1 INTRODUCTION

Multi-label learning aims to build classification models for objects assigned with multiple class labels simultaneously [45]. Multi-label objects widely exist in various real-world applications, such as text categorization where a news document could cover several topics including politics, economics, and reform [31], [39], [40], multimedia content annotation where one image could demonstrate several scenes including beach and building [3], [4], [49], bioinformatics where one gene could have a number of functionalities including metabolism, protein synthesis, and transcription [1], [5], [13].

Formally, let $X = \mathbb{R}^d$ denote the $d$-dimensional input space and $Y = \{l_1, l_2, \ldots, l_q\}$ denote the label space with $q$ class labels. Then, the task of multi-label learning is to derive a multi-label classification function $h : X \rightarrow 2^Y$ which assigns each instance $x \in X$ with a set of relevant labels $h(x) \subseteq Y$. In recent years, significant amount of learning approaches have been proposed to dealing with multi-label data [59]. One common strategy adopted by existing approaches is that all the class labels in $Y$ are discriminated based on identical feature representation of the instance, i.e. $x$. In other words, a family of $q$ functions $\{f_1, f_2, \ldots, f_q\}$ are induced from the multi-label training examples, where each function $f_k : X \rightarrow \mathbb{R}$ determines the class membership of $l_k$ to each instance, i.e. $h(x) = \{l_k \mid f_k(x) > 0, 1 \leq k \leq q\}$. Here, each function in the family manipulates with the same feature set $x$.

Although the above strategy has been successful in designing many multi-label learning algorithms [59], it might be suboptimal as each class label is supposed to possess specific characteristics of its own. For example, in automatic image annotation, context-based features would be preferred in discriminating sky and non-sky images, texture-based features would be preferred in discriminating desert and non-desert images, while both context- and texture-based features might be useful in discriminating other labels in the label space. For another example, in text categorization, features related to word terms such as government, national security and presidential election would be informative in discriminating political and non-political documents, while features related to word terms such as GDP, tax reduction and stock markets would be informative in discriminating economic and non-economic documents.

Therefore, we hypothesize that if label-specific features, i.e. the most pertinent and discriminative features for each class label, could be used in the learning process, a more effective approach to learning from multi-label data could be achieved. In this paper, a new algorithm named LIFT, i.e. multi-label learning with Label specific Features, is proposed. Briefly, LIFT learns from multi-label data with two intuitive steps. Firstly, for each class label $l_k \in Y$, clustering analysis is performed on its positive as well as negative training instances, and then features specific to $l_k$ are constructed by querying the clustering results. Secondly, a family of $q$ classifiers are induced with each of them being derived from the generated label-specific features other than the original ones.

To thoroughly evaluate the performance of the proposed approach, comparative studies over seventeen regular-scale and large-scale data sets have been conducted. Experimental results show that: (a) LIFT achieves
superior performance against several state-of-the-art multi-label learning algorithms; (b) LIFT’s label-specific features have the potential of being a general strategy to improve multi-label learning algorithms comprising a number of binary classifiers; (c) LIFT’s label-specific features are highly comparable to other feature manipulation mechanisms including feature selection [17], meta-level features [52] and shared subspace [23], [24].

The rest of this paper is organized as follows. Section 2 briefly reviews existing approaches to multi-label learning. Section 3 presents the proposed LIFT approach. Section 4 reports comparative experimental results over a wide range of multi-label data sets. Finally, Section 5 concludes and discusses several issues for future work.

2 Related Work

Early researches on multi-label learning originate from the investigation of multi-label text categorization techniques [31], [40], [46]. In recent years, multi-label learning has drawn much attentions from machine learning and related communities, which has been successfully applied to diverse applications such as bioinformatics [1], [5], multimedia contents analysis including image [3], [4], [49], audio [27], [34], [42], and video [35], [41], web mining [25], [33], [36], information retrieval [52], [54], etc. Accordingly, significant amount of algorithms have been proposed to learning from multi-label data [45], [59].

Generally speaking, the task of inducing multi-label classification functions is challenging as the classifier’s output space is exponential in size to the number of possible class labels (i.e. $2^{|Y|} = 2^q$). To cope with this issue, one common strategy adopted by existing approaches is to exploit label correlations to facilitate the learning process. Based on the order of correlations being considered, existing approaches can be roughly grouped into three major categories [56], namely first-order approaches, second-order approaches and high-order approaches.

First-order approaches tackle multi-label learning problem by decomposing it into a number of independent binary classification problems. Here, one function $f_k$ is learned for each possible class label $l_k$ ($1 \leq k \leq q$) by ignoring the co-existence of other labels $l_j$ ($j \neq k$). For any multi-label training example $(x, Y)$ ($x \in \mathcal{X}, Y \subseteq \mathcal{Y}$), $x$ will be regarded as one positive example to learn $f_k$ if $l_k \in Y$, and one negative example otherwise. Based on those training examples, $f_k$ can be induced with popular learning techniques such as AdaBoost [40], $k$-nearest neighbor [58], kernel methods [3], decision trees [8], [9], etc. The major advantage of first-order approaches lies in their conceptual simplicity and high efficiency. While on the other hand, these approaches could be less effective due to their ignorance of label correlations.

Second-order approaches tackle multi-label learning problem by exploiting pairwise relationships between the labels. One way to consider pairwise relationship is to impose the ranking criterion that for any multi-label training example $(x, Y)$, given its pair of relevant label $l_k \in Y$ and irrelevant label $l_{k'} \notin Y$, $f_k$ should yield larger output than $f_{k'}$ on $x$. The ranking criterion can be incorporated into the objective function to be optimized by learning models such as support vector machines [13], neural networks [28], [57]. Another way to consider pairwise relationship is to exploit the co-occurrence patterns over each of the $(q\choose2)$ label pairs $(l_k, l_j)$ ($k \neq j$). Label co-occurrence patterns can be adopted as expectation constraints for maximum entropy classifiers [15], [60], or be utilized to decompose multi-label learning problem into $(q\choose2)$ pairwise comparison problems [14], [29]. Second-order approaches address label correlations to certain extent and thus are relatively effective. However, in real-world scenarios label correlations could be rather complex and go beyond second-order.

