Information Sciences 632 (2023) 617–636

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

LSTC: When label-specific features meet third-order label correlations

Xing-Yi Zhang [a,b], Fan Min [a,b,e,*], Guojie Song [c], Hong Yu [d]

[a] School of Computer Science, Southwest Petroleum University, Chengdu 610500, China
[b] Lab of Machine Learning, Southwest Petroleum University, Chengdu 610500, China
[c] School of Sciences, Southwest Petroleum University, Chengdu 610500, China
[d] College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[e] Institute for Artificial Intelligence, Southwest Petroleum University, Chengdu 610500, China

ARTICLE INFO

Keywords:
Label correlations
Label-specific features
Multi-label learning
Neural network

ABSTRACT

Multi-label learning (MLL) copes with applications such as text classification and image recognition where each instance is associated with multiple labels. Currently, some label-specific feature conversion approaches such as LIFT serve each individual label well, however do not take advantage of label correlation. Some label correlation exploitation approaches such as GLOCAL have independent model prediction and label correlation stages, thus limiting their learning ability. This paper introduces LSTC, a new algorithm that generates label-specific features and exploits third-order label correlation. On the one hand, for each label, we choose the same number of positive and negative representative instances inspired by density peak. These instances are used to generate a new data matrix with label-specific features. On the other hand, we train a paired output prediction network for each label based on these matrices of its own and that of two auxiliary labels. In this way, third-order label correlations are implicitly exploited. Specifically, the two auxiliary labels are the most similar and least similar, respectively, to avoid getting stuck in local optima. Experiments are conducted on seventeen benchmark datasets in comparison with ten popular algorithms. Results on eight measures demonstrate the superiority of LSTC on data from various domains except the text-domain. The source code is available at github.com/fansmale/lstc.

1. Introduction

Multi-label learning (MLL) [21] has become a popular machine learning task due to its wide range of applications. In text classification [17,21,26], MLL can simultaneously model the topical aspects of the feature data in the text, such as papers, authors, and publication venues. It provides services such as recommendation and information retrieval. In image classification [18], MLL can easily use more labels to embody a large amount of information in image compared with multi-classification. It also utilizes the connection between latent information and supervision information to achieve more accurate predictions. In video annotation [30], the use of MLL improves the reliability of video browsing, searching, and navigation. Specifically, by label correlations, the complex concepts in video can be better inferred based on the correlations with the other concepts.

* Corresponding author at: School of Computer Science, Southwest Petroleum University, Chengdu 610500, China.
E-mail address: minfan@swpu.edu.cn (F. Min).

https://doi.org/10.1016/j.ins.2023.03.056
Received 3 September 2022; Received in revised form 14 February 2023; Accepted 4 March 2023

Available online 11 March 2023
0020-0255/

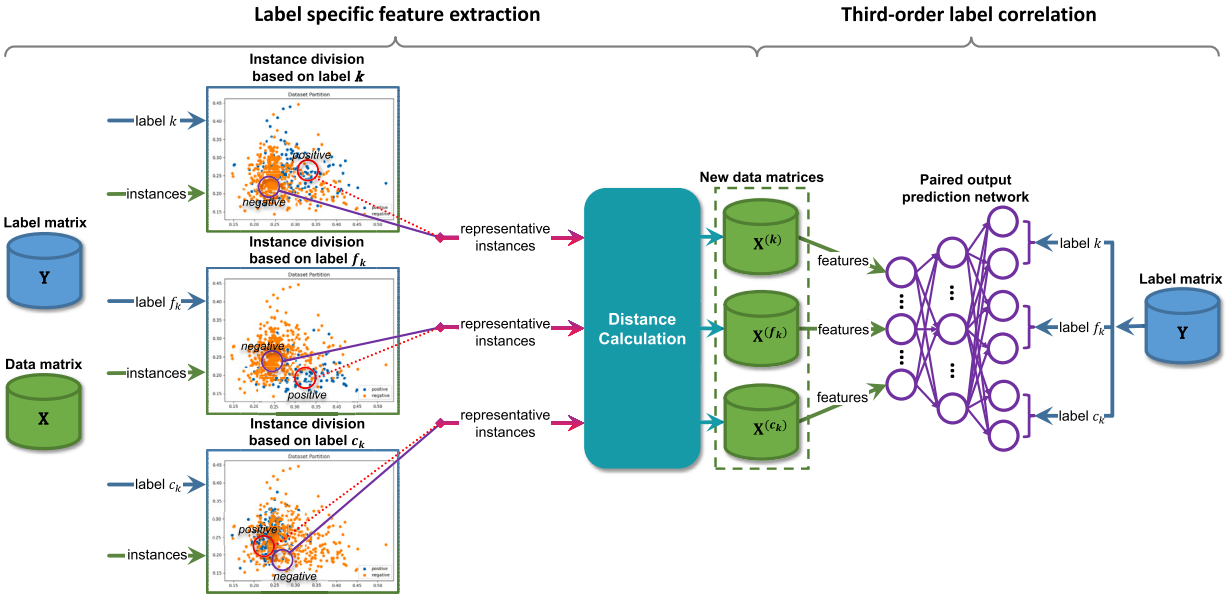


Fig. 1. The framework of LSTC. The left part is feature conversion through representative instance selection and distance calculation; the right part is the neural network training using data from the current label and two auxiliary labels.

There are various methods that deal with different characteristics of MLL tasks. When an instance contains a label (e.g., there is a cat in a given picture), we call them a positive instance and a positive label, respectively. Otherwise, we call them a negative instance and a negative label, respectively. For label sparsity [4,16,31], i.e., few (often $< 5\%$) positive labels, LIFT [45] extracts the same number of features for positive and negative instances. Deep pre-trained transformer models [4] are also good at this issue. For data sparsity, i.e., most data features are missing, CNN models [17] are often effective. For missing labels [31], DM2L [19] imposes a high-rank structure on predictions from instances with different labels to provide better discriminability. For noisy labels, [38] adopts HISC technique to handle partial multi-label without recovering ground-truth labels. For label correlation exploitation [11], ACkEL [41] transforms MLL into a set of multi-class learning problems. Meanwhile, BP-MLL [46] and MASP [22] use neural network to embody this correlation.

