Learning for Tail Label Data: A Label-Specific Feature Approach

Tong Wei[1] , **Wei-Wei Tu**[2] and **Yu-Feng Li**[1]

[1]National Key Laboratory for Novel Software Technology Nanjing University, Nanjing 210023, China
[2]4Paradigm Inc., Beijing, China

{weit,liyf}@lamda.nju.edu.cn , tuww.cn@gmail.com

Abstract

Tail label data (TLD) is prevalent in real-world tasks, and large-scale multi-label learning (LMLL) is its major learning scheme. Previous LMLL studies typically need to additionally take into account extensive head label data (HLD), and thus fail to guide the learning behavior of TLD. In many applications such as recommender systems, however, the prediction of tail label is very necessary, since it provides very important supplementary information. We call this kind of problem as *tail label learning*. In this paper, we propose a novel method for the tail label learning problem. Based on the observation that the raw feature representation in LMLL data usually benefits HLD, which may not be suitable for TLD, we construct effective and rich label-specific features through exploring labeled data distribution and leveraging label correlations. Specifically, we employ clustering analysis to explore discriminative features for each tail label replacing the original high-dimensional and sparse features. In addition, due to the scarcity of positive examples of TLD, we encode knowledge from HLD by exploiting label correlations to enhance the label-specific features. Experimental results verify the superiority of the proposed method in terms of performance on TLD.

1 Introduction

Tail label data (TLD) is prevalent in real-world applications. It follows a power-law distribution (as illustrated in Figure 1), and provides irreplaceable information in comparison with head label data (HLD). For instance, in web page categorization [Ioannis *et al.*, 2015], there are thousands of labels from Wikipedia and more than 70% of them occur in at most 15 web pages. Little information is gained by predicting popular labels such as "Poems" for the Divine Comedy article as compared to predicting relatively infrequent labels such as "Epic poems in Italian" (which implies "Poems" and more); in recommender systems [McAuley *et al.*, 2015], popular items are well-known by users and recommending long-tailed items can delight users and boost the sales. Similar applications can be found in image annotation [Deng *et al.*, 2009], video classification [Sami *et al.*, 2016] and so on.

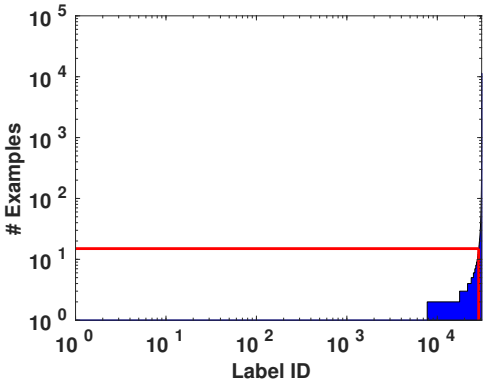

Figure 1: The number of examples for each label is presented on Wiki10 data set. The horizontal axis indicates the indices of labels, while the vertical axis indicates the number of associated examples in the training data. The vertical red line indicates that labels to the left of it (more than 70%) occur in at most 15 examples.

The major learning scheme to model TLD is large-scale multi-label learning (LMLL) [Hsiang-Fu *et al.*, 2014]. It attempts to annotate unseen data with the most relevant subset of labels out of a huge collection including both head labels and tail labels. In the past few years, LMLL has attracted considerable attention and a large number of LMLL algorithms have been proposed, such as FastXML [Prabhu and Varma, 2014], LEML [Hsiang-Fu *et al.*, 2014] and so on.

Previous LMLL studies need to additionally take into account extensive HLD, and fail to directly guide the learning of TLD. Specifically, existing approaches train models leveraging the entire label set and evaluate their learning performance considering both head labels and tail labels. Due to the large population of HLD, the learning performance is primarily dominated by HLD rather than TLD [Wei and Li, 2018; 2019]. In many applications, however, the prediction of tail label is very necessary, since it provides very important supplementary information. The question of learning prediction for TLD has not been thoroughly studied, though it is widely stated that tail labels are rewarding if predicted correctly [Himanshu *et al.*, 2016; Xu *et al.*, 2016].