High-order approaches tackle multi-label learning problem by exploring high-order relationships among the labels. One straightforward choice is to model interactions among all class labels, i.e. to consider all other labels’ influences on each label. This choice can be accomplished by assuming linear combination [7], nonlinear mapping [16], [32], or shared subspace [24], [51] over the whole label space. Another choice is to model interactions among a subset of class labels instead of all of them. This choice can be accomplished by selecting the label subsets randomly [26], [38], [43] or by determining the label subsets specified by graph structure [2], [18], [56]. Apparently, high-order approaches have stronger correlation-modeling capabilities than the first-order and second-order counterparts. Nevertheless, these approaches would be more computationally demanding and less scalable.

As reviewed above, existing approaches have the common property of handling multi-label data by focusing on the output space, while identical feature set inherited from the original input space is employed in discriminating all the class labels. In the next section, a new approach named LIFT is proposed which handles multi-label data by focusing on the input space via label-specific features.

3 The LIFT Approach

3.1 Algorithmic Details

Given a set of $m$ multi-label training examples $\mathcal{D} = \{(x_i, Y_i) \mid 1 \leq i \leq m\}$, where $x_i \in \mathcal{X}$ is a $d$-dimensional feature vector and $Y_i \subseteq \mathcal{Y}$ is the set of relevant labels associated with $x_i$. Then, LIFT learns from $\mathcal{D}$ by taking two elementary steps, i.e. label-specific features construction and classification models induction.

In the first step, LIFT aims to construct features which could effectively capture the specific characteristics of each label, so as to provide appropriate distinguishing information to facilitate its discrimination process. To achieve this, it is necessary to investigate the underlying properties of the training instances with respect to each
class label. Specifically, for one class label $l_k \in \mathcal{Y}$, the
set of positive training instances $\mathcal{P}_k$ as well as the set of
negative training instances $\mathcal{N}_k$ correspond to:
$$
\begin{aligned}
\mathcal{P}_k &= \{x_i \mid (x_i, Y_i) \in D, l_k \in Y_i\}
\mathcal{N}_k &= \{x_i \mid (x_i, Y_i) \in D, l_k \notin Y_i\}
\end{aligned}
(1)
$$
In other words, $\mathcal{P}_k$ and $\mathcal{N}_k$ consist of the training
instances with and without label $l_k$ respectively.
To gain insights on the properties of $\mathcal{P}_k$ and $\mathcal{N}_k$, LIFT
chooses to employ clustering techniques which have been widely
used as stand-alone tools for data analysis. In this paper, the popular $k$-means
algorithm [22] is adopted due to its effectiveness and simplicity. Therefore,
suppose $\mathcal{P}_k$ is partitioned into $m_k^+$ disjoint clusters whose
centers are denoted as $\{p_{1k}, p_{2k}, \ldots, p_{m_k^+k}\}$. Similarly, $\mathcal{N}_k
is also partitioned into $m_k^-$ disjoint clusters whose centers are
denoted as $\{n_{1k}, n_{2k}, \ldots, n_{m_k^-k}\}$. Multi-label learning
tasks usually encounter the issue of class-imbalance [59], where
the number of positive instances for each class label is much smaller than the number of negative ones,
i.e. $|\mathcal{P}_k| \ll |\mathcal{N}_k|$. To mitigate potential risks brought
by the class-imbalance problem, LIFT sets equivalent number of
clusters for $\mathcal{P}_k$ and $\mathcal{N}_k$, i.e. $m_k^+ = m_k^- = m_k$.
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 $\mathcal{P}_k$ and $\mathcal{N}_k$ is set as follows:
$$m_k = \lfloor r \cdot \min (|\mathcal{P}_k|, |\mathcal{N}_k|) \rfloor
(2)$$
Here, $| \cdot |$ returns the set cardinality and $r \in [0, 1]$ is a
ratio parameter controlling the number of clusters being retained.
Conceptually, cluster centers identified by the $k$-means
algorithm characterize the underlying structure of the
training instances with regard to $l_k$, which can be served as
appropriate building blocks (prototypes) for the
construction of label-specific features. Here, a mapping $\phi_k : \mathcal{X} \rightarrow \mathcal{Z}_k$ from the original $\dim\mathcal{X}$-dimensional input space
$\mathcal{X}$ to the $2m_k$-dimensional label-specific feature space is
created as follows: $^1$
$$
\phi_k(x) = 
\begin{bmatrix}
d(x, p_{1k}^k), \ldots, d(x, p_{m_k^+k}^k),
d(x, n_{1k}^k), \ldots, d(x, n_{m_k^-k}^k)
\end{bmatrix}
(3)
$$
Here, $d(\cdot, \cdot)$ returns the distance between two instances and
is set to Euclidean metric in this paper.
In the second step, a family of $q$ classification models
$\{g_1, g_2, \ldots, g_q\}$ are induced with the generated label-specific
features. Here, for each class label $l_k \in \mathcal{Y}$, a new
binary training set $B_k$ with $m$ examples is created from
the original multi-label training set $D$ and the mapping $\phi_k$ as
follows:
$$B_k = \{(\phi_k(x_i), Y_i(k)) \mid (x_i, Y_i) \in D\}
\text{ where }
Y_i(k) = +1 \text{ if } l_k \in Y_i; \text{ Otherwise, } Y_i(k) = -1
(4)$$
Based on $B_k$, any binary learner $\mathcal{L}$ can be applied to
induce a classification model $g_k : \mathcal{Z}_k \rightarrow \mathbb{R}$ for $l_k$.
Given an unseen example $u \in \mathcal{X}$, its associated label
set is predicted as:
$$Y = \{l_k \mid g_k(\phi_k(u)) > 0, 1 \leq k \leq q\}
(5)$$
In other words, classification model $f_k$ corresponding to
each label $l_k$ can be viewed as the composition of $g_k$ and
$\phi_k$, i.e. $f_k(u) = [g_k \circ \phi_k](u) = g_k(\phi_k(u))$.
Table 1 illustrates the complete procedure of LIFT.