LIFT [45] is an original method with two main advantages. One is that it opens up a new idea of multi-label classification with embedded features, i.e., *label-specific features*. This is done by clustering the positive and negative instances of each label and then forming new features based on the cluster centers. Another is to handle label sparsity (or equivalently, imbalanced data). This is done by building an equal number of features from positive/negative instances of the imbalanced label [24]. However, LIFT does not consider label correlation, which leads to the overlook of important information.

In this paper, we propose LSTC, a new algorithm that generates label-specific features and exploits third-order label correlations. Fig. 1 presents the main framework of LSTC, which follows the feature conversion and model training scheme of LIFT. In the first stage, each label independently guides the partitioning of the data matrix $\mathbf{X}$. We select a set of representative instances in each partition through a clustering algorithm. Then, the distance between each instance in $\mathbf{X}$ and those representative instances is calculated to construct a new data matrix $\mathbf{X}^{(k)}$. In the second stage, the features of these new data matrices are fed into the network in parallel for training. Meanwhile, the new data matrices (denoted by $\mathbf{X}^{(f_k)}$ and $\mathbf{X}^{(f_c)}$) of two auxiliary labels participate in the network training of the current label. The contribution of this algorithm is three-fold.

- We adopt the density peak under Gaussian kernel to select representative instances, which are employed further to construct new features. Compared with the k-Means clustering technique employed by LIFT [45], our representative instances are not influenced by outliers. Consequently, our new features are more stable. Regarding algorithm implementation, we design techniques to obtain a trade-off between time and space complexity.
- We use data from two auxiliary labels to help train the predictive neural network for each label. In this way, the third-order label correlations are implicitly exploited. We require auxiliary labels to be the most and least similar according to Manhattan distance. The idea behind is to avoid getting stuck in local optima. Also, the network has paired outputs for more stable results. Compared with other popular label correlation approaches that convert MLL to multi-class learning [3,39,41] or consider association rules [8], our algorithm does not change the relationship between data and labels.
- For prediction, we use the softmax function to obtain numerical outputs from paired neural network outputs. Compared with simple normalization, this function is more reasonable especially when the output values are close. With numerical output, our algorithm can be compared with other ranking-based measures that are more appropriate than accuracy or F_1-measure for MLL tasks.

Table 1
Notations.

Notation	Meaning
$\mathbf{X} = [x_{ij}]_{n\times d}$	Data matrix
$\boldsymbol{x}_i = [x_{i1}, \ldots, x_{id}]$	The i-th instance
$\mathbf{Y} = [y_{ik}]_{n\times q}$	Label matrix
$\boldsymbol{y}_i = [y_{i1}, \ldots, y_{iq}]$	Labels of $\boldsymbol{x}_i$
$\mathbf{X}^{(k)} = [x_{ij}^{(k)}]_{n\times 2m_k}$	Data matrix after k-th label feature conversion
$\mathbf{P}^{(k)}$	The set of positive instances wrt. the k-th label
$\mathbf{N}^{(k)}$	The set of negative instances wrt. the k-th label
Θ_k	Network model for the k-th label

Experiments were undertaken on seventeen datasets on label-based measures including Micro-AUC and macro-AUC, sample-based measures including Hamming Loss, ranking-based measures including Peak F_1-score, NDCG, One-Error, Coverage and Ranking Loss. Results show that: 1) All components of LSTC are appropriate; 2) LSTC outperforms ten state-of-the-art algorithms on twelve datasets from various domains; and 3) LSTC is inappropriate for five text-domain data.

The rest of this paper is organized as follows. Section 2 introduces some preliminary knowledge and related work. Section 3 presents our new algorithm with complexity analysis. Section 4 presents experimental results. Finally, Section 5 draws our conclusions and outlines further research directions.

2. Related work

This section first introduces the data model used by LSTC. Then we will discuss some techniques for dealing with MLL, including feature conversion and label correlation exploitation. Some of them will be compared with LSTC in the experimental section.

2.1. Data model

Table 1 lists some important notations used throughout the paper. The data matrix is denoted by $\mathbf{X} = [x_{ij}]_{n\times d} \in \mathbb{R}^{n\times d}$, where n and d are the number of instances and features, respectively. The i-th row, denoted by $\boldsymbol{x}_i$, represents an instance, e.g., a piece of music or an image, which is described by a set of features with distinct semantics. The label matrix is denoted by $\mathbf{Y} = [y_{ik}]_{n\times q} \in \{-1, +1\}^{n\times q}$, where q is the number of labels. The i-th row, denoted by $\boldsymbol{y}_i$, is the label array of $\boldsymbol{x}_i$. $y_{ik} = +1$ indicates $\boldsymbol{x}_i$ has the k-th label, while $y_{ik} = -1$ indicates it does not have this label. Obviously, an instance $\boldsymbol{x}_i$ has q labels, and these q labels form $\boldsymbol{y}_i$.

2.2. Feature conversion and label correlation approaches to MLL

Feature conversion for MLL is more important than that of traditional classification (e.g., single-label learning). The first category is to learn label-specific features. The original method is LIFT [45], which is based on the idea that different labels are determined by different information. This is the first study that introduces label-specific feature conversion. However, LIFT does not make full use of the information between labels, nor does it remove redundant information for extracted features. LLSF [13] considers second-order label correlation by analyzing the features of two labels. The shared features of two strongly correlated labels extend the information that a classifier can learn. But LLSF weakens the possible contribution of irrelevant labels to learned information. FRS-SS-LIFT [43] adopts a fuzzy rough set and sample selection to reduce unnecessary interference in feature space. It removes the redundant information of extracted features, but does not fully consider the correlation between different labels. LSML [10] considers missing labels through learning a $q \times q$ supplementary matrix for high-order label correlations. It addresses inaccurate label correlation exploitation due to missing labels. However, it only considers positive label correlations, which may cause limitations for some datasets [5]. The second category is closely related to data clustering. CBMLC [27] decomposes a dataset into several disjoint data clusters, and extracts features for each cluster. It improves the learning speed when the number of labels is large, but reduces the data in each classifier, so the performance might degrade [25]. OMLC [28] uses a decay mechanism during training and changes features during evolution. It combines multi-label and clustering algorithms under the framework of an incremental learning approach. IMCC [34] applies data augmentation techniques to obtain virtual centers of clusters so that local label correlation can be learned. But the limited number of virtual centers obtained by clustering may limit the overall performance.