In this work, focusing on the learning performance of TLD, we evaluate LMLL approaches explicitly on TLD rather than HLD. That is, during the inference phase, only tail labels with

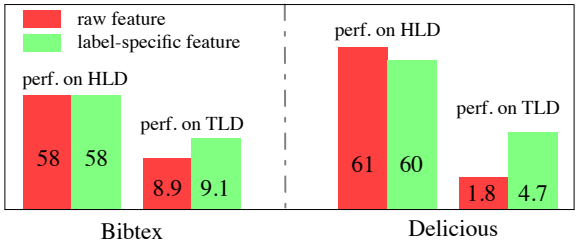

Figure 2: Performance (P@1) of Binary Relevance approach on Bibtex and Delicious data sets with the raw feature representation and the constructed label-specific features respectively. As can be seen, raw features usually benefit HLD rather than TLD. With label-specific features, the performance on TLD is clearly increased.

the top-ranked predictive score for each unseen instance are predicted as relevant. We call this kind of learning problem as *tail label learning*. The main difficulty lies in the scarcity of positive examples of TLD. That is, only a handful of examples are positive for each tail label and the rest are negative examples. Moreover, the raw feature representation in LMLL data is usually high-dimensional and sparse [Prabhu and Varma, 2014], which usually benefits HLD and may not be suitable for TLD (as illustrated in Figure 2).

To alleviate these issues, in this paper, we present an efficient algorithm named TAIL, i.e., learning for TAIL label data. Inspired by [Zhang and Wu, 2015], we consider constructing discriminative features specific to every tail label, i.e., *label-specific features*, with a low-dimensional feature space. Our basic idea is that different class labels usually carry specific characteristics of their own, and it could be beneficial to exploit different feature sets for the discrimination of different labels. Based on this recognition, TAIL induces classification learners TLD based on generated label-specific features rather than the original input features. Specifically, we construct label-specific features from two perspectives. First, to explore discriminative features, we construct *label-specific features w.r.t instances* through clustering analysis on its positive and negative instances for each tail label. Second, on the aspect of label correlations, we encode knowledge from HLD by exploiting the relationship between head labels and tail labels, i.e., *label-specific features w.r.t labels*, to enhance label-specific features. Concretely, we build a k-NN graph which characterizes the affinity among labels and aggregate the predictive information from head label classifiers into generated features for each tail label. Furthermore, to mitigate the class-imbalance problem, we leverage the negative sampling strategy to balance the population of positive and negative instances. Extensive experimental comparisons and studies verify the effectiveness of the proposed method.

The rest of this paper is organized as follows. We start by a brief review of related work. Then we present the proposed approach. After that, experimental results are reported followed by the conclusion of this work.

2 Related Work

This work is mostly related to tail label problem and feature construction in LMLL.

Tail Label in LMLL

Recently, there are some discussions on the power law distribution in LMLL.Himanshu *et al.* [2016] explained that infrequently occurring tail labels are harder to predict than frequently occurring ones since they have little training examples. Xu *et al.* [2016] 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 *et al.* [2017] cast the tail label problem as transfer learning by transferring knowledge from the data-rich head to the data-poor tail class labels. Babbar and Schölkopf [2018] viewed the tail label problem as a setup in which an adversary is generating test examples such that the features of the test set instances are quite different from those in the training set. The tail label problem is also related to weakly supervised learning [Li and Liang, 2019; Li *et al.*, 2016]. Most of these studies, learn from LMLL data by manipulating with the identical feature set, which may be not suitable for TLD, i.e., the original high-dimensional and sparse features are employed in training and inference processes of the entire label set.

LMLL Feature Construction

There are some studies about multi-label feature selection. For example, Zhang *et al.* [2009] adapted the classical naive Bayes classifiers. Ma *et al.* [2012] proposed to learn a feature subspace that is shared among multiple different classes. Jian *et al.* [2016] introduced a principled way of exploiting label correlations for feature selection in the presence of noisy and incomplete label information.

Existing approaches need to additionally take into account HLD and are unable to guide the learning of TLD. In addition, the performance of LMLL approaches on tail labels has not been investigated. To the best of our knowledge, this is the first time learning label-specific features for tail labels is studied.

3 The Proposed Approach

In the following, we first introduce the problem setup and then present the label-specific feature construction method.

3.1 Preliminary

Let $\mathcal{X}$ denote the input space and $\mathcal{Y}$ the output space, and the number of labels $K := |\mathcal{Y}|$, where $|\cdot|$ represents the set cardinality. Labeled samples are pairs $(\boldsymbol{x}, \mathcal{P})$ with $\boldsymbol{x} \in \mathcal{X}$ and $\mathcal{P} \in \mathcal{Y}$ which denotes the set of correct labels for the instance $\boldsymbol{x}$. We use the notation $\mathcal{N} := \mathcal{Y} \setminus \mathcal{P}$ to denote the set of negative labels for the example. Given a collection of N training samples $\{\boldsymbol{x}_i, \mathcal{P}_i\}_{i=1}^{N}$, LMLL aims to learn a scoring function $f : \mathcal{X} \to \mathbb{R}^K$ for a large output space $\mathcal{Y}$.