<table>
<thead>
<tr>
<th>TABLE 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pseudo-code of LIFT.</td>
</tr>
<tr>
<td><strong>Inputs:</strong></td>
</tr>
</tbody>
</table>
| $D$ : multi-label training set $\{(x_i, Y_i) \mid 1 \leq i \leq m\}$
$such that (x_i, Y_i) \in D, X = \mathbb{R}^d, Y = \{l_1, l_2, \ldots, l_q\}$
$r$ : ratio parameter as used in Eq.(2)
$\mathcal{L}$ : binary learner for classifier induction
$u$ : unseen example $(u \in X)$
| **Outputs:** |
| $Y$ : predicted label set for $u$ ($Y \subseteq \mathcal{Y}$) |
| **Process:** |
| 1. for $k = 1$ to $q$ do |
| 2. Form $\mathcal{P}_k$ and $\mathcal{N}_k$ based on $D$ according to Eq.(1);
3. Perform $k$-means clustering on $\mathcal{P}_k$ and $\mathcal{N}_k$, each
with $m_k$ clusters as defined in Eq.(2);
4. Create the mapping $\phi_k$ for $l_k$ according to Eq.(3);
5. endfor |
| 6. for $k = 1$ to $q$ do |
| 7. Form $B_k$ according to Eq.(4);
8. Induce $g_k$ by invoking $\mathcal{L}$ on $B_k$, i.e. $g_k \leftarrow \mathcal{L}(B_k)$;
9. endfor |
| 10. Return $Y$ according to Eq.(5) |

1. As discussed above, the distribution of positive instances and neg-
itive instances for each cluster is usually imbalanced with $|\mathcal{P}_k| \ll |\mathcal{N}_k|$.
Thus, the dimensionality of $\mathcal{Z}_k$, i.e. $2 \cdot \lfloor r \cdot \min (|\mathcal{P}_k|, |\mathcal{N}_k|) \rfloor$, would be
of reasonable size in most cases. As an instance, for the
bibtext data set with 7935 examples and 1836 features (Table 2), the dimensionality
of the label-specific feature space is only around 23 ± 20 across all labels
with $r = 0.1$.

3.2 Remarks
In terms of label-specific features construction, the
process shown in Table 1 (steps 1 to 5) only represents an
intuitive implementation and is not meant to be
the best possible practice for constructing label-specific
features. Actually, the mapping $\phi_k$ can be implemented in numerous alternative ways, such as setting different number of clusters for positive and negative instances (i.e. $m_k^+ \neq m_k^-$), utilizing more sophisticated distance for $d(\cdot, \cdot)$ other than the Euclidean metric, or even employing kNN rule [47] to identify the prototypes for feature mapping other than invoking the $k$-means procedure, etc.\footnote{URL 1: \url{http://mulan.sourceforge.net/datasets.html} URL 2: \url{http://meka.sourceforge.net/#datasets} URL 3: \url{http://cse.seu.edu.cn/people/zhangml/Resources.htm#data}}; In terms of classification models induction, the process shown in Table 1 (steps 6 to 9) is similar to those of the first-order approaches as discussed in Section 2. The major difference lies that LIFT induces the classifier on $l_2$ with label-specific feature set $\phi_k(x)$ instead of the original feature set $x$.

In general sense, LIFT embodies three major merits that any practically useful algorithm is desirable to have: 1) \textit{Flexibility}: Besides the label-specific features which could be generated in various ways as discussed above, the classification models could also be induced with various binary learners $\mathcal{L}$ to meet different requirements (e.g. $\mathcal{L}$="decision tree" for low training cost; $\mathcal{L}$="rule learner" for good comprehensibility, etc.); 2) \textit{Simplicity}: As shown in Table 1, the LIFT algorithm is succinct and easy to implement. Specifically, LIFT is affiliated with only one single parameter (i.e. $r$) which keeps the algorithm away from the sophisticated (and often tricky) issue of tuning a number of parameters simultaneously; 3) \textit{Effectiveness}: As to be reported in Section 4, LIFT shows highly competitive performance against the state-of-the-art multi-label learning methods. To convincingly validate the effectiveness of LIFT, a total of seventeen benchmark multi-label data sets have been employed for experimental studies.

2. Several variants of LIFT have been studied in Subsection 4.4.2.

4 \textbf{EXPERIMENTS}

4.1 \textbf{Experimental Configuration}

4.1.1 \textbf{Data Sets}

For each data set $S = \{(x_i, Y_i) \mid 1 \leq i \leq p\}$, we use $|S|$, $\text{dim}(S)$, $L(S)$ and $F(S)$ to denote the number of examples, number of features, number of possible class labels, and feature type for $S$ respectively. In addition, several other multi-label properties \cite{38, 45} are denoted as:

- $\text{LCard}(S) = \frac{1}{p} \sum_{i=1}^{p} |Y_i|$: label cardinality which measures the average number of labels per example;
- $\text{LDen}(S) = \frac{\text{LCard}(S)}{|S|}$: label density which normalizes $\text{LCard}(S)$ by the number of possible labels;
- $\text{DL}(S) = |\{Y \mid (x, Y) \in S\}|$: distinct label sets which counts the number of distinct label combinations in $S$;
- $\text{PDL}(S) = \frac{\text{DL}(S)}{|S|}$: proportion of distinct label sets which normalizes $\text{DL}(S)$ by the number of examples.

Table 2 summarizes detailed characteristics of all multi-label data sets used in the experiments. Roughly ordered by $|S|$, eight regular-scale data sets (first part, $|S| \leq 5000$) as well as nine large-scale data sets (second part, $|S| > 5000$) are included. Furthermore, dimensionality reduction is performed on three text data sets with huge number of features ($\text{dim}(S) > 47000$), including rcv1 (subset 1), rcv1 (subset 2), and tmce2007. Specifically, the top 2% features with highest document frequency \cite{53} are retained.\footnote{As a routine process, stop words frequently appearing in the document (e.g. the function words) are not included in the vocabulary. Furthermore, existing studies show that based on document frequency, no loss will be incurred by retaining 10% features and just a small loss will be incurred by retaining 1% features \cite{59, 57}.”}

