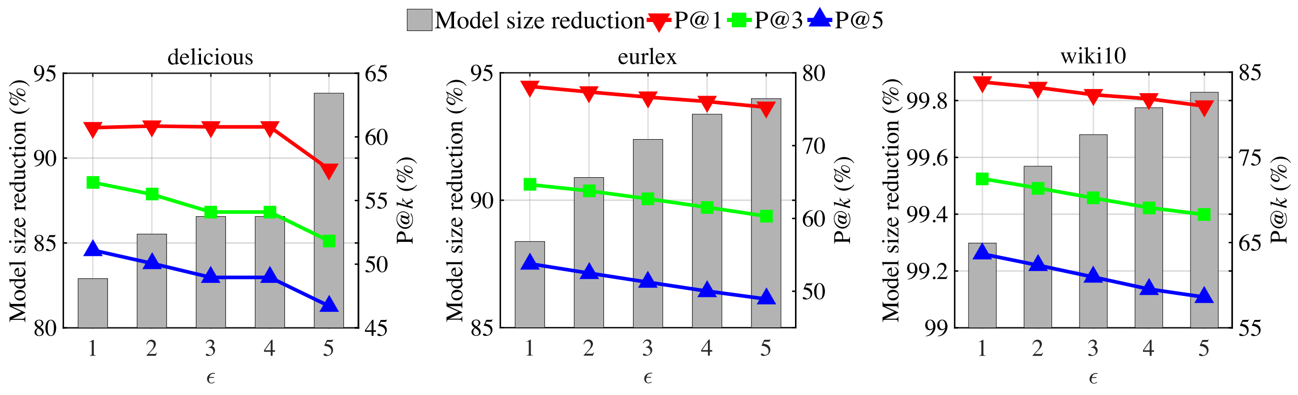

Fig. 5. Study on different value of ϵ with λ set to 0.01. x-axis: value of ϵ (%). y-axis (Left): percentage of model size reduction compared to the plain BR. y-axis (Right): P@k.

labels. We apply our strategies not only to embedding-based and tree-based approaches but also to classic multi-label learning method BR, which validates the applicability of our strategies.

VI. Discussion

Our analyses and empirical studies suggest that to evaluate the performance of LMLL methods on tail labels, the choice of the performance metric is critical because of the long tail distribution. Although the propensity score-based performance metrics (PSP@k and PSnDCG@k) proposed in [16] can, to a certain extent, alleviate this problem, the parameters A, B, and C are set in a heuristic manner when calculating label propensities for the given data set. Opposed to [16], [20] claimed that hamming loss is a more suitable loss function to optimize since it takes all labels into account during optimization instead of only the top k predictions considered in PSP@k and PSnDCG@k.

Additionally, to increase the diversity of predicted labels, we adapt the term *coverage*, which has been widely used in recommender systems to measure the diversity of recommended items [50], [51]. Coverage is defined as the fraction of items that appear in the users' recommendation lists. In the context of LMLL, we define the *coverage* as

$$\text{Coverage} = \frac{|\cup_{1\le i\le N} \mathbf{R}_i|}{L}$$

where $\mathbf{R}_i$ is the set of relevant labels of instance i correctly annotated. However, directly optimizing the *coverage* objective is very difficult, and we leave this problem for future work.

VII. Conclusion

In this article, we study how tail labels impact the frequently used LMLL performance metrics. Our analyses based on examining the performance impact for the missing labels and the misclassified labels, explicitly disclose that label impact is directly related to the label weights and the label frequencies. In a particular case where all labels share equal weights, tail labels impact much less than common labels because of the scarcity of relevant examples, which reduces our preliminary work in [48]. Based on our analyses, we propose developing low-complexity LMLL methods to facilitate fast prediction and compact model by restraining less performance-influential labels. Two approaches based on less performance-influential labels removing and discriminant parameters preserving are presented for the variants of demands. Extensive experiments verify the efficacy of the proposed methods in terms of promising classification performance, as well as the clear benefits on prediction time and model size reduction. The contribution of this work is that we provide a different aspect for LMLL, revealing that significant attention should be paid to the design of performance metrics, to effectively exploit the great merit of tail labels.