Considering that, head label prediction could be done very well using off-the-shelf LMLL approaches [Hsiang-Fu *et al.*, 2014; Bhatia *et al.*, 2015; Babbar and Schölkopf, 2017], for the inference concerning tail labels, we employ a specially designed model to achieve better performance than LMLL approaches. Such setting is feasible because, in LMLL systems, we could always separately predict a few head labels and tail labels as relevant. By splitting the label space into two parts, $\mathcal{Y}_c \subset \mathcal{Y}$ and $\mathcal{Y}_t = \mathcal{Y} \setminus \mathcal{Y}_c$ represent head labels and tail labels,

respectively. Let $K_c := |\mathcal{Y}_c|$ and $K_t := |\mathcal{Y}_t|$. Formally, we define the head label and tail label in Definition 1.

Definition 1 (Head Label & Tail Label). *Let $\mathcal{D} = \{\boldsymbol{x}_i, \mathcal{P}_i\}_{i=1}^N$ be a large-scale multi-label data set where labels follow a power-law distribution. Suppose labels $\{l_1, \cdots, l_K\}$ are organized by frequencies in descending order where $\sum_{j=1}^N \mathbb{I}(l_i \in \mathcal{P}_j) >= \sum_{j=1}^N \mathbb{I}(l_{i+1} \in \mathcal{P}_j)$, $\forall 1 \leq i \leq K-1$. Frequently occurring labels $\{l_1, \cdots, l_{K_c}\}$ are referred to as head labels and infrequently occurring ones $\{l_{K_c+1}, \cdots, l_K\}$ are referred to as tail labels.*

3.2 Label-specific Feature Construction

Recently, many effective strategies are proposed to learn more discriminative features [Zhang and Wu, 2015; Jia and Zhang, 2019]. However, these studies typically focus on traditional multi-label learning problems, which do not finalize a systematical solution for tail label learning. Inspired by previous studies, we propose to improve the learning performance on TLD through constructing *label-specific features*.

Specifically, given a data set $\mathcal{D} = \{\boldsymbol{x}_i, \mathcal{P}_i\}_{i=1}^N$, TAIL constructs *label-specific features* for each tail label from $\mathcal{D}$ following two elemental steps, i.e., *label-specific feature construction w.r.t instances* and *label-specific feature construction w.r.t labels*. Then it induces classification models based on generated features instead of the original input features. In the following, we present details of the label-specific feature construction strategies.

Label-specific Feature Construction w.r.t Instances

In the first step, TAIL aims to generate distinguishing features which capture the specific characteristics of each tail label to facilitate its discrimination process. To this end, TAIL investigates data distribution properties by employing clustering analysis method which has been widely used [Zhang and Wu, 2015]. In particular, with respect to tail label $l_i, \forall K_c < i \leq K$, the set of positive training instances $\tilde{\mathcal{P}}_i$ as well as the set of negative training instances $\tilde{\mathcal{N}}_i$ are denoted as follows:

$$\begin{aligned} \tilde{\mathcal{P}}_i &= \{\boldsymbol{x}_j | (\boldsymbol{x}_j, \mathcal{P}_j) \in \mathcal{D}, l_i \in \mathcal{P}_j\} \\ \tilde{\mathcal{N}}_i &= \{\boldsymbol{x}_j | (\boldsymbol{x}_j, \mathcal{P}_j) \in \mathcal{D}, l_i \notin \mathcal{P}_j\} \end{aligned} \tag{1}$$

In other words, $\tilde{\mathcal{P}}_i$ and $\tilde{\mathcal{N}}_i$ consist of the training instances in $\mathcal{D}$ with and without label l_i, respectively. Similar to [Zhang and Wu, 2015], we adopt the popular k-means algorithm to partition $\tilde{\mathcal{P}}_i$ into m_i^+ disjoint clusters whose centers are denoted as $\left\{\boldsymbol{p}_1^i, \boldsymbol{p}_2^i, \cdots, \boldsymbol{p}_{m_i^+}^i\right\}$. Similarly, $\tilde{\mathcal{N}}_i$ is also partitioned into m_i^- disjoint clusters whose centers are denoted as $\left\{\boldsymbol{n}_1^i, \boldsymbol{n}_2^i, \cdots, \boldsymbol{n}_{m_i^-}^i\right\}$. Following the setting in [Zhang and Wu, 2015], we choose to set equivalent number of clusters for $\tilde{\mathcal{P}}_i$ and $\tilde{\mathcal{N}}_i$, i.e. $m_i^+ = m_i^- = m_i$. In this way, clustering information gained from positive instances as well as negative instances are treated with equal importance. Specifically, the number of clusters retained for both positive and negative instances is set to be:

$$m_i = \gamma \cdot \min(|\tilde{\mathcal{P}}_i|, |\tilde{\mathcal{N}}_i|) \tag{2}$$

Here, $\gamma \in [0, 1]$ is the ratio parameter controlling the number of clusters. Intuitively, the retained cluster centers characterize the underlying structure of input space and can be used as the bases for label-specific feature construction.

