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

Tong Wei¹, Wei-Wei Tu² and Yu-Feng Li¹
¹National Key Laboratory for Novel Software Technology Nanjing University, Nanjing 210023, China
²4Paradigm 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.

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
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.

Figure 2: Performance (P@1) of Binary Relevance approach on Bibtext 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.

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 \( (x, P) \) with \( x \in \mathcal{X} \) and \( P \in \mathcal{Y} \) which denotes the set of correct labels for the instance \( x \). We use the notation \( \mathcal{N} := \mathcal{Y} \setminus P \) to denote the set of negative labels for the example. Given a collection of \( N \) training samples \( \{x_i, P_i\}_{i=1}^N \), 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 \( K_c := |Y_c| \) and \( K_t := |Y_t| \). Formally, we define the head label and tail label in Definition 1.

**Definition 1 (Head Label & Tail Label).** Let \( D = \{x_i, P_i\}_{i=1}^N \) 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 \( \sum_{j=1}^N \mathbb{1}(l_i \in P_j) = \sum_{j=1}^N \mathbb{1}(l_{i+1} \in P_j), \forall 1 \leq i < K - 1. \) 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 systematic 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 \( D = \{x_i, P_i\}_{i=1}^N \), TAIL constructs label-specific features for each tail label from \( D \) 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 \( \mathcal{P}_i \) as well as the set of negative training instances \( \mathcal{N}_i \) are denoted as follows:

\[
\mathcal{P}_i = \{x_j | (x_j, P_j) \in D, l_i \in P_j\}
\]

\[
\mathcal{N}_i = \{x_j | (x_j, P_j) \in D, l_i \notin P_j\}
\]

In other words, \( \mathcal{P}_i \) and \( \mathcal{N}_i \) consist of the training instances in \( D \) with and without label \( l_i \), respectively. Similar to [Zhang and Wu, 2015], we adopt the popular k-means algorithm to partition \( \mathcal{P}_i \) into \( m_i^+ \) disjoint clusters whose centers are denoted as \( \{p_1^i, p_2^i, \cdots, p_{m_i^+}^i\} \). Similarly, \( \mathcal{N}_i \) is also partitioned into \( m_i^- \) disjoint clusters whose centers are denoted as \( \{n_1^i, n_2^i, \cdots, n_{m_i^-}^i\} \). Following the setting in [Zhang and Wu, 2015], we choose to set equivalent number of clusters for \( \mathcal{P}_i \) and \( \mathcal{N}_i \), 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:

\[
m_i = \gamma \cdot \min(|\mathcal{P}_i|, |\mathcal{N}_i|)
\]

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 \( \phi_i \) from the original \( d \)-dimensional feature space \( \mathcal{X} \) to the \( 2m_i \)-dimensional label-specific feature space as follows:

\[
\phi_i(x) = \left[ d(x, p_1^i), \cdots, d(x, p_{m_i^+}^i), d(x, n_1^i), \cdots, d(x, n_{m_i^-}^i) \right] \tag{3}
\]

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 \( \text{sim}(l_i, l_j) = |P_i \cap P_j|, \forall 1 \leq j \leq K_c < i \leq K \). After that, we construct a \( k \)-NN graph that is known for its good performance [Ebert et al., 2010; Maier et al., 2009] using \( \text{dist}(l_i, l_j) = N - \text{sim}(l_i, l_j) \) 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 \( x \), we apply \( K_c \) classifiers \( \{f_1, \cdots, f_{K_c}\} \) for head labels on \( x \) and take the predictive information as transferred knowledge. More precisely, for tail label \( l_i \) and head label \( l_j \), the predictive information of \( f_j \) is filtered by \( W_{i,j} \odot f_j(x) \). If \( W_{i,j} = 1 \), then \( f_j(x) \) 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 \( \psi_i \) for tail label \( l_i \) from the original \( D \)-dimensional feature space \( \mathcal{X} \) to the \( k \)-dimensional label specific feature space as follows:

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

Finally, TAIL induces a family of \( K_i = K - K_c \) classification models \( \{f_{K_c+1}, \cdots, f_K\} \) by aggregating feature mappings \( \phi(\cdot) \) and \( \psi(\cdot) \). Specifically, for tail label \( l_i \), a binary training set \( D_i^T \) with \( m \) examples is extracted from \( D \) 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 \( O(2m_n DT) \) and \( O(NL^2 + K_cK_c \log k) \) 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, \( L \) 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 \( O(2m_n + k)n \). Therefore the total computational cost to train TAIL for
Algorithm 1 The pseudo-code of TAIL.