References

[1] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.
[2] D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang, "Multi-label prediction via compressed sensing," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2009, pp. 772–780.
[3] I. Partalas *et al.*, "LSHTC: A benchmark for large-scale text classification," 2015, *arXiv:1503.08581*. [Online]. Available: https://arxiv.org/abs/1503.08581
[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Miami, FL, USA, Jun. 2009, pp. 248–255.
[5] J. McAuley, R. Pandey, and J. Leskovec, "Inferring networks of substitutable and complementary products," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Sydney, NSW, Australia, 2015, pp. 785–794.
[6] M.-L. Zhang and Z.-H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.
[7] G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification," in *Proc. Eur. Conf. Mach. Learn.*, Warsaw, Poland, 2007, pp. 406–417.
[8] M.-L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Washington, DC, USA, 2010, pp. 999–1008.
[9] J. Weston, S. Bengio, and N. Usunier, "WSABIE: Scaling up to large vocabulary image annotation," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, Barcelona, Spain, vol. 11, 2011, pp. 2764–2770.
[10] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma, "Multi-label learning with millions of labels: Recommending advertiser bid phrases for Web pages," in *Proc. 22nd Int. Conf. World Wide Web*, Rio de Janeiro, Brazil, 2013, pp. 13–24.
[11] H.-F. Yu, P. Jain, P. Kar, and I. S. Dhillon, "Large-scale multi-label learning with missing labels," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 593–601.
[12] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2014, pp. 263–272.

[13] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, "Sparse local embeddings for extreme multi-label classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 730–738.

[14] I. E. H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. S. Dhillon, "PD-Sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1–9.

[15] C. Xu, D. Tao, and C. Xu, "Robust extreme multi-label learning," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 1275–1284.

[16] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, 2016, pp. 935–944.

[17] C.-K. Yeh, W.-C. Wu, W.-J. Ko, and Y.-C. F. Wang, "Learning deep latent space for multi-label classification," in *Proc. 31st AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2017, pp. 2838–2844.

[18] R. Babbar and B. Schölkopf, "Dismec: Distributed sparse machines for extreme multi-label classification," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Cambridge, U.K., 2017, pp. 721–729.

[19] I. E. H. Yen, X. Huang, W. Dai, P. Ravikumar, I. Dhillon, and E. Xing, "PPDsparse: A parallel primal-dual sparse method for extreme classification," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Halifax, NS, Canada, 2017, pp. 545–553.

[20] R. Babbar and B. Schölkopf, "Adversarial extreme multi-label classification," 2018, *arXiv:1803.01570*. [Online]. Available: https://arxiv.org/abs/1803.01570

[21] Y. Tagami, "AnnexML: Approximate nearest neighbor search for extreme multi-label classification," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Halifax, NS, Canada, 2017, pp. 455–464.

[22] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Compact multi-label learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, 2018, pp. 4066–4073.

[23] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4324–4338, Sep. 2017.

[24] X. Shen, W. Liu, Y. Luo, Y.-S. Ong, and I. W. Tsang, "Deep binary prototype multi-label learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, 2018, pp. 2675–2681.

[25] T. Wei, W.-W. Tu, and Y.-F. Li, "Learning for tail label data: A label-specific feature approach," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Macau, China, 2019, pp. 3842–3848.

[26] Q. Mao, I. W.-H. Tsang, and S. Gao, "Objective-guided image annotation," *IEEE Trans. Image Process.*, vol. 22, no. 4, pp. 1585–1597, Apr. 2013.

[27] Y.-X. Wang and M. Hebert, "Learning to learn: Model regression networks for easy small sample learning," in *Proc. Eur. Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 616–634.

[28] Y.-X. Wang, D. Ramanan, and M. Hebert, "Learning to model the tail," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 7032–7042.

[29] J. Li, K. Lu, Z. Huang, and H.-T. Shen, "Two birds one stone: On both cold-start and long-tail recommendation," in *Proc. ACM Multimedia Conf.*, Silicon Valley, CA, USA, 2017, pp. 898–906.

[30] W. Liu, I. W. Tsang, and K.-R. Müller, "An easy-to-hard learning paradigm for multiple classes and multiple labels," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 3300–3337, 2017.

[31] Y. Zhang and J. Schneider, "Multi-label output codes using canonical correlation analysis," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, 2011, pp. 873–882.

[32] Y.-N. Chen and H.-T. Lin, "Feature-aware label space dimension reduction for multi-label classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1529–1537.

[33] A. Kapoor, R. Viswanathan, and P. Jain, "Multilabel classification using Bayesian compressed sensing," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 2645–2653.

[34] M. M. Cisse, N. Usunier, T. Artières, and P. Gallinari, "Robust bloom filters for large multilabel classification tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2013, pp. 1851–1859.