In detail, TAIL builds a mapping ϕ_i from the original d-dimensional feature space $\mathcal{X}$ to the $2m_i$-dimensional label-specific feature space as follows:

$$\begin{aligned} &\phi_i(\boldsymbol{x}) = \\ &\left[d(\boldsymbol{x}, \boldsymbol{p}_1^i), \cdots, d(\boldsymbol{x}, \boldsymbol{p}_{m_i}^i), d(\boldsymbol{x}, \boldsymbol{n}_1^i), \cdots, d(\boldsymbol{x}, \boldsymbol{n}_{m_i}^i)\right] \end{aligned} \tag{3}$$

Here, $d(\cdot, \cdot)$ represents the distance metric and is set to the Euclidean metric following [Zhang and Wu, 2015].

Label-specific Feature Construction w.r.t Labels

In the second step, TAIL aims to enhance label-specific features by exploiting label correlations between head labels and tail labels. We leverage label cooccurrence statistics obtained from training data to build a connection between head labels and tail labels. Specifically, similarity is computed for each pair of tail label and head label (l_i, l_j) by $sim(l_i, l_j) = |\tilde{\mathcal{P}}_i \cap \tilde{\mathcal{P}}_j|$, $\forall 1 \leq j \leq K_c < i \leq K$. After that, we construct a k-NN graph that is known for its good performance [Ebert *et al.*, 2010; Maier *et al.*, 2009] using $dist(l_i, l_j) = N - sim(l_i, l_j)$ as the distance metric, i.e.,

$$W_{i,j} = \begin{cases} 1 & \text{if } l_j \text{ is in the } k\text{-NN of } l_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

To leverage the correlations between head labels and tail labels, for instance $\boldsymbol{x}$, we apply K_c classifiers $\{f_1, \ldots, f_{K_c}\}$ for head labels on $\boldsymbol{x}$ and take the predictive information as transferred knowledge. More precisely, for tail label l_i and head label l_j, the predictive information of $\bar{f}_j$ is filtered by $W_{i,j} \odot f_j(\boldsymbol{x})$. If $W_{i,j} = 1$, then $f_j(\boldsymbol{x})$ is selected as one of generated features, otherwise discarded. By doing this, the relationship between labels is encoded as augmented feature representations, which is proved beneficial for building classification models in our experiments.

In detail, TAIL builds a label correlation aware mapping ψ_i for tail label l_i from the original D-dimensional feature space $\mathcal{X}$ to the k-dimensional label specific feature space as follows:

$$\psi_i(\boldsymbol{x}) = [W_{i,1} \odot f_1(\boldsymbol{x}), \cdots W_{i,K_c} \odot f_{K_c}(\boldsymbol{x})] \tag{5}$$

Finally, TAIL induces a family of $K_t = K - K_c$ classification models $\{f_{K_c+1}, \ldots, f_K\}$ by aggregating feature mappings $\phi(\cdot)$ and $\psi(\cdot)$. Specifically, for tail label l_i, a binary training set $\mathcal{D}_i^*$ with m examples is created from $\mathcal{D}$ by applying our two label-specific feature generation steps. Algorithm 1 lists the details of TAIL approach.

3.3 Computational Complexity Analysis

For each tail label, it first takes $\mathcal{O}(2m_i nDT)$ and $\mathcal{O}(N\bar{L}^2 + K_t K_c \log k)$ to construct two types of label-specific features, respectively. Here, T is the number of iterations when performing k-means, k is set to 5 in k-NN search, $\bar{L}$ is the averaged number of relevant labels per instance, and $n \ll N$ is the total number of examples after negative downsampling. Subsequently, TAIL builds a linear classifier in $\mathcal{O}((2m_i + k)n)$. Therefore the total computational cost to train TAIL for