\begin{table}[h]
\centering
\caption{Characteristics of the experimental data sets.}
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline
Data set & $|S|$ & $\text{dim}(S)$ & $L(S)$ & $F(S)$ & $\text{LCard}(S)$ & $\text{LDen}(S)$ & $\text{DL}(S)$ & $\text{PDL}(S)$ & Domain & URL* \\
\hline
CAL500 & 502 & 68 & 174 & numeric & 26.044 & 0.150 & 502 & 1.000 & music & URL 1 \\
language log & 1460 & 1004 & 75 & nominal & 1.180 & 0.016 & 286 & 0.196 & text & URL 2 \\
enron & 1702 & 1001 & 53 & nominal & 3.378 & 0.064 & 753 & 0.442 & text & URL 2 \\
image & 2000 & 294 & 5 & numeric & 1.236 & 0.247 & 20 & 0.010 & images & URL 3 \\
scene & 2407 & 294 & 6 & numeric & 1.074 & 0.179 & 15 & 0.006 & images & URL 1 \\
yeast & 2417 & 103 & 14 & numeric & 4.237 & 0.303 & 198 & 0.082 & biology & URL 3 \\
slashdot & 3782 & 1079 & 22 & nominal & 1.181 & 0.054 & 156 & 0.041 & text & URL 2 \\
corel5k & 5000 & 499 & 374 & nominal & 3.522 & 0.009 & 3175 & 0.635 & images & URL 1 \\
rcv1 (subset 1) & 6000 & 944 & 101 & numeric & 2.880 & 0.029 & 1028 & 0.171 & text & URL 1 \\
rcv1 (subset 2) & 6000 & 944 & 101 & numeric & 2.634 & 0.026 & 954 & 0.159 & text & URL 1 \\
bibtex & 7395 & 1836 & 159 & numeric & 2.402 & 0.015 & 2856 & 0.386 & text & URL 1 \\
eurlex(directory code) & 19348 & 5000 & 412 & numeric & 1.292 & 0.003 & 1565 & 0.084 & text & URL 1 \\
eurlex(subject matter) & 19348 & 5000 & 201 & numeric & 2.213 & 0.011 & 2504 & 0.129 & text & URL 1 \\
corel16k (sample 1) & 13766 & 500 & 153 & nominal & 2.859 & 0.019 & 4803 & 0.349 & images & URL 1 \\
mediamill & 43907 & 120 & 101 & numeric & 4.376 & 0.043 & 6555 & 0.149 & video & URL 1 \\
scene & 2407 & 294 & 6 & numeric & 1.074 & 0.179 & 15 & 0.006 & images & URL 1 \\
corel5k (subset 1) & 6000 & 944 & 101 & numeric & 2.880 & 0.029 & 1028 & 0.171 & text & URL 1 \\
rcv1 (subset 2) & 6000 & 944 & 101 & numeric & 2.634 & 0.026 & 954 & 0.159 & text & URL 1 \\
bibtex & 7395 & 1836 & 159 & numeric & 2.402 & 0.015 & 2856 & 0.386 & text & URL 1 \\
eurlex(directory code) & 19348 & 5000 & 412 & numeric & 1.292 & 0.003 & 1565 & 0.084 & text & URL 1 \\
eurlex(subject matter) & 19348 & 5000 & 201 & numeric & 2.213 & 0.011 & 2504 & 0.129 & text & URL 1 \\
corel16k (sample 1) & 13766 & 500 & 153 & nominal & 2.859 & 0.019 & 4803 & 0.349 & images & URL 1 \\
mediamill & 43907 & 120 & 101 & numeric & 4.376 & 0.043 & 6555 & 0.149 & video & URL 1 \\

\end{tabular}
\end{table}
As shown in Table 2, the seventeen data sets cover a broad range of cases with diversified multi-label properties. To the best of our knowledge, few works on multi-label learning have conducted experimental evaluation across such broad range of data sets. Few notable exceptions are [30] and [38] where a total of 11 and 15 data sets have been considered respectively. Therefore, experimental studies reported in this paper are quite comprehensive which aim to provide a solid basis for thorough evaluation of LIFT’s effectiveness.\(^1\)

4.1.2 Evaluation Metrics

Performance evaluation on multi-label learning algorithms is somewhat complicated as each object is associated with multiple class labels simultaneously, where traditional single-label criteria such as accuracy, precision, recall, etc. can not be directly applied [59]. Given the test data set \(T = \{(x_i, Y_i) \mid 1 \leq i \leq t\}\) and the family of \(q\) learned functions \(\{f_1, f_2, \cdots, f_q\}\), five evaluation metrics widely-used in multi-label learning [45], [59] are employed in this paper:

• **Hamming loss**:

\[
hloss = \frac{1}{l} \sum_{i=1}^{t} |h(x_i) \Delta Y_i|
\]

Here, \(h(x_i) = \{l_k \mid f_k(x_i) > 0, 1 \leq k \leq q\}\) corresponds to the predicted set of relevant labels for \(x_i\), and \(\Delta\) stands for the symmetric difference between two sets. Hamming loss evaluates the fraction of instance-label pairs which have been misclassified, i.e. a relevant label is missed or an irrelevant label is predicted.

• **One-error**:

\[
one-\text{error} = \frac{1}{l} \sum_{i=1}^{t} |\{\arg \max_{l_k \in S} f_k(x_i)\} \not\subseteq Y_i|
\]

Here, for any predicate \(\pi\), \([\pi]\) returns 1 if \(\pi\) holds and 0 otherwise. One-error evaluates the fraction of examples whose top-ranked predicted label is not in the ground-truth relevant label set.

• **Coverage**:

\[
\text{coverage} = \frac{1}{q} \left( \frac{1}{l} \sum_{i=1}^{t} \max \{\text{rank}(x_i, l_k)\} - 1 \right)
\]

Here, \(\text{rank}(x_i, l_k) = \sum_{j=1}^{q} 1[f_j(x_i) \geq f_k(x_i)]\) returns the rank of \(l_k\) when all class labels in \(Y\) are sorted in descending order according to \(\{f_1(x_i), f_2(x_i), \cdots, f_q(x_i)\}\). Coverage evaluates how many steps are needed, on average, to move down the ranked label list of an example so as to cover all its relevant labels. Furthermore, the coverage metric is normalized by the number of possible class labels (i.e. \(q\)) in this paper.