Input:
\( D: \) LMLL training set \( D = \{x_i, P_i\}_{i=1}^{N} \)
\( K_c: \) the number of head labels
\( k: \) the number of nearest neighbors considered
\( \gamma: \) the ratio parameter controlling the number of clusters
\( \mathcal{L}: \) the binary classification learner

Output:
\( \{f_i\}_{i=K_c+1}^{K_c+k}: \) a family of tail label classifiers

Process:
1. for \( i = K_c + 1, \ldots, K \) do
2. Construct \( \tilde{P}_i \) and \( \tilde{N}_i \) based on \( D \) according to Eq. (1)
3. Perform \( k \)-means clustering on \( \tilde{P}_i \) and \( \tilde{N}_i \), each with \( m_i \) clusters as defined by Eq. (2)
4. Construct the mapping \( \phi_i \) for \( l_i \) according to Eq. (3)
5. Compute \( k \)-NN adjacent matrix according to Eq. (4)
6. Construct the mapping \( \psi_i \) for \( l_i \) according to Eq. (5)
7. Aggregate label-specific features by concatenating \( \phi_i \) and \( \psi_i \) for \( l_i \)
8. Induce \( f_i \) by invoking a binary learner \( \mathcal{L} \) on the constructed label-specific features for \( l_i \)
9. end for
10. return \( \{f_i\}_{i=K_c+1}^{K_c+k} \)

Each label is \( O(m_i n DT + N\bar{L}^2 + K_c K_c \log k) \) thanks to \( (2m_i + k)n \ll m_i n DT \). Note that, for the \( i \)-th tail label, \( 1 \leq i \leq K_c \), the dimensionality of constructed feature representation in \( D_i^* \) is exactly \( 2m_i + k \ll D \). In the testing stage, the computational cost for TAIL to predict a new instance is \( O((2m_i + k)D) \) 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 \( \gamma \) 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

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
Data set & Train \( N \) & Features \( D \) & Labels \( K \) & Test \( M \) & Avg. labels per point & Avg. points per label \\
\hline
BiBtx & 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 \\
\hline
\end{tabular}
\caption{Data set statistics}
\end{table}

\section*{Compared Methods}

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

\begin{itemize}
\item Binary Relevance [Zhang and Zhou, 2014] builds OvR SVM for each label using Liblinear [Fan et al., 2008].
\item LMLL [Hsiang-Fu et al., 2014] is an embedding method based on low-rank empirical risk minimization.
\item FastXML [Prabhu and Varma, 2014] is a random forest-based LMLL approach.
\item SLEEC [Bhatia et al., 2015] learns the embedding of labels by preserving the pairwise distances between a few nearest label neighbors.
\item CoH [Shen et al., 2018] proposes a co-hashing method which jointly compresses the input and output into compact binary embeddings.
\item DisMEC [Babbar and Schölkopf, 2017] learns a 1-vs-A linear-SVM in a distributed fashion.
\item PD-Sparse [Yen et al., 2016] proposes to solve \( \ell_1 \) regularized multi-class loss using Frank-Wolfe based algorithm.
\item 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.
\end{itemize}

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.

\section*{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

\[ P@k = \frac{1}{k} \sum_{l \in \text{rank}_k(z)} I(l \in P) \]

\[ nDCG@k = \frac{DCG@k(z, P)}{\sum_{l = \min(k, |P|)} \frac{1}{\log(l + 1)}} \]