Algorithm 1 The pseudo-code of TAIL

Input:
$\mathcal{D}$: LMLL training set $\mathcal{D} = \{\boldsymbol{x}_i, \mathcal{P}_i\}_{i=1}^N$
K_c: the number of head labels
k: the number of nearest neighbors considered
γ: the ratio parameter controlling the number of clusters
$\mathcal{L}$: the binary classification learner
Output:
$\{f_i\}_{i=K_c+1}^K$: a family of tail label classifiers
Process:

```
for i = K_c + 1, ..., K do
    Construct P̃_i and Ñ_i based on D according to Eq. (1)
    Perform k-means clustering on P̃_i and Ñ_i, each with m_i clusters as defined by Eq. (2)
    Construct the mapping φ_i for l_i according to Eq. (3)
    Compute k-NN adjacent matrix according to Eq. (4)
    Construct the mapping ψ_i for l_i according to Eq. (5)
    Aggregate label-specific features by concatenating φ_i and ψ_i for l_i
    Induce f_i by invoking a binary learner L on the constructed label-specific features for l_i
end for
return {f_i}_{i=K_c+1}^K
```

each label is $\mathcal{O}(m_i nDT + N\bar{L}^2 + K_t K_c \log k)$ thanks to $(2m_i + k)n \ll m_i nDT$. Note that, for the i-th tail label, $1 \leq i \leq K_t$, the dimensionality of constructed feature representation in $\mathcal{D}_i^*$ is exactly $2m_i + k \ll D$. In the testing stage, the computational cost for TAIL to predict a new instance is $\mathcal{O}((2m_i + k)D)$ per label. both the training and testing phase of multiple tail labels can be easily parallelized.

The analysis shows that the total computational complexities scale linearly with size of the data set. Thus, both methods are very suitable for the LMLL applications.

4 Experiments

We conduct comprehensive experiments on LMLL benchmark data sets to evaluate the efficacy of our proposal.

4.1 Experimental Setup

Experiments are conducted on four benchmark data sets with the number of labels ranging from 159 to 30K. Table 1 lists the detailed statistics. We report and compare the results using the same train/test splits of data sets. All the data sets as well as the code of compared methods are publicly available[1]. Notably, hundreds of labels on EUR-Lex data set do not have any positive example available in the training set, and thus we discard such labels. In all of our experiments, we fix the number of nearest neighbors considered to 5, i.e., $k = 5$. We set the ratio parameter γ during clustering to 0.1 following the setting in [Zhang and Wu, 2015]. For other comparison methods, we use the default parameter settings in the code.

Computational Device
All experimental comparisons are conducted on the same PC machine with an Intel i5-6500 3.20GHz CPU and 32GB RAM.

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

Data set	Train N	Features D	Labels K	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

Table 1: Data set statistics

Compared Methods

We compare our method to Binary Relevance (BR) and seven state-of-the-art LMLL approaches.

- Binary Relevance [Zhang and Zhou, 2014] builds OvR SVM for each label using Liblinear [Fan *et al.*, 2008].
- LEML [Hsiang-Fu *et al.*, 2014] is an embedding method based on low-rank empirical risk minimization.
- FastXML [Prabhu and Varma, 2014] is a random forest-based LMLL approach.
- SLEEC [Bhatia *et al.*, 2015] learns the embedding of labels by preserving the pairwise distances between a few nearest label neighbors.
- CoH [Shen *et al.*, 2018] proposes a co-hashing method which jointly compresses the input and output into compact binary embeddings.
- DisMEC [Babbar and Schölkopf, 2017] learns a 1vsA linear-SVM in a distributed fashion.
- PD-Sparse [Yen *et al.*, 2016] proposes to solve ℓ_1 regularized multi-class loss using Frank-Wolfe based algorithm.
- REML [Xu *et al.*, 2016] proposes to decompose label matrix into a low-rank matrix and a sparse matrix to model head labels and tail labels respectively.

We build TAIL based on Liblinear using constructed label-specific features. For comparison methods, we first obtain predictive scores over the entire label set and take top k tail labels with the highest predictive score for evaluation.

Performance Metrics

In LMLL applications, e.g., recommender systems, only the top k ranked labels are concerned, where P@k and nDCG@k are widely used [Himanshu *et al.*, 2016]. Accordingly, P@k and nDCG@k are defined as

$$P@k = \frac{1}{k} \sum_{l \in \text{rank}_k(\boldsymbol{z})} \mathbb{I}(l \in \mathcal{P})$$

$$nDCG@k = \frac{DCG@k(\boldsymbol{z}, \mathcal{P})}{\sum_{l=1}^{\min(k, |\mathcal{P}|)} \frac{1}{\log(l+1)}},$$

where $DCG@k(\boldsymbol{z}, \mathcal{P}) := \sum_{l \in \text{rank}_k(\boldsymbol{z})} \frac{\mathbb{I}(l \in \mathcal{P})}{\log(l+1)}$. Here, $\boldsymbol{z}$ is the predicted score vector of instance $\boldsymbol{x}$ and $\mathcal{P}$ is the true label set. The indicator function $\mathbb{I}(\cdot)$ returns 1 if the condition is true, otherwise 0.