• **Ranking loss**:

\[
rloss = \frac{1}{l} \sum_{i=1}^{t} \frac{|\{(l_k, l_j) \mid f_k(x_i) \leq f_j(x_i), (l_k, l_j) \in Y_i \times Y_i\}|}{\|Y_i\|}
\]

Here, \(\bar{Y}_i\) is the complementary set of \(Y_i\) in \(\mathcal{Y}\). Ranking loss evaluates the average fraction of misordered label pairs, i.e. an irrelevant label of an example is ranked higher than its relevant one.

• **Average precision**:

\[
\text{avgprec} = \frac{1}{l} \sum_{i=1}^{t} \frac{1}{\|Y_i\|} \sum_{l_k \in Y_i} \frac{|R(x_i, l_k)|}{\text{rank}(x_i, l_k)}
\]

Here, \(R(x_i, l_k) = \{l_j \mid \text{rank}(x_i, l_j) \leq \text{rank}(x_i, l_k), l_j \in Y_i\}\)

Average precision evaluates the average fraction of relevant labels ranked higher than a particular label \(l_k \in Y_i\).

The above metrics are example-based ones which work by evaluating the classification models’ performance on each test example separately, and then returning the averaged value across the test set. Specifically, **hamming loss** considers how \(\{f_1, f_2, \cdots, f_q\}\) perform in terms of classification quality, while the other four metrics considers how they perform in terms of ranking quality.

In addition to example-based metrics, **label-based metrics** work by evaluating the binary classification performance on each class label separately, and then returning the averaged value across all class labels. In this paper, we choose to employ the AUC criterion (area under the ROC curve) for performance evaluation on each class label:\(^5\)

• **Macro-averaging AUC**:

\[
\text{AUC}_{\text{macro}} = \frac{1}{q} \sum_{k=1}^{q} \text{AUC}_k
\]

\[
= \frac{1}{q} \sum_{k=1}^{q} \frac{|\{(x', x'') \mid f_k(x'') \geq f_k(x'), (x', x'') \in \mathcal{P}_k \times \mathcal{N}_k\}|}{|\mathcal{P}_k| \cdot |\mathcal{N}_k|}
\]

Here, the AUC value on each class label (i.e. \(\text{AUC}_k\)) is calculated based on the relationship between AUC and the Wilcoxon-Mann-Whitney statistic [20].

Note that for all the six multi-label metrics, their values vary between [0,1]. Furthermore, for average precision and macro-averaging AUC, the larger the values the better the performance; While for the other four metrics, the smaller the values the better the performance. These metrics serve as good indicators for comprehensive comparative studies as they evaluate the performance of the learned models from various aspects.

\(^1\) For brevity, in the rest of this paper we use rcv1-s1, rcv1-s2, corel16k-s1, corel16k-s2, eurex-sm and eurex-ak to rename data sets rcv1 (subset 1), rcv1 (subset 2), corel16k (sample 1), corel16k (sample 2), eurex (subject matter), and eurex (directory code) respectively.

\(^5\) Here, it is possible to employ other single-label criteria such as accuracy, precision, recall, etc. to yield the label-based metric. In this paper, AUC criterion is used due to its capability of evaluating binary classification performance in a more comprehensive way.
In this subsection, we compare Lift against several state-of-the-art multi-label learning approaches to validate its predictive performance:

- **Binary relevance (Br)** [3]: This is a first-order approach which decomposes the multi-label learning problem into \( q \) independent binary classification problems. Binary relevance could be viewed as a plain version of Lift where the label-specific features \( \phi_k(x) \) is kept to the original features \( x \).

- **Calibrated label ranking (CLR)** [14]: This is a second-order approach which transforms the multi-label learning problem into a label ranking problem, where pairwise comparison [21] is employed to train \( \mathcal{L} \) binary classifiers to yield the ranking among labels which are further bi-partitioned via an embedded calibration mechanism.

- **Ensembles of classifier chains (ECC)** [38]: This is a high-order approach which transforms the multi-label learning problem into a chain of binary classification problems, where binary classifiers in the chain are successively built upon the predictions of preceding ones. In addition, ensemble learning is employed to address chain order randomness. Here, the ensemble size is set to be 100 to accommodate sufficient number of classifier chains.

- **Random k-labelsets (RAKEL)** [43]: This is a high-order approach which transforms the multi-label learning problem into an ensemble of multi-class classification problems, where each multi-class classification problem is generated by applying the label powerset techniques [37], [59] on a randomly chosen \( k \)-labelset in \( 3^q \). Here, the ensemble size is set to be \( 2q \) with \( k = 3 \) as suggested in the literature.

The only parameter of Lift, i.e. ratio \( r \) as used in Eq.(2), is set to be 0.1 in this paper. Furthermore, for fair comparison, LIBSVM (with linear kernel) [6] is employed.

### 4.2 Comparative Studies

<table>
<thead>
<tr>
<th>Comparing algorithm</th>
<th>CAL500 language log enron image scene yeast slash dot corel5k</th>
<th>Hamming loss ( \downarrow )</th>
<th>Coverage ( \uparrow )</th>
<th>Micro-averaging AUC ( \uparrow )</th>
<th>Macro-averaging AUC ( \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lift</td>
<td>0.137±0.002 0.015±0.001 0.048±0.001 0.016±0.003 0.084±0.002 0.197±0.002 0.160±0.001 0.019±0.001</td>
<td>0.003 0.003 0.001 0.003 0.001 0.003 0.001 0.001</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
<td>0.492 0.492 0.492 0.492 0.492 0.492 0.492 0.492</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
</tr>
<tr>
<td>Br</td>
<td>0.137±0.002 0.015±0.001 0.048±0.001 0.016±0.003 0.084±0.002 0.197±0.002 0.160±0.001 0.019±0.001</td>
<td>0.003 0.003 0.001 0.003 0.001 0.003 0.001 0.001</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
<td>0.492 0.492 0.492 0.492 0.492 0.492 0.492 0.492</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
</tr>
<tr>
<td>CLR</td>
<td>0.137±0.002 0.015±0.001 0.048±0.001 0.016±0.003 0.084±0.002 0.197±0.002 0.160±0.001 0.019±0.001</td>
<td>0.003 0.003 0.001 0.003 0.001 0.003 0.001 0.001</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
<td>0.492 0.492 0.492 0.492 0.492 0.492 0.492 0.492</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
</tr>
<tr>
<td>ECC</td>
<td>0.137±0.002 0.015±0.001 0.048±0.001 0.016±0.003 0.084±0.002 0.197±0.002 0.160±0.001 0.019±0.001</td>
<td>0.003 0.003 0.001 0.003 0.001 0.003 0.001 0.001</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
<td>0.492 0.492 0.492 0.492 0.492 0.492 0.492 0.492</td>
<td>0.971 0.971 0.971 0.971 0.971 0.971 0.971 0.971</td>
</tr>
</tbody>
</table>

**Table 3**: Predictive performance of each comparing algorithm (mean±std. deviation) on the eight regular-scale data sets.
as the binary learner for binary classifier induction for LIFT, BR, CLR and ECC.