Label correlation exploitation is essential to MLL. The first category is to analyze the label matrix independently. GLOCAL [48] decomposes the label matrix into two low-rank subspaces and then builds a linear prediction model for the first subspaces. It is the first MLL algorithm to exploit global and local correlations of the label matrix, but it is unsuitable for handling asymmetric correlations. k-Means based Apriori multi-label classification [8] uses k-Means to divide pre-processing space of Apriori [1] in advance, thereby generating association rules for each cluster to independently predict each label. This algorithm shows that association rules are also good strategies for extracting information from the label matrix. LDMI [32] transforms logical label into label distribution with an approach of random variable distribution with granular computation. It can use mutual information and label enhancement to eliminate redundant and irrelevant information. But the discretization of features may also lead to the loss of key information [37]. LFCMLL [12] learns a sparse weight parameter vector for each label through a linear regression model and directly constrains the label correlation in output. This algorithm combines the reconstruction coefficient matrix with the label-specific features. It is further

demonstrated that the information matrix facilitates label correlation. The second category is to consider a set of labels in each learning subproblem. LP [3] considers a label subset each time, therefore converting MLL into a set of multi-class learning problems. But LP lacks optimization for label subsets, resulting in data starvation and high runtime. RAkEL [39] considers the runtime of LP, thus it decomposes the initial label set into length-k subsets randomly. Although the performance has been significantly improved, the class-imbalance [24] in label subset still exists. As an extension of RAkEL, ACkEL [41] considers factors such as separability and class balance in the process of label subset construction. Although ACkEL generally outperforms RAkEL, its label selection is time-consuming.

3. The proposed approach

In this section, we present our approach, which consists of feature conversion, network training, and label prediction.

3.1. Feature conversion

Similar to LIFT [45], we generate new features for each label. This is done by selecting a few representative instances. The distances between all instances and all representative instances will form a new data matrix $\mathbf{X}^{(k)}$. In this way, the number of new features is equal to the number of representative instances. Therefore, we are faced with two sub-problems: 1) how to determine the number of representative instances, and 2) how to calculate the representativeness of instances?

3.1.1. Determination of the number of representative instances

When we are concerned with the k-th label, the data matrix $\mathbf{X}$ can be split in two as

$$\mathbf{P}^{(k)} = \left\{ \boldsymbol{x}_i \mid y_{ik} = +1 \right\} \tag{1}$$

and

$$\mathbf{N}^{(k)} = \left\{ \boldsymbol{x}_i \mid y_{ik} = -1 \right\}. \tag{2}$$

Due to label sparsity, in many datasets $|\mathbf{P}^{(k)}| \ll |\mathbf{N}^{(k)}|$. We will select the same number of representative instances from $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$ to embody the equal importance of positive and negative instances. We set this number to

$$m_k = \min\left\{ R, \left\lceil r\left|\mathbf{P}^{(k)}\right|\right\rceil, \left\lceil r\left|\mathbf{N}^{(k)}\right|\right\rceil \right\}, \tag{3}$$

where R is the user-specified upper bound, r is the control ratio with domain $[0, 1]$. The idea behind is that the number of representative positive and negative instances should not exceed their respective totals, nor should they have excessive features. Consequently, the total number of representative instances is $2m_k \le 2R$.

3.1.2. Instance representativeness calculation

We compute instance representativeness inspired by the density peak clustering algorithm [33]. There are two factors [40] that characterize the instance representativeness: 1) its density, and 2) its independency (i.e., its distance to master).

When we consider all instances in $\mathbf{X}$, we obtain the *global density* under the Gaussian kernel as

$$\rho_i = \sum_{j \neq i} e^{-\left(\frac{d_{ij}}{d_c}\right)^2}, \tag{4}$$

where d_{ij} is the Euclidean distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and d_c is a custom threshold describing the degree of distance decay.

However, when dealing with data conversion for the k-th label, we would like to calculate the density of $\boldsymbol{x}_i$ considering $\mathbf{P}^{(k)}$ if $y_{ik} = +1$, and $\mathbf{N}^{(k)}$ otherwise. The *local density* is therefore given by

$$\rho_i^{(k)} = \begin{cases} \sum_{\boldsymbol{x}_j \in \mathbf{P}^{(k)} \setminus \{\boldsymbol{x}_i\}} e^{-\left(\frac{d_{ij}}{d_c}\right)^2}, & \text{if } y_{ik} = +1; \\ \sum_{\boldsymbol{x}_j \in \mathbf{N}^{(k)} \setminus \{\boldsymbol{x}_i\}} e^{-\left(\frac{d_{ij}}{d_c}\right)^2} = \rho_i - \sum_{\boldsymbol{x}_j \in \mathbf{P}^{(k)}} e^{-\left(\frac{d_{ij}}{d_c}\right)^2}, & \text{otherwise,} \end{cases} \tag{5}$$

where the expression for the second condition is rewritten for reducing time complexity, as will discussed later.

The independency of $\boldsymbol{x}_i$ is given by

$$\delta_i^{(k)} = \min_{y_{jk} = y_{ik}, \rho_j^{(k)} > \rho_i^{(k)}} d_{ij}, \tag{6}$$

where $y_{jk} = y_{ik}$ indicates that the instance is only compared these with the same class value in terms of the k-th label. Let

$$z = \operatorname*{argmin}_{y_{jk} = y_{ik}, \rho_j^{(k)} > \rho_i^{(k)}} d_{ij}, \tag{7}$$

then $\boldsymbol{x}_z$ is called the *local master* of $\boldsymbol{x}_i$ [40].

The representativeness of $\boldsymbol{x}_i$ wrt. label k is therefore computed by

$$\gamma_i^{(k)} = \rho_i^{(k)} \delta_i^{(k)}. \tag{8}$$

3.1.3. New data matrix construction

According to representativeness, we obtain the most representative m_k positive instances $\mathbf{P}_c^{(k)} = \{\boldsymbol{p}_1^k, \ldots, \boldsymbol{p}_{m_k}^k\} \subset \mathbf{P}^{(k)}$ and the most representative m_k negative instances $\mathbf{N}_c^{(k)} = \{\boldsymbol{n}_1^k, \ldots, \boldsymbol{n}_{m_k}^k\} \subset \mathbf{N}^{(k)}$. The new data matrix is given by

$$\mathbf{X}^{(k)} = \left(x_{ij}^{(k)}\right)_{n \times 2m_k}, \tag{9}$$

where

$$x_{ij}^{(k)} = \begin{cases} d(\boldsymbol{x}_i, \boldsymbol{p}_j^k), & \text{if } j \le m_k; \\ d(\boldsymbol{x}_i, \boldsymbol{n}_{j-m_k}^k), & \text{otherwise,} \end{cases} \tag{10}$$

where $d(\boldsymbol{x}_i, \boldsymbol{p}_j^k)$ stand for Euclidean distance between instance $\boldsymbol{x}_i$ and representative instances $\boldsymbol{p}_j^k$. It is a feasible approach to alleviate the single-label imbalance by replacing features with distances between general instances and positive/negative instances [24,45].