Data set		P@1 (%)	P@3 (%)	P@5 (%)	nDCG@1 (%)	nDCG@3 (%)	nDCG@5 (%)
Bibtex	BR	8.95	**4.04**	**2.54**	8.95	10.29	10.52
	TAIL	**9.18**	**4.04**	**2.54**	**9.18**	**10.37**	**10.58**
Delicious	BR	1.85	1.55	1.41	1.85	2.98	3.83
	TAIL	**4.71**	**2.69**	**1.92**	**4.71**	**5.87**	**6.41**
EUR-Lex	BR	**6.98**	**3.68**	**2.49**	2.68	8.01	**8.47**
	TAIL	6.30	3.57	2.28	**3.06**	**8.62**	8.31
Wiki10	BR	5.34	4.10	4.09	4.36	4.12	4.17
	TAIL	**5.52**	**5.10**	**4.46**	**4.60**	**5.12**	**5.57**

Table 2: Performance comparison between the proposed TAIL and BR in terms of P@k and nDCG@k with the number of tail labels $K_t = \frac{K}{10}$ for small data sets (Bibtex, Delicious) and $K_t = \frac{K}{2}$ for large ones (EUR-Lex, Wiki10). The best results in terms of each metric are in bold.

Data set		FastXML	LEML	SLEEC	CoH	DiSMEC	PD-Sparse	REML	TAIL
Bibtex	P@1 (%)	5.90	5.57	**8.97**	6.53	8.43	4.34	5.72	**9.18**
	P@3 (%)	1.97	1.86	**3.93**	2.20	3.90	2.77	2.03	**4.04**
	P@5 (%)	1.18	1.11	1.50	1.59	**2.54**	1.93	1.24	**2.54**
	nDCG@1 (%)	6.17	5.61	2.21	6.58	**8.60**	4.34	2.01	**9.18**
	nDCG@3 (%)	2.28	1.86	6.99	3.62	**10.30**	4.49	2.76	**10.37**
	nDCG@5 (%)	2.11	1.67	8.46	2.06	**10.32**	5.72	4.58	**10.58**
Delicious	P@1 (%)	2.38	2.24	**3.11**	1.24	2.02	1.22	2.77	**4.71**
	P@3 (%)	0.79	0.75	**2.21**	1.82	1.78	1.03	1.26	**2.69**
	P@5 (%)	0.48	0.45	**1.53**	0.11	1.50	0.85	0.93	**1.92**
	nDCG@1 (%)	1.27	**1.92**	1.11	1.43	1.85	1.22	1.55	**4.71**
	nDCG@3 (%)	1.71	0.64	2.50	2.16	**2.96**	1.80	1.82	**5.87**
	nDCG@5 (%)	2.62	0.55	3.41	2.20	**3.87**	2.61	3.00	**6.41**
EUR-Lex	P@1 (%)	6.15	0.21	6.24	3.87	**7.22**	2.28	5.79	**6.30**
	P@3 (%)	2.72	0.07	2.09	1.98	**3.54**	2.00	2.54	**3.57**
	P@5 (%)	1.63	0.04	1.76	1.89	**2.83**	1.89	1.20	**2.28**
	nDCG@1 (%)	2.24	0.23	2.25	**3.78**	2.62	1.27	1.91	**3.06**
	nDCG@3 (%)	5.95	0.07	5.35	1.49	**8.30**	1.96	4.20	**8.62**
	nDCG@5 (%)	6.83	0.06	6.30	1.92	**8.90**	2.44	5.41	**8.31**
Wiki10	P@1 (%)	4.34	4.82	**5.14**	3.03	4.61	3.98	3.48	**5.52**
	P@3 (%)	1.45	1.61	**4.86**	1.56	4.10	1.60	1.27	**5.10**
	P@5 (%)	0.87	0.96	3.40	1.09	**4.80**	1.19	0.43	**4.46**
	nDCG@1 (%)	**5.06**	4.46	**5.14**	3.41	5.00	1.01	4.28	4.60
	nDCG@3 (%)	1.48	1.47	4.87	3.25	**5.27**	1.90	1.59	**5.12**
	nDCG@5 (%)	1.29	1.25	3.46	3.09	**5.36**	1.96	1.30	**5.57**

Table 3: Comparison with state-of-the-art approaches in terms of PSP@k and PSnDCG@k with $K_t = \frac{K}{10}$ for small data sets and $K_t = \frac{K}{2}$ for large data sets. The best and second best results are in bold.

4.2 Comparison with Baseline Approach

We first study the effectiveness of TAIL at improving performance in comparison to Binary Relevance (BR) using raw features. Table 2 depicts the comparison results. On relatively small data set Bibtex, TAIL achieves competitive results across six different metrics. Considering the relatively balanced label distribution due to the small label set, it might be inaccurate to capture label relationship between head labels and tail labels. On the other three larger data sets with high-dimensional label space, TAIL improves the prediction accuracy with a relatively large margin in most cases. This justifies the superiority of constructed label-specific features to raw features.