where \( DCG@k(z, P) := \sum_{l \in \text{rank}_k(z)} \frac{1}{\log(l + 1)} I(l \in P) \). Here, \( z \) is the predicted score vector of instance \( x \) and \( P \) is the true label set. The indicator function \( I(\cdot) \) returns 1 if the condition is true, otherwise 0.
<table>
<thead>
<tr>
<th>Data set</th>
<th>P@1 (%)</th>
<th>P@3 (%)</th>
<th>P@5 (%)</th>
<th>nDCG@1 (%)</th>
<th>nDCG@3 (%)</th>
<th>nDCG@5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibtex</td>
<td>8.95</td>
<td>4.04</td>
<td>2.54</td>
<td>8.95</td>
<td>10.29</td>
<td>10.52</td>
</tr>
<tr>
<td>TAIL</td>
<td>9.18</td>
<td>4.04</td>
<td>2.54</td>
<td>9.18</td>
<td>10.37</td>
<td>10.58</td>
</tr>
<tr>
<td>Delicious</td>
<td>1.85</td>
<td>1.55</td>
<td>1.41</td>
<td>1.85</td>
<td>2.98</td>
<td>3.83</td>
</tr>
<tr>
<td>TAIL</td>
<td>4.71</td>
<td>2.69</td>
<td>1.92</td>
<td>4.71</td>
<td>5.87</td>
<td>6.41</td>
</tr>
<tr>
<td>EUR-Lex</td>
<td>6.98</td>
<td>3.68</td>
<td>2.49</td>
<td>2.68</td>
<td>8.01</td>
<td>8.47</td>
</tr>
<tr>
<td>TAIL</td>
<td>6.30</td>
<td>3.57</td>
<td>2.28</td>
<td>3.06</td>
<td>8.62</td>
<td>8.31</td>
</tr>
<tr>
<td>Wiki10</td>
<td>5.34</td>
<td>4.10</td>
<td>4.09</td>
<td>4.36</td>
<td>4.12</td>
<td>4.17</td>
</tr>
<tr>
<td>TAIL</td>
<td>5.52</td>
<td>5.10</td>
<td>4.46</td>
<td>4.60</td>
<td>5.12</td>
<td>5.57</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Data set</th>
<th>FastXML</th>
<th>LEML</th>
<th>SLEEC</th>
<th>CoH</th>
<th>DiSMEC</th>
<th>PD-Sparse</th>
<th>REML</th>
<th>TAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bibtex</td>
<td>5.90</td>
<td>5.57</td>
<td>8.97</td>
<td>6.53</td>
<td>8.43</td>
<td>4.34</td>
<td>5.72</td>
<td>9.18</td>
</tr>
<tr>
<td>P@1 (%)</td>
<td>2.38</td>
<td>2.24</td>
<td>3.11</td>
<td>1.24</td>
<td>2.02</td>
<td>1.22</td>
<td>2.77</td>
<td>4.71</td>
</tr>
<tr>
<td>P@3 (%)</td>
<td>0.79</td>
<td>0.75</td>
<td>2.21</td>
<td>1.82</td>
<td>1.78</td>
<td>1.03</td>
<td>1.26</td>
<td>2.69</td>
</tr>
<tr>
<td>P@5 (%)</td>
<td>0.48</td>
<td>0.45</td>
<td>1.53</td>
<td>0.11</td>
<td>1.50</td>
<td>0.85</td>
<td>0.93</td>
<td>1.92</td>
</tr>
<tr>
<td>Delicious</td>
<td>nDCG@1 (%)</td>
<td>6.17</td>
<td>5.61</td>
<td>2.21</td>
<td>6.65</td>
<td>8.60</td>
<td>4.34</td>
<td>9.18</td>
</tr>
<tr>
<td>P@1 (%)</td>
<td>1.27</td>
<td>1.92</td>
<td>1.11</td>
<td>1.43</td>
<td>1.85</td>
<td>1.22</td>
<td>1.55</td>
<td>4.71</td>
</tr>
<tr>
<td>P@3 (%)</td>
<td>1.71</td>
<td>0.64</td>
<td>2.50</td>
<td>2.16</td>
<td>2.96</td>
<td>1.80</td>
<td>1.82</td>
<td>5.87</td>
</tr>
<tr>
<td>P@5 (%)</td>
<td>2.62</td>
<td>0.55</td>
<td>3.41</td>
<td>2.20</td>
<td>3.87</td>
<td>2.61</td>
<td>3.00</td>
<td>6.41</td>
</tr>
<tr>
<td>EUR-Lex</td>
<td>nDCG@1 (%)</td>
<td>2.24</td>
<td>0.23</td>
<td>2.25</td>
<td>3.78</td>
<td>2.62</td>
<td>1.27</td>
<td>3.06</td>
</tr>
<tr>
<td>P@1 (%)</td>
<td>5.95</td>
<td>0.07</td>
<td>5.35</td>
<td>1.49</td>
<td>8.30</td>
<td>1.96</td>
<td>4.20</td>
<td>8.62</td>
</tr>
<tr>
<td>P@3 (%)</td>
<td>6.83</td>
<td>0.06</td>
<td>6.30</td>
<td>1.92</td>
<td>8.90</td>
<td>2.44</td>
<td>5.41</td>
<td>8.31</td>
</tr>
<tr>
<td>P@5 (%)</td>
<td>4.34</td>
<td>4.82</td>
<td>5.14</td>
<td>3.03</td>
<td>4.61</td>
<td>3.98</td>
<td>3.48</td>
<td>5.52</td>
</tr>
<tr>
<td>Wiki10</td>
<td>nDCG@1 (%)</td>
<td>1.45</td>
<td>1.61</td>
<td>4.86</td>
<td>1.56</td>
<td>4.10</td>
<td>1.60</td>
<td>1.27</td>
</tr>
<tr>
<td>P@1 (%)</td>
<td>0.87</td>
<td>0.96</td>
<td>3.40</td>
<td>1.09</td>
<td>4.80</td>
<td>1.19</td>
<td>0.43</td>
<td>4.46</td>
</tr>
<tr>
<td>P@3 (%)</td>
<td>1.48</td>
<td>1.47</td>
<td>4.87</td>
<td>3.25</td>
<td>5.27</td>
<td>1.90</td>
<td>1.59</td>
<td>5.12</td>
</tr>
<tr>
<td>P@5 (%)</td>
<td>1.29</td>
<td>1.25</td>
<td>3.46</td>
<td>3.09</td>
<td>5.36</td>
<td>1.96</td>
<td>1.30</td>
<td>5.57</td>
</tr>
</tbody>
</table>

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-
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.

### 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