3.1.4. Algorithm description and analysis

Algorithm 1 LSTC feature conversion for all labels.

Input: Data matrix $\mathbf{X}$, label matrix $\mathbf{Y}$, target label k.
Output: The new data matrices for all labels.

1: Compute ρ_i for $1 \le i \le n$ according to Eq. (4);
2: **for** $1 \le k \le q$ **do**
3: Form $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$ based on k-th label according to Eqs. (1) and (2);
4: Compute the density $\rho_i^{(k)}$ for each $\boldsymbol{x}_i$ in $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$ according to Eq. (5);
5: Compute the independence $\delta_i^{(k)}$ according to Eq. (6);
6: Compute the representativeness $\gamma_i^{(k)}$ according to Eq. (8);
7: Construct the most representative m_k instances $\{\boldsymbol{p}_1^k, \ldots, \boldsymbol{p}_{m_k}^k\}$ in $\mathbf{P}^{(k)}$ and $\{\boldsymbol{n}_1^k, \ldots, \boldsymbol{n}_{m_k}^k\}$ in $\mathbf{N}^{(k)}$ according to $\gamma_i^{(k)}$;
8: Construct $\mathbf{X}^{(k)}$ through all instances in $\mathbf{X}$ and $2m_k$ representative instances extracted for the k-th label according to Eqs. (9) and (10);
9: **end for**
10: Return $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(q)})$;

Algorithm 1 illustrates the complete process of LSTC feature conversion for all labels. Step 1 goes through preprocessing to obtain the global density. Step 3 splits the data matrix $\mathbf{X}$ into positive and negative matrices $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$. Steps 4 to 6 obtain representativeness by sequentially calculating the local density and independence for both matrices $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$. Steps 7 to 8 calculates the distance between the representative instances and each instance to construct $\mathbf{X}^{(k)}$. Finally, we calculate $\mathbf{X}^{(k)}$ corresponding to each label, obtain $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(q)})$ and return.

Now we consider the time and space complexity of Algorithm 1. First, in order to save some time overhead, we will pre-calculate global density ρ_i according to (4). The time complexity of calculating each of instance is $O(dn)$. So the time complexity of global preprocessing is $O(dn^2)$. At the same time, during this process, we calculate the distance between all instances. However, these distance values actually will be reused in subsequent calculations. Therefore, in order to avoid multiple calculation, we will consume $O(n^2)$ space complexity to store them.

For each label k, the time complexity of feature conversion can be described in five parts. For simplicity, in the following description we will replace $|\mathbf{N}^{(k)}|$ and $|\mathbf{P}^{(k)}|$ with s_n and s_p.

1) According to Eqs. (1) and (2), the time complexity of dividing the data matrix $\mathbf{X}$ based on positive/negative labels is $O(n)$.
2) After dividing the positive/negative matrices, the representative selection strategy is performed on the two matrices respectively.
   a) For the negative matrix $\mathbf{N}^{(k)}$, we can assist the local density $\rho_i^{(k)}$ computation with global density ρ_i according to Eq. (5). Meanwhile, we also can know that the time complexity of computing its all local densities $\rho_i^{(k)}$ is $O(s_n s_p)$.
   b) For the positive matrix $\mathbf{P}^{(k)}$, since the number of instances is sufficiently small compared to n. Therefore, all local densities can be calculated by traversing the matrix $\mathbf{P}^{(k)}$ twice. Then the time complexity is $O(s_p^2)$.
3) According to Eq. (6), determining the master of any instance $\boldsymbol{x}_i$ also requires traversing the entire matrix with time complexity $O(dn)$. Therefore, the total time complexity of determining each $\delta_i^{(k)}$ of $\mathbf{P}^{(k)}$ and $\mathbf{N}^{(k)}$ is $O(s_p^2 + s_n^2)$.
4) The time complexity of obtaining a representativeness array is $O(s_p + s_n) = O(n)$ through Eq. (8). We can then use a *simple selection sort* with complexity $O(R(s_p + s_n) = O(Rn)$ to filter out the most representative R instances.
5) Then according to Eqs. (9) and (10), the time complexity of constructing the new matrix $\mathbf{X}^{(k)}$ is $O(2Rn)$.

Let $K = \max_{1 \le k \le q} s_p$, in case of sparse labels, $K \ll n$. Similarly, because of label sparsity, we can also approximate $s_p^2 + s_n^2$ as n^2. The total time complexity is shown in Table 2.

Table 2
Time complexity of Algorithm 1.

Part	Complexity	Description
Line 1	$O(dn^2)$	Compute the global density
Line 2	$O(q)$	Enumerate all labels
Line 3	$O(n)$	Divide positive/negative based on k-th label
Line 4	$O(nK)$	Density calculation
Line 5	$O(n^2)$	Independence calculation
Line 6	$O(n)$	Representativeness calculation
Line 7	$O(Rn)$	Representative instances selection
Line 8	$O(2Rn)$	Construct $\mathbf{X}^{(k)}$
Total	$O(dn^2 + q(n + nK + n^2 + n + Rn + 2Rn) = O((d + q)n^2)$	

In general, the setting of R is small compared to dn and ds_p. Therefore, the time complexity can be simplified to $O((d + q)n^2)$ on feature conversion.

Regarding the analysis of space complexity, we can find that the largest storage overhead in the whole process is to use the space complexity of $O(n^2)$ to store the distance between all instances. Although for some large datasets, such space overhead will take up too much memory. However, in this algorithm, this space-for-time scheme can greatly save the program runtime.

In general, the process of converting the original feature matrix $\mathbf{X}$ into $\mathbf{X}^{(k)}$ is similar to the LIFT algorithm, but the difference between this feature conversion approach from LIFT are two-folds.