4.3 Comparison with State-of-the-art Approaches

In this experiment, we compare the performance of TAIL with state-of-the-art methods: FastXML, LEML, SLEEC, DiS-

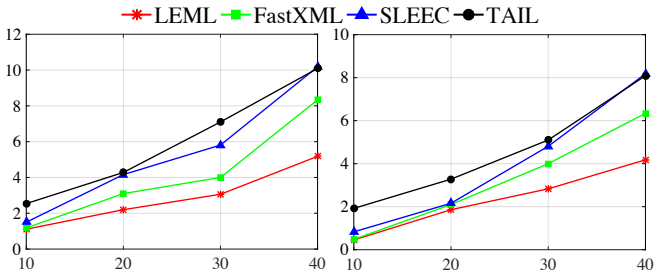

Figure 3: Studies on different values of K_t on data sets Bibtex (left) and Delicious (right). X-axis: value of $K_t(\%)$. Y-axis: P@5 (%).

Data set	P@1 (%)	P@3 (%)	P@5 (%)
Bibtex	7.12	2.20	1.61
Delicious	3.82	2.07	0.88
EUR-Lex	4.50	2.00	0.42

Table 4: Experimental results of employing only label-specific features w.r.t. instances.

MEC, CoH, PD-Sparse, and REML. As demonstrated in Table 3, TAIL achieves better performance compared to state-of-the-art approaches, which demonstrates the merit of label-specific features. Specifically, TAIL achieves the best or second best performance on TLD in 23 out of 24 cases. Sophisticated solvers, such as FastXML, LEML, and SLEEC, does not achieve as good performance on tail labels as on head labels. The reason may owe to the fact of population bias among the training set. Note that, the predictive accuracy on TLD is very limited especially on larger data sets because scarce positive examples are not sufficient to learn satisfactory models. Specifically, there are more than 20% of labels have no more than 1 associated instance on Wiki10 and EUR-Lex data sets. In order to gain better learning performance on tail labels, it is necessary to leverage side information, such as the semantic meaning of each class label or the underlying structure among labels.

4.4 Influences of Two Feature Construction Steps

To study the effectiveness of label-specific features w.r.t. instances and labels separately, we report performance by employing only label-specific features w.r.t. instances in Table 4 and label-specific features w.r.t. instances in Table 5. As depicted in Table 4, in comparison with the results in Table 2, it results in more than 30% performance degradation depicting the importance of label relationship. Conversely, performance degrades when only label-specific features w.r.t. labels are employed, which is in line with the observations of Table 4. The case study justifies that both label-specific feature construction steps are vital to the learning performance of TLD.

4.5 Parameter Sensitivities Analysis

We further investigate the influence of the number of tail labels K_t to the performance of TAIL in comparison with LEML, FastXML, and SLEEC. We vary the percentage of tail labels ranging from $\{10\%, 20\%, 30\%, 40\%\}$ for comparison. Figure 3 demonstrates that the performance is getting better as the value of K_t grows, which is very intuitive because it is easier to model head labels compared with tail ones and richer information can be leveraged and make knowledge transferring feasible. For different values of K_t, TAIL consistently outperforms competing methods. It can be seen that TAIL can capture label relationships as good as leading LMLL approach SLEEC when extra HLD is available.

Data set	P@1 (%)	P@3 (%)	P@5 (%)
Bibtex	6.30	1.32	0.66
Delicious	3.02	1.96	0.38
EUR-Lex	4.83	2.43	1.13

Table 5: Experimental results of employing only label-specific features w.r.t. labels.

5 Conclusion

In this paper, for the first time, we attempt to improve the learning performance on tail label data and we call this kind of learning problem as *tail label learning*. A data-level solution named TAIL is proposed to directly guide the learning of tail label data through extracting label-specific features. It replaces the original high-dimensional and sparse feature representation which may not be suitable for tail label data. Specifically, TAIL constructs label-specific features concerning each tail label through exploring data distribution and leveraging label correlations. Extensive empirical studies on benchmark data sets demonstrate that the learning performance of tail label data is clearly improved and validate the effectiveness of the proposed approach. In the sequel, it is interesting to investigate the sample generation mechanism of tail label learning.

Acknowledgments

This research was supported by the National Key R&D Program of China (2017YFB1002201), the National Natural Science Foundation of China (61772262) and the Fundamental Research Funds for the Central Universities (020214380053).