Tables 3 and 4 report the detailed experimental results of each comparing algorithm on the regular-scale and large-scale data sets respectively. On each data set, 50% examples are randomly sampled without replacement to form the training set, and the rest 50% examples are used to form the test set. The sampling process is repeated for ten times and the average predictive performance across ten training/testing trials are recorded. For each evaluation metric, "↑" indicates "the larger the better" while "↓" indicates "the smaller the better" while "\( \pm \)" denote the average rank denoted the average rank \( \#\text{ data sets} \).

Tables 3 and 4 report the detailed experimental results of each comparing algorithm on the regular-scale and large-scale data sets respectively. On each data set, 50% examples are randomly sampled without replacement to form the training set, and the rest 50% examples are used to form the test set. The sampling process is repeated for ten times and the average predictive performance across ten training/testing trials are recorded. For each evaluation metric, "↑" indicates "the larger the better" while "↓" indicates "the smaller the better" while "\( \pm \)" denote the average rank denoted the average rank \( \#\text{ data sets} \).

To perform performance analysis among the comparing algorithms systematically, Friedman test [10] is employed here which is widely-accepted as the favorable statistical test for comparisons of multiple algorithms over a number of data sets. Given \( k \) comparing algorithms and \( N \) data sets, let \( r_{ij} \) denote the rank of the \( j \)-th algorithm on the \( i \)-th data set (mean ranks are shared in case of ties). Let \( R_{k} = \frac{1}{N} \sum_{i=1}^{N} r_{ij} \) denote the average rank.

![Table 4](image)

**TABLE 4** Predictive performance of each comparing algorithm (mean±std. deviation) on the nine large-scale data sets.

<table>
<thead>
<tr>
<th>Comparing algorithm</th>
<th>Hamming loss ↓</th>
<th>One-error ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcv1-s1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rcv1-s2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bibtex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>corel1k-s1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>corel1k-s2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eurlex-sm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>eurlex-dc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tm2007</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mediamill</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Table 5](image)

**TABLE 5** Summary of the Friedman statistics \( F_{P} \) (\( k = 5, N = 17 \)) and the critical value in terms of each evaluation metric (\( k: \# \text{ comparing algorithms}; N: \# \text{ data sets} \)).

<table>
<thead>
<tr>
<th>Evaluation metric</th>
<th>( F_{P} )</th>
<th>critical value (( \alpha = 0.05 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hamming loss</td>
<td>9.0555</td>
<td>2.5153</td>
</tr>
<tr>
<td>One-error</td>
<td>52.1303</td>
<td></td>
</tr>
<tr>
<td>Coverage</td>
<td>256.8024</td>
<td></td>
</tr>
<tr>
<td>Ranking loss</td>
<td>256.8024</td>
<td></td>
</tr>
<tr>
<td>Average precision</td>
<td>115.3636</td>
<td></td>
</tr>
<tr>
<td>Macro-averaging AUC</td>
<td>77.6981</td>
<td></td>
</tr>
</tbody>
</table>
Fig. 1. Comparison of LIFT (control algorithm) against other comparing algorithms with the Bonferroni-Dunn test. Algorithms not connected with LIFT in the CD diagram are considered to have significantly different performance from the control algorithm (significance level $\alpha = 0.05$).

Table 5 summarizes the Friedman statistics $F_F$ and the corresponding critical values on each evaluation metric. As shown in Table 5, at significance level $\alpha = 0.05$, the null hypothesis of “equal” performance among the comparing algorithms is clearly rejected in terms of each evaluation metric. Consequently, we need to proceed with certain post-hoc test [10] to further analyze the relative performance among the comparing algorithms. As we are interested in whether the proposed LIFT approach achieves competitive performance against other state-of-the-art approaches, the Bonferroni-Dunn test [12] is employed to serve the above purpose by treating LIFT as the control algorithm. Here, the difference between the average ranks of LIFT and one comparing algorithm is compared with the following critical difference (CD):

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$  \hspace{1cm} (6)

For Bonferroni-Dunn test, we have $q_\alpha = 2.498$ at significance level $\alpha = 0.05$ and thus $\text{CD} = 1.3547 \,(k = 5, \, N = 17)$. Accordingly, the performance between LIFT and one comparing algorithm is deemed to be significantly different if their average ranks over all data sets differ by at least one CD.

To visually present the relative performance of LIFT and other comparing algorithms, Fig. 1 illustrates the CD diagrams [10] on each evaluation metric, where the average rank of each comparing algorithm is marked along the axis (lower ranks to the right). In each subfigure, any comparing algorithm whose average rank is within one CD to that of LIFT is interconnected with a thick line. Otherwise, any algorithm not connected with LIFT is considered to have significantly different performance between each other.

Based on the above results, the following observations can be made:

1) As shown in Fig. 1, LIFT significantly outperforms BR in terms of each evaluation metric. As BR can be regarded a plain version of LIFT by keeping the original feature vector untouched, the superior performance of LIFT against BR clearly verifies the effectiveness of employing label-specific features.

2) As shown in Fig. 1, LIFT achieves statistically superior or at least comparable performance against ECC and RAKEL in terms of each evaluation metric. Note that ensemble learning strategy has been
incorporated into ECC and RAKEL to deal with the inherent randomness in their learning procedure, similar strategy may also be utilized by LIFT to account for the randomness in its clustering procedure (Table 1, step 3).

3) Furthermore, Fig. 1 shows that LIFT achieves comparable performance against CLR in terms of each evaluation metric. However, CLR needs to train \( \frac{q}{2} \) binary classifiers leading to quadratic (i.e. \( O(q^2) \)) computational complexity.