1) LSTC adopts a more efficient and convenient density peak clustering algorithm to compute representative instances. First, the density peak algorithm is not only appropriate for spherical data. Second, the clustering center is a concrete point, not a virtual center points. Third, it is insensitive to outliers. Last, its complexity is smaller than traditional clustering schemes like k-Means.
2) Because LSTC considers the correlation of multiple labels, the final number of features input to the network is the sum of converted features of three labels, thus LSTC adopts a different control Eq. (3) for the number of clusters than LIFT. The details will be explained in Section 3.2.

3.2. Model training

We use two auxiliary matrices and the respective labels to help train the model of each label. Specifically, for label k, let the most similar and least similar labels be c_k and f_k, respectively. One is the most similar, given by

$$c_k = \underset{j \neq k}{\operatorname{argmin}} \sum_{i=1}^{n} \left| y_{ik} - y_{ij} \right|, \tag{11}$$

while the other is the least similar, given by

$$f_k = \underset{j \neq k}{\operatorname{argmax}} \sum_{i=1}^{n} \left| y_{ik} - y_{ij} \right|. \tag{12}$$

It is a natural idea that the most similar labels serve as auxiliary information. We could also use the more similar labels, such as 2NN, however the enhancement is trivial. The use of the least similar is to avoid overfitting and getting stuck in local optima. We will show that the effectiveness of technique over the 2NN technique.

The input of the Θ_k is given by $\mathbf{X}^{(k)}$, $\mathbf{X}^{(c_k)}$, and $\mathbf{X}^{(f_k)}$. Therefore, Θ_k has $2(m_k + m_{c_k} + m_{f_k})$ input nodes.

Fig. 2 illustrates an example of our prediction network. There are three parts of the input, one for each label. Then current instance is $\boldsymbol{x}_i^{(k)} = [x_{i,1}^{(k)}, \cdots, x_{i,2m_k}^{(k)}, x_{i,1}^{(c_k)}, \ldots, x_{i,2m_{c_k}}^{(c_k)}, x_{i,1}^{(f_k)}, \ldots, x_{i,2m_{f_k}}^{(f_k)}]$, which determines the features of the three related labels. Meanwhile, in order to fully embody the correlation between the k-th label and its two auxiliary labels during the training process, the output of Θ_k will be composed of three pairs of ports with domain $[0, 1]$, corresponding to label k, label c_k, and label f_k respectively.

Therefore the network Θ_k has six output nodes. The network is formally denoted by a mapping

$$F_k : \mathbb{R}^{2(m_k + m_{c_k} + m_{f_k})} \to [0, 1]^6. \tag{13}$$

Let $\hat{g}_k = (\hat{g}_k^-, \hat{g}_k^+)$ denote the two prediction outputs for the negative and positive classes, respectively. During training, the relationship between $\hat{g}_k^-(\boldsymbol{x}_i^{(k)})$ and $\hat{g}_k^+(\boldsymbol{x}_i^{(k)})$ can define $\hat{y}_{ik}^{(k)}$ given by

$$\hat{y}_{ik}^{(k)} = \begin{cases} -1, & \text{if } \hat{g}_k^-(\boldsymbol{x}_i^{(k)}) > \hat{g}_k^+(\boldsymbol{x}_i^{(k)}); \\ +1, & \text{otherwise.} \end{cases} \tag{14}$$

The predicted output of the example shown in Fig. 2 is $\hat{\boldsymbol{y}}_i^{(k)} = [+1, -1, -1]$.

Also, it is necessary to use the inverse process of Eq. (14) when training the model. When the target label is $+1$ or -1, pair value (g_k^-, g_k^+) should be set to $(0, 1)$ or $(1, 0)$. The value of six target outputs of the example shown in Fig. 2 are $[g_k^-, g_k^+, g_{c_k}^-, g_{c_k}^+, g_{f_k}^-, g_{f_k}^+] = [0, 1, 1, 0, 1, 0]$.

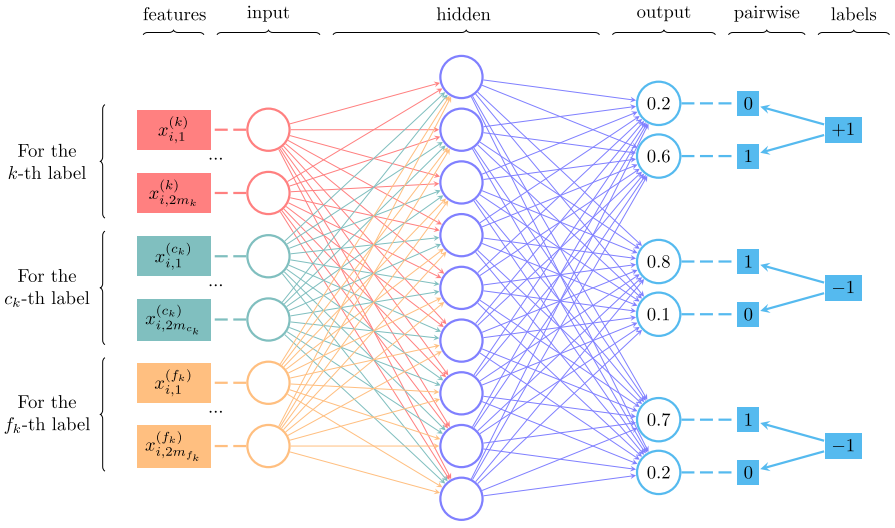


Fig. 2. A paired output network with a width of $2(m_k + m_{c_k} + m_{f_k})$ at the input and six output nodes applied to each label. The number of nodes in the hidden layer does not represent the actual scheme used and biases are omitted for brevity.

So it is possible to convert the F_k mapping to a new mapping G_k that better describes a particular k-th label output

$$\begin{aligned} G_k &: \mathbb{R}^{2(m_k+m_{c_k}+m_{f_k})} \to [0,1]^2, \text{ and} \\ F_k &= [G_k, G_{c_k}, G_{f_k}]. \end{aligned} \tag{15}$$

As shown in the example of Fig. 2, the F_k of this network Θ_k is $[G_k, G_{c_k}, G_{f_k}] = [0.2, 0.6, 0.8, 0.1, 0.7, 0.2]$.