[35] W. Bi and J. T. Kwok, "Efficient multi-label classification with many labels," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 405–413.

[36] Z. Lin, G. Ding, M. Hu, and J.-M. Wang, "Multi-label classification via feature-aware implicit label space encoding," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 325–333.

[37] A. E. Choromanska and J. Langford, "Logarithmic time online multiclass prediction," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 55–63.

[38] K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hullermeier, "Extreme F-measure maximization using sparse probability estimates," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 1435–1444.

[39] H. Daumé, III, N. Karampatziakis, J. Langford, and P. Mineiro, "Logarithmic time one-against-some," 2016, *arXiv:1606.04988*. [Online]. Available: https://arxiv.org/abs/1606.04988

[40] W. Siblini, F. Meyer, and P. Kuntz, "CRAFTML, an efficient clustering-based random forest for extreme multi-label learning," in *Proc. 35th Int. Conf. Mach. Learn.*, Stockholm, Sweden, 2018, pp. 4671–4680.

[41] W. Zhang, J. Yan, X. Wang, and H. Zha, "Deep extreme multi-label learning," in *Proc. 2018 ACM Int. Conf. Multimedia Retr.*, Yokohama, Japan, 2018, pp. 100–107.

[42] C. Boutsidis, M. W. Mahoney, and P. Drineas, "An improved approximation algorithm for the column subset selection problem," in *Proc. 20th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2009, pp. 968–977.

[43] A. Niculescu-Mizil and M. Abbasnejad, "Label filters for large scale multilabel classification," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, 2017, pp. 1448–1457.

[44] W. Liu, D. Xu, I. Tsang, and W. Zhang, "Metric learning for multi-output tasks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 408–422, Feb. 2019.

[45] W. Liu and I. W. Tsang, "Making decision trees feasible in ultrahigh feature and label dimensions," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 2814–2849, 2017.

[46] D. Lim, J. McAuley, and G. Lanckriet, "Top-N recommendation with missing implicit feedback," in *Proc. 9th ACM Conf. Recommender Syst.*, Vienna, Austria, 2015, pp. 309–312.

[47] R. E. Schapire, "The strength of weak learnability," *Mach. Learn.*, vol. 5, no. 2, pp. 197–227, 1990.

[48] T. Wei and Y.-F. Li, "Does tail label help for large-scale multi-label learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Stockholm, Sweden, 2018, pp. 2847–2853.

[49] M.-L. Zhang, Y.-K. Li, X.-Y. Liu, and X. Geng, "Binary relevance for multi-label learning: An overview," *Frontiers Comput. Sci.*, vol. 12, no. 2, pp. 191–202, 2018.

[50] M. Ge, C. Delgado-Battenfeld, and D. Jannach, "Beyond accuracy: Evaluating recommender systems by coverage and serendipity," in *Proc. 4th ACM Conf. Recommender Syst.*, Barcelona, Spain, 2010, pp. 257–260.

[51] P. Castells, N. J. Hurley, and S. Vargas, "Novelty and diversity in recommender systems," in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2015, pp. 881–918.

Tong Wei is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Nanjing University, Nanjing, China.

His current research interest includes machine learning.

Mr. Wei is currently a member of the LAMDA Group.

Yu-Feng Li (M'16) received the B.Sc. and Ph.D. degrees in computer science from Nanjing University, Nanjing, China, in 2006 and 2013, respectively.

In 2013, he joined the Department of Computer Science and Technology, Nanjing University, as an Assistant Researcher. He is currently an Associate Professor with the National Key Laboratory for Novel Software Technology, Department of Computer Science, Nanjing University. He has authored or coauthored more than 40 papers in top-tier journal and conferences such as *Journal of Machine Learning Research* (JMLR), IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), *Artificial Intelligence Journal* (AIJ), ICML, NIPS, AAAI, and so on. His current research interests include machine learning, especially semisupervised learning, weakly supervised learning, statistical learning, and optimization.

Dr. Li is currently a member of the LAMDA Group. He is/was a Senior Program Committee Member of top-tier AI conferences such as IJCAI'15, IJCAI'17, AAAI'19, and an Editorial Board Member of machine learning journal special issues. He was a recipient of the Outstanding Doctoral Dissertation Award from the China Computer Federation (CCF), the Outstanding Doctoral Dissertation Award from Jiangsu Province, and the Microsoft Fellowship Award.