4) As shown in Tables 3 and 4, across all evaluation metrics, LIFT ranks 1st in 50.0% cases on the text data sets (language log, enron, slashdot, rcv1-s1, rcv1-s2, bibtex, eurlex-sm, eurlex-dc and tmc2007). On the other hand, LIFT ranks 1st in more than 65.0% cases on the images data sets (image, scene, corel5k, corel16k-s1 and corel16k-s2). These results indicate that LIFT tends to work better in application domains with dense feature representation (e.g. images) than those with sparse feature representation.
4.3 Generality of Label-Specific Features

As shown in the above subsection, LIFT can significantly improve the performance of Br by employing label-specific features in inducing each binary classifier. Specifically, the label-specific features generation process for LIFT (Table 1, steps 2 to 4) is applicable to binary classification problem with positive training instances $\mathcal{P}$ and negative training instances $\mathcal{N}$. Therefore, label-specific features can be applied internally to any multi-label learning algorithm whenever its learning system comprises a number of binary classifiers. In this subsection, the generality of label-specific features is studied by incorporating them into two state-of-the-art multi-label learning algorithms CLR [14] and ECC [38].

CLR works by constructing a total of $\binom{q}{2}$ binary classifiers, each for a pair of possible class labels. Given a label pair $(l_j, l_k)$ ($1 \leq j < k \leq q$), the corresponding positive and negative training instances are determined as $\mathcal{P} = \{x_i \mid (x_i, Y_i) \in \mathcal{D}, l_j \in Y_i, l_k \notin Y_i\}$ and $\mathcal{N} = \{x_i \mid (x_i, Y_i) \in \mathcal{D}, l_j \notin Y_i, l_k \in Y_i\}$. By applying the label-specific generation process of LIFT on $\mathcal{P}$ and $\mathcal{N}$ while keeping other components of CLR unchanged, a Lifted version of CLR can be instantiated.

ECC works by constructing a chain of $q$ binary classifiers, where each binary classifier in the chain takes predictions of preceding ones as extra input features. To induce the binary classifier in the chain for class label $l_k$, the corresponding positive and negative training instances are determined as $\mathcal{P} = \{x_i \mid (x_i, Y_i) \in \mathcal{D}, l_k \in Y_i\}$ and $\mathcal{N} = \{x_i \mid (x_i, Y_i) \in \mathcal{D}, l_k \notin Y_i\}$. By applying the label-specific generation process of LIFT on $\mathcal{P}$ and $\mathcal{N}$ while keeping other components of ECC unchanged, a Lifted version of ECC can be instantiated.

Table 6 reports the detailed experimental results of Lifted CLR and Lifted ECC on all the experimental data sets.\(^8\) Similar to Subsection 4.2, linear kernel LIBSVM [6] is employed as the binary learner for Lifted CLR and Lifted ECC, and the ensemble size for Lifted ECC is set to be 100.

For CLR (Tables 3 and 4) and its Lifted version (Table 6), out of the 17 benchmark data sets, Lifted CLR achieves better performance on 12, 13, 12, 11, 9 and 12 of them in terms of hamming loss, one-error, coverage, ranking loss, average precision and macro-averaging AUC respectively. Accordingly, for ECC (Tables 3 and 4) and its Lifted version (Table 6), Lifted ECC only achieves better performance on 10, 9, 3, 3, 4 and 3 data sets in terms of each evaluation metric respectively.

Furthermore, to show whether the Lifted version performs significantly better than the original algorithm, the traditional Wilcoxon signed-ranks test [10], [50] is employed to serve this purpose. Table 7 summarizes the statistical test results at significance level $\alpha = 0.05$, where the $p$-values for the corresponding tests are also shown in the brackets.

As shown in Table 7, label-specific features do help improve performance of Br and CLR, where the Lifted versions achieve statistically superior or at least comparable performance in terms of each evaluation metric. On the other hand, Lifted ECC performs inferiorly to ECC in terms of coverage, ranking loss, average precision and macro-averaging AUC. One possible reason might be that the extra features introduced by preceding classifiers in the chain are not well compatible with the label-specific features created in the $k$-means clustering procedure (Table 1, step 3). In summary, these results indicate that LIFT’s label-specific features are capable of providing more useful representation than the original features, while its generality in improving multi-label learning algorithms’ performance needs to be further investigated.

4.4 Auxiliary Results

4.4.1 Properties of LIFT

In this subsection, the parameter sensitivity as well as efficiency of LIFT will be further studied. To this end, additional experiments are conducted on two regular-scale data sets image, yeast and two large-scale data sets rcv1-s2, corel16k-s2 (with the same experimental setup as

---

\(^8\) In this sense, the Lifted algorithm studied in Subsection 4.2 can be regarded as Lifted Br.
b) The performance of $L_{IF T}$ when the value of ratio parameter $r$ in terms of each evaluation metric. Specifically, the performance of $L_{IF T}$ is most cases: (a) $L_{0.1, 0.2, 0.3, 0.4}$ and $0.5$. It is obvious from Fig. 2 that $L_{IF T}$ justifies $L_{r}$ of $r$ set to be 0.1 in Subsection 4.2. To study the sensitivity of $r$ needed to be specified for $L_{IF T}$ parameter needed to be specified for $L_{r}$ increases beyond 0.1. Therefore, these observations hold for the other data sets as well.

- **Parameter Sensitivity:** As shown in Table 1, the only parameter needed to be specified for $L_{IF T}$ is $r$, which is set to be 0.1 in Subsection 4.2. To study the sensitivity of $L_{IF T}$ with respect to $r$, Fig. 2 illustrates how the performance of $L_{IF T}$ changes with increasing value of $r$ in terms of each evaluation metric. Specifically, the ratio parameter $r$ successively takes values of 0.01, 0.05, 0.1, 0.2, 0.3, 0.4 and 0.5. It is obvious from Fig. 2 that in most cases: (a) $L_{IF T}$ has slightly worse performance when the value of $r$ is small ($r = 0.01$ and $r = 0.05$); b) The performance of $L_{IF T}$ becomes stable as the value of $r$ increases beyond 0.1. Therefore, these observations justify $L_{IF T}$’s parameter setting in Subsection 4.2.