MSE (Mean-Square Error), a loss function commonly used for regression problems, is a simple and efficient strategy for calculating loss value in many network settings. Briefly, MSE computes the error between the predicted value and the target value. Moreover, it uses the magnitude of the error as a source of penalty during back propagation of the network. This scheme is exactly the loss function used by the network in LSTC, i.e.,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(g_k(\boldsymbol{y}_i^{(k)}) - \hat{g}_k(\boldsymbol{x}_i^{(k)})\right)^2, \tag{16}$$

where $g_k(\boldsymbol{y}_i^{(k)})$ can be defined by the target labels vector $\boldsymbol{y}_i^{(k)}$ of the network, while $\hat{g}_k(\boldsymbol{x}_i^{(k)})$ can be defined by the input node vector $\boldsymbol{x}_i^{(k)}$.

When the loss function is determined, it is important to calculate the activation function backwards, layer by layer. In fact, LSTC employs more particular activation functions and layer settings for various data set properties. The configuration file of the source code contains this setting.

3.3. Prediction

The prediction approach used in Eq. (14) may be sufficient in training and computing accuracy. But it should be noted that because of general sparse for the label matrix, the idea of using Eq. (14) in actual prediction may be insufficient. Therefore, a new strategy will be adopted here to predict and evaluate the algorithm.

As shown in Fig. 3, we use softmax function to calculate the probability that label is positive

$$p_{ik} = \frac{\exp g_k^+(\boldsymbol{x}_i^{(k)})}{\exp g_k^+(\boldsymbol{x}_i^{(k)}) + \exp g_k^-(\boldsymbol{x}_i^{(k)})}. \tag{17}$$

Consequently, we have a parametric prediction

$$\hat{y}_{ik}^{(k)} = \begin{cases} -1, & \text{if } p_{ik} < \theta; \\ +1, & \text{otherwise}, \end{cases} \tag{18}$$

where the threshold θ with domain $[0,1]$ can be varied according to the requirement of the application. In fact, for ranking-based measures, we do not need to learn this parameter. However, if we need to predict logic labels, this parameter is very important.

Assuming a semi-supervised learning scenario, a model was previously built successfully from a set of n multi-label training examples $\mathbf{X}$ and $\mathbf{Y}$. Now, if there is a new testing matrix $\mathbf{U}$ with labels to be predicted, LSTC should be able to predict the label matrix $\hat{\mathbf{Y}}$ of the $\mathbf{U}$.

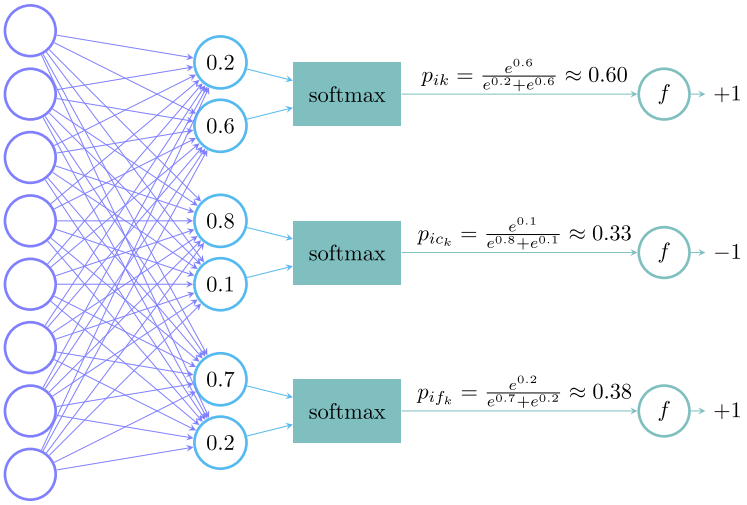


Fig. 3. The prediction process of a paired output network. Predicting from six nodes to three determined label classes requires calculating the probability that the label is positive through softmax function in Eq. (17), after then, outputs the label prediction through the threshold θ of the probability in Eq. (18) (θ in the figure is assumed to be 0.35).

If q well-established models are obtained, for any instance $\boldsymbol{u}_i$ in $\mathbf{U}$, we can use $\mathbf{P}_c^{(k)}$ and $\mathbf{N}_c^{(k)}$ defined during training to compute the $\mathbf{U}^{(k)}$ according to Eqs. (9) and (10). At the same time, it is natural to get the $\mathbf{U}^{(c_k)}$ and $\mathbf{U}^{(f_k)}$ based on two other auxiliary labels and their representative instances.

Hence $\boldsymbol{u}_i^{(k)}$ is used as the input of the network Θ_k to obtain six real-valued output. Ultimately, the output of the model is parsed into the $\hat{\boldsymbol{y}}_i^{(k)}$ by Eqs. (17) and (18).

Also, a note about the difference between training and prediction: we do not need to know the predictions of the three labels during the prediction process but only the results of the first label k. In fact, the training process only uses the auxiliary label c_k and f_k to assist in label correlation training. Consequently, when defining the model, this model already incorporates the correlation of auxiliary labels.

We can use a straightforward functional h that only concentrate on the relationship between the input and result to describe prediction process:

$$\hat{\boldsymbol{y}}_i^{(k)} = h\left(\boldsymbol{u}_i, \mathbf{P}_c^j, \mathbf{N}_c^j, \Theta_k\right), j \in \{k, f_k, c_k\}. \tag{19}$$

After training, only the first parameter in Eq. (19) is unknown. Hence given the determined independent variable $\boldsymbol{u}_i$, the only dependent variable $\hat{\boldsymbol{y}}_i^{(k)}$ must be determined. Simultaneously, if we apply Eq. (19) to each label and instance, we can get $\hat{\mathbf{Y}}$ of testing data matrix $\mathbf{U}$.

4. Experiments

We conducted experiments to analyze the effectiveness of the LSTC algorithm by answering the following questions.

1) Is the time complexity analysis of Algorithm 1 valid?
2) Are the feature conversion and label prediction modules of LSTC better than the alternatives?
3) Does LSTC outperform state-of-the-art algorithms?
4) What kind of data is LSTC not suitable for?

The experiments were conducted on a computer using Pytorch and NVIDIA GTX 1660. The relevant parameters settings of different comparison algorithm were taken from the respective publications. In addition, some parameter settings were partially adjusted to ensure the good performance.

4.1. Datasets and evaluation measures

Table 3 presents datasets covering various domains such as music, medicine, image, biology, etc. They can be downloaded for free from Mulan,[1] the research group KDIS[2] at the University of Cordoba, and PALM[3] at the Southeast University of China. Twelve

[1] http://mulan.sourceforge.net/datasets-mlc.html.
[2] https://www.uco.es/kdis/mllresources/.
[3] http://palm.seu.edu.cn/zhangml/files/Image.rar.