References

[Babbar and Schölkopf, 2017] Rohit Babbar and Bernhard Schölkopf. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 721–729, Cambridge, UK, 2017.

[Babbar and Schölkopf, 2018] Rohit Babbar and Bernhard Schölkopf. Adversarial extreme multi-label classification. *arXiv preprint arXiv:1803.01570*, 2018.

[Bhatia *et al.*, 2015] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jbabain. Sparse local embeddings for extreme multi-label classification. In *Advances in Neural Information Processing Systems 28*, pages 730–738. Montreal, Canada, 2015.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the 22nd*

IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, Miami, FL, 2009.

[Ebert *et al.*, 2010] Sandra Ebert, Diane Larlus, and Bernt Schiele. Extracting structures in image collections for object recognition. In *Proceedings of European Conference on Computer Vision*, pages 720–733, Heraklion, Greece, 2010.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Himanshu *et al.*, 2016] Jain Himanshu, Prabhu Yashoteja, and Manik Varma. 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, San Francisco, CA, 2016.

[Hsiang-Fu *et al.*, 2014] Yu Hsiang-Fu, Jain Prateek, Kar Purushottam, and Dhillon Inderjit S. Large-scale multi-label learning with missing labels. In *Proceedings of the 31st International Conference on Machine Learning*, pages 593–601, Beijing, China, 2014.

[Ioannis *et al.*, 2015] Partalas Ioannis, Kosmopoulos Aris, Baskiotis Nicolas, Artieres Thierry, Paliouras George, Gaussier Eric, Androutsopoulos Ion, Amini Massih-Reza, and Galinari Patrick. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.

[Jia and Zhang, 2019] Bin-Bin Jia and Min-Ling Zhang. Multi-dimensional classification via knn feature augmentation. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, 2019.

[Jian *et al.*, 2016] Ling Jian, Jundong Li, Kai Shu, and Huan Liu. Multi-label informed feature selection. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1627–1633, New York City, NY, 2016.

[Li and Liang, 2019] Yu-Feng Li and De-Ming Liang. Safe semi-supervised learning: a brief introduction. *Frontiers of Computer Science*, 13(4):669–676, 2019.

[Li *et al.*, 2016] Yu-Feng Li, Shao-Bo Wang, and Zhi-Hua Zhou. Graph quality judgement: A large margin expedition. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1725–1731, New York City, NY, 2016.

[Ma *et al.*, 2012] Zhigang Ma, Feiping Nie, Yi Yang, Jasper R. Uijlings, and Nicu Sebe. Web image annotation via subspace-sparsity collaborated feature selection. *IEEE Transactions on Multimedia*, 14(4):1021–1030, 2012.

[Maier *et al.*, 2009] Markus Maier, Ulrike von Luxburg, and Matthias Hein. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems 22*, pages 1025–1032, Vancouver, Canada, 2009.

[McAuley *et al.*, 2015] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, Sydney, Australia, 2015.

[Prabhu and Varma, 2014] Yashoteja Prabhu and Manik Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 263–272, New York City, NY, 2014.

[Sami *et al.*, 2016] Abu-El-Haija Sami, Kothari Nisarg, Lee Joonseok, Natsev Paul, Toderici George, Varadarajan Balakrishnan, and Vijayanarasimhan Sudheendra. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[Shen *et al.*, 2018] Xiaobo Shen, Weiwei Liu, Ivor W. Tsang, Quan-Sen Sun, and Yew-Soon Ong. Compact multi-label learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4066–4073, New Orleans, LA, 2018.

[Wang *et al.*, 2017] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems 30*, pages 7032–7042, Long Beach, CA, 2017.

[Wei and Li, 2018] Tong Wei and Yu-Feng Li. Does tail label help for large-scale multi-label learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2847–2853, Stockholm, Sweden, 2018.

[Wei and Li, 2019] Tong Wei and Yu-Feng Li. Learning compact model for large-scale multi-label data. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Honolulu, HI, 2019.

[Xu *et al.*, 2016] Chang Xu, Da-Cheng Tao, and Chao Xu. Robust extreme multi-label learning. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284, San Francisco, CA, 2016.

[Yen *et al.*, 2016] Ian E.-H. Yen, Xiangru Huang, Kai Zhong, Pradeep Ravikumar, and Inderjit S. Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 3069–3077, New York City, NY, 2016.

[Zhang and Wu, 2015] Min-Ling Zhang and Lei Wu. Lift: Multi-label learning with label-specific features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):107–120, 2015.

[Zhang and Zhou, 2014] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

[Zhang *et al.*, 2009] Min-Ling Zhang, José M. Peña, and Victor Robles. Feature selection for multi-label naive bayes classification. *Information Sciences*, 179(19):3218–3229, 2009.