- **Execution Time:** To study the runtime efficiency of $L_{IF T}$, the execution time of each comparing algorithm (mean±std. deviation) on four benchmark multi-label data sets is recorded and shown in Table 8.

---

Table 8 records the execution time (training phase as well as testing phase) of all the comparing algorithms investigated in Subsection 4.2. Each comparing algorithm is implemented within the MULAN multi-label learning library [44], which is built upon the well-known WEKA platform [19]. A Linux server equipped with Intel Xeon CPU (48 cores @ 2.67GHz) and 250GB memory is used for supporting the experiments. As shown in Table 8, on regular-scale data sets (image, yeast), $L_{IF T}$ consumes comparable execution time to the other algorithms. On large-scale data sets (rcv1-s2, corl16k-s2), $L_{IF T}$ consumes least execution time in training phase and is comparable to CLR and more efficient than ECC in the testing phase. These results validate the efficiency of $L_{IF T}$ in learning from multi-label data.
The resulting variant of LIFT is termed as LIFT-IG.

**Meta-level features:** Another feature manipulation mechanism to multi-label learning is to derive meta-level features from the original feature space to capture relationships between instances and class labels. Here, meta-level features employed by the MLF method [52] are used to instantiate the third variant of LIFT. MLF transforms each instance \( x \) into a meta-level feature vector \( [\psi(x, l_1), \psi(x, l_2), \ldots, \psi(x, l_q)] \) with \( q \cdot (3r + 2) \) features, where \( \psi(x, l_k) \) is a \( (3r + 2) \)-dimensional meta-level representation derived from the \( r \)-nearest neighbors of \( x \) in \( P_k \). Specifically, \( \psi(x, l_k) \) corresponds to the concatenation of the \( L_2 \)-distance, \( L_1 \)-distance, and cosine similarity distance of \( x \) with respect to each of the \( r \) nearest neighbors, as well as the \( L_2 \)-distance and cosine similarity distance of \( x \) with respect to the centroid of \( P_k \). Thereafter, the mapping functions \( \phi_k \) are set by associating each meta-level representation \( \psi(x, \cdot) \) to one class label, i.e.:

\[
\phi_k(x) = \psi(x, l_k)
\]

The resulting variant of LIFT is termed as LIFT-MLF.

In addition to the above variants of LIFT, another work closely-related to label-specific features is also studied in this subsection:

**Shared subspace:** In shared subspace model [23], [24], the learning system is composed of \( q \) linear classifiers defined over an extended instance space with shared features:

\[
f_k(x) = (w_k + \Theta^T v_k)^T x \quad (1 \leq k \leq q)
\]

Here, \( \Theta \) is an \( r \times d \) linear transformation matrix with orthonormal rows \( (\Theta\Theta^T = I) \) parameterizing the \( r \)-dimensional shared subspace \( (r < d) \). Therefore, \( \Theta \cdot x \) can be regarded as a common part shared by all class labels which is handled by weight vector \( v_k \) specific to each class label. The model parameters \( \Theta, W = [w_1, \ldots, w_q] \) and \( V = [v_1, \ldots, v_q] \) are optimized by solving a regularized least squares problem, where the corresponding algorithm is termed as MLLS.

For LIFT-IG, the number of selected features (i.e. \( d_k \)) is set to be the same as the number of features constructed by LIFT (i.e. \( 2m_k \)). For LIFT-MLF, the number of nearest neighbors (i.e. \( r \)) are tuned within the range \([10, 100]\) as suggested in the literature. As a result, \( r \) is set to be 10 which turns out to yield stable performance. Similar to LIFT, LIBSVM (with linear kernel) is also employed as the binary learner for classifier induction for LIFT-IG and LIFT-MLF. For MLLS, the dimensionality of shared subspace (i.e. \( r \)) is set to be \( q - 1 \) as suggested in the literature.

Following the same experimental setup as used in Subsection 4.2, experiments on LIFT-IG, LIFT-MLF and MLLS are conducted on the regular-scale as well as large-scale data sets. Furthermore, Wilcoxon signed-ranks test [10], [50] is employed to show whether LIFT performs significantly better than LIFT-IG, LIFT-MLF and MLLS. Table 9 summarizes the statistical test results at significance level \( \alpha = 0.05 \), where the \( p \) values for the corresponding tests are also shown in the brackets.

As shown in Table 9, LIFT achieves statistically superior performance than LIFT-IG and LIFT-MLF in terms of each evaluation metric. In addition, LIFT outperforms MLLS in terms of hamming loss, ranking loss and macro-averaging AUC and is comparable to MLLS in terms of the other evaluation metrics. These results clearly validate the effectiveness of LIFT’s label-specific features generation process based on clustering analysis.

## 5 Conclusion

There have been numerous multi-label algorithms which learn from training examples by manipulating the label

### Table 9

<table>
<thead>
<tr>
<th>Comparing algorithm</th>
<th>Hamming loss</th>
<th>One-error</th>
<th>Coverage</th>
<th>Ranking loss</th>
<th>Avg. precision</th>
<th>Macro-avg. AUC</th>
</tr>
</thead>
</table>

9. Detailed experimental results on LIFT-IG, LIFT-MLF and MLLS can be found in the online complementary file.
space, i.e. exploiting label correlations. In this paper, an extension to our preliminary research [55] is presented which learns from training examples by manipulating the input space, i.e. exploiting features specific to each class label. The major contribution of our work is to utilize label-specific features for multi-label learning, which suggests a promising direction for learning from multi-label data.\textsuperscript{10}

Experiments across the largest number of benchmark data sets up to date show that: (a) LIFT achieves highly competitive performance against other state-of-the-art multi-label learning algorithms; (b) Multi-label learning algorithms comprising binary classifiers might be improved by utilizing label-specific features; (c) Exploiting label-specific features is rather effective compared to other feature manipulation mechanisms. In the future, it is interesting to design other label-specific features generation strategies, incorporate label-specific features into other multi-label learning algorithms such as ML-KNN [58], and investigate how to consider label correlations in generating label-specific features.

REFERENCES


10. Source code of LIFT (with usage guide) is publicly-available at http://cse.seu.edu.cn/PersonalPage/zhgml/files/LIFT.rar.