Table 3
The description for each dataset. Cardinality stands for the average number of positive labels for each instance. Density stands for the proportion of non-zero elements in data matrix. GpositiveGO and WaterQuality are abbreviated as Gpositive and Water, respectively.

Dataset	#instance	#label	#feature	Cardinality	Density	Domain
Birds[1]	645	19	260	1.01	0.5850	Audio
Cal500[1]	502	174	68	26.04	0.9960	Music
CHD49[2]	555	6	49	2.58	0.3947	Medicine
Emotion[1]	593	6	72	1.87	0.9893	Music
Flags[1]	194	7	19	3.39	0.4034	Image
Foodtruck[2]	407	12	21	2.29	0.8378	Recommend
Image[3]	2,000	5	294	1.24	0.9984	Image
Gpositive[2]	519	4	912	1.24	0.0091	Biology
Scene[1]	2407	6	294	1.07	0.9885	Image
VirusGo[2]	207	6	749	1.22	0.0152	Biology
Water[2]	1,060	14	16	5.07	0.9455	Chemistry
Yeast[1]	2,417	14	103	4.24	0.9992	Biology
Art[1]	5,000	26	462	1.64	0.0754	Text
Business[1]	5,000	30	438	1.59	0.0775	Text
Enron[1]	1,702	53	1,001	3.38	0.0840	Text
Recreation[1]	5,000	22	606	1.42	0.0727	Text
Social[1]	5,000	39	1047	1.28	0.0613	Text

of them are from non-text domains and five are from the text domain. The reason for this division will be discussed experimentally.

MLL usually faces label sparsity, which corresponds to class-imbalance [24] for single-label learning. Therefore, ranking-based measures may be more suitable for this task. Then we will use six of them, including Peak F_1-score, NDCG, One-Error, Coverage and Ranking Loss. Meanwhile, for the sake of objectivity, we also use label-based and sample-based measures including micro-AUC, macro-AUC and Hamming Loss [7].

4.1.1. AUC

AUC (Area Under Curve) [20] is an evaluation measure obtained by calculating the area of the ROC (Receiver Operating Characteristic) curve [35]. Meanwhile, ROC curve is obtained based on the change process of TPR and FPR of confusion matrix [29]. AUC is measured on the whole, so if we care about the global performance of prediction, AUC can be more representative than F_1-score [6]. In short, if the cost of our false positives is high, such as recommended system, we will choose AUC as the more appropriate measure. On the contrary, if the cost of missing the correct target is high, such as medical-domain, then it is better to use F_1-score. AUC is usually divided into micro-AUC and macro-AUC due to differences in calculation details. The micro-AUC considers the entire label matrix as a whole, i.e.,

$$\text{micro-AUC} = \frac{|(y', y'')|\hat{y}' \geq \hat{y}''(y', y'') \in \mathcal{P} \times \mathcal{N}|}{|\mathcal{P}||\mathcal{N}|}, \tag{20}$$

where $\mathcal{P}$ and $\mathcal{N}$ stand for the set of positive and negative label values in the label matrix $\mathbf{Y}$, respectively. This label pair (y', y'') is an element in the Cartesian product result of $\mathcal{P}_k$ and $\mathcal{N}_k$. $(\hat{y}', \hat{y}'')$ is the result of the label pair (y', y'') prediction at the original position. For macro-AUC, similar operations are limited to each label vector. In the end, we average the AUC values of each label, i.e.,

$$\text{macro-AUC} = \frac{1}{q}\sum_{k=1}^{q}\text{AUC}_k = \sum_{k=1}^{q}\frac{1}{q}\frac{|(y', y'')|\hat{y}' \geq \hat{y}'', (y', y'') \in \mathcal{P}_k \times \mathcal{N}_k|}{|\mathcal{P}_k||\mathcal{N}_k|}. \tag{21}$$

For an instance $\boldsymbol{x}_k$ without positive labels, we give $\text{AUC}_k = 0$. Conversely, if this instance $\boldsymbol{x}_k$ has no negative labels, given $\text{AUC}_k = 1$.

4.1.2. Peak F_1-score

Peak F_1-score is based on the F_1-score [6], which in turn is an evaluation measure based on precision and recall of confusion matrix [29], i.e.,

$$F_1 = \frac{2PR}{P+R}, \tag{22}$$

where P represents precision and R represents recall. However, in order to calculate precision and recall, a threshold is needed to convert numerical predictions to binary classification. Similar to ROC curve, Peak F_1-score [9,22] avoids this difficulty as follows. Labels are sorted according to their predicted probability in descendant order. By predicting the top-k labels as positive and the remaining ones as negative, we obtain respective F_1-score. By enumerating k from 0 to nq, the maximal value will be the Peak F_1-score. In this way, Peak F_1-score is threshold free.

NDCG [14] is commonly used in the evaluation of search and recommendation system. Also, it is applicable to evaluation of multi-label classification. First, DCG (Discounted Cumulative Gain) is a measure that emphasizes the influence of overall order on the prediction result. It can be expressed as the relationship between the gain g_i and the logarithm of the ordinal number, i.e.,

$$\mathrm{DCG} = \sum_{i=1}^{nq} \frac{g_i}{\log_2(i+1)} \tag{23}$$

where g_i stands for the sorted label according to their predicted probability in descendant order. Second, IDCG can be considered as the DCG in the best sorted state. Therefore, we just sort the label according to the original label matrix $\mathbf{Y}$. Assuming that there are k positive labels in the $\mathbf{Y}$ matrix, then the sorted array is all positive labels in domain $[1, k]$ and all negative labels in domain $[k+1, nq]$. IDCG can be calculated according to the idea of Eq. (23) and this array. Finally, we can describe the NDCG (Normalized Discounted Cumulative Gain) as normalized DCG, i.e.,

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}}. \tag{24}$$

The reason why we consider NDCG as the evaluation measure of this paper is to consider the following advantages of NDCG.

1) NDCG can judge whether a multi-label algorithm reasonably predicts those labels with a significant probability of being positive.
2) NDCG always increase regardless of whether individual labels are correctly predicted or not. So the penalty for wrong prediction is not very serious.
3) DCG uses logarithms to mitigate the loss, therefore compared with other ranking-based evaluation measures, the change of NDCG is relatively stable.

4.1.3. Hamming Loss

Hamming Loss can be considered as an inverse problem of accuracy. It evaluates the proportion of labels for which the predicted label matrix does not match the true label matrix, i.e.,

$$\text{Hamming Loss} = \sum_{1<i<n, 1<j<q} \frac{\lambda(h(\hat{y}_{ij}) \neq y_{ij})}{nq}, \tag{25}$$

where $h(\cdot)$ denotes

$$h(\hat{y}_{ij}) = \begin{cases} -1, & \text{if } \hat{y}_{ij} < 0.5; \\ +1, & \text{otherwise.} \end{cases} \tag{26}$$

For any predicate π, $\lambda(\pi)$ returns 1 if π holds, and 0 otherwise. Therefore, the lower the Hamming Loss is, the closer the predicted label is to the true label, then the more prominent the model prediction effect is.

4.1.4. One-Error

The One-Error evaluation measure reflects the proportion of instances with errors to the total number of instances. These erroneous instances give the label with the highest predicted probability the wrong prediction, i.e.,

$$\text{One-Error} = \frac{1}{n} \sum_{i=1}^{n} \lambda\left(y_{ik} = +1\right), k = \underset{1<j<q}{\operatorname{argmax}} \hat{y}_{ij}. \tag{27}$$

Therefore, the lower the One-Error, the better the ability to classify the highest probability label of the model. Also, we need to note that One-Error does not count instances without positive labels.

4.1.5. Coverage

Coverage discusses how many steps the ranked label list of an instance traverses to cover all positive labels in the ground truth vector, i.e.,

$$\text{Coverage} = \sum_{i=1}^{n} \text{Coverage}_i = \frac{1}{n} \sum_{i=1}^{n} \max\left\{\mathrm{rank}(\hat{\mathbf{y}}_i, j) \mid j \in \mathcal{Y}_i^+\right\} - 1. \tag{28}$$

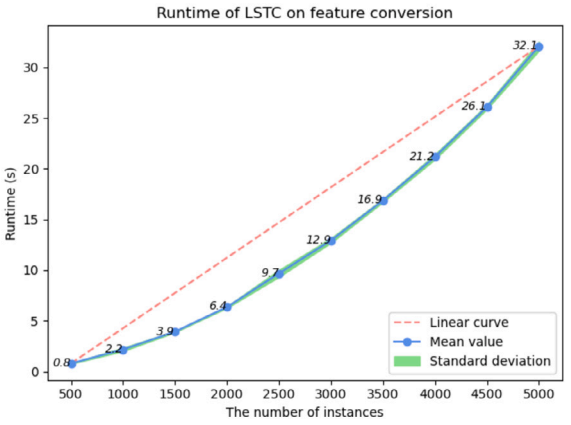
Any number j, as long as it satisfies $1 < j < q$ and belongs to the set $\mathcal{Y}_i^+$ at the same time, then $y_{ij} = +1$ must exist. First, in the function $\mathrm{rank}(\hat{\mathbf{y}}_i, j)$, the vector $\hat{\mathbf{y}}_i$ (the predicted label array of $\boldsymbol{x}_i$) will be sorted and the rank of each $\hat{y}_{ij}$ will be obtained. Second, we select those ranks that satisfy the $\mathcal{Y}_i^+$ constraints (the minimum rank is 1). In the end, we use the largest of the selected ranks as the Coverage_i. It should be noted that when $\mathcal{Y}_i^+ = \varnothing$, the Coverage_i is 0.

4.1.6. Ranking Loss

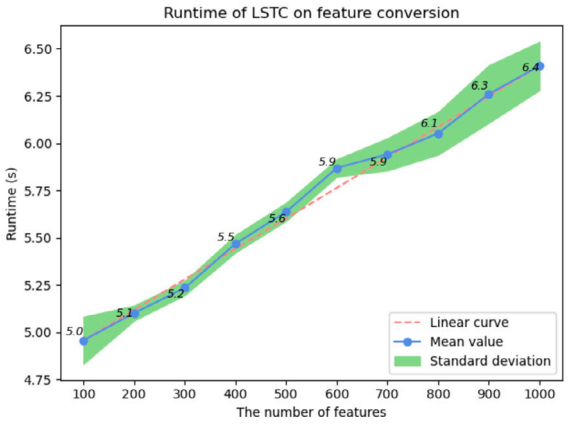
Ranking loss reflects the penalty average fraction of the model for misestimating the rank. Furthermore, an irrelevant label of an instance is ranked higher than its relevant one, i.e.,

$$\text{Ranking Loss} = \sum_{i=1}^{n} \text{Ranking Loss}_i = \sum_{i=1}^{n} \frac{1}{n} \frac{|(y', y'') | \hat{y}' \geq \hat{y}'', (y', y'') \in \mathcal{N}_i \times \mathcal{P}_i|}{|\mathcal{N}_i||\mathcal{P}_i|}. \tag{29}$$

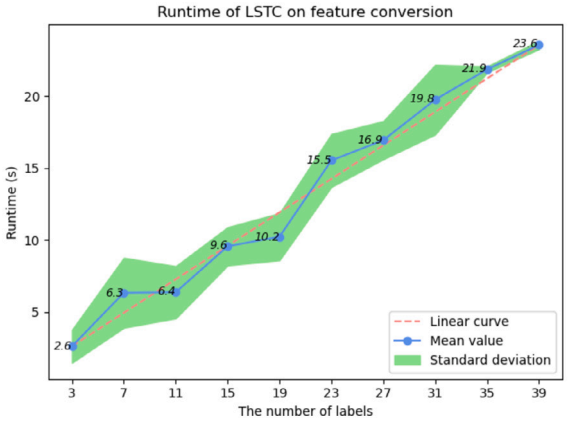
In fact, Ranking Loss can be seen as an inverse problem of macro-AUC. The actual difference between them is as follows.



(a) Runtime for different number of instances with $d = 500$ and $q = 10$.



(b) Runtime for different number of features with $n = 2000$ and $q = 10$.



(c) Runtime for different number of labels with $n = 2000$ and $d = 500$.

Fig. 4. The runtime change of feature conversion with the increase of the number of instances, features, and labels.

1) The order of the Cartesian product of sets is reversed.
2) Ranking Loss is the average evaluation of each instance, while macro-AUC is each label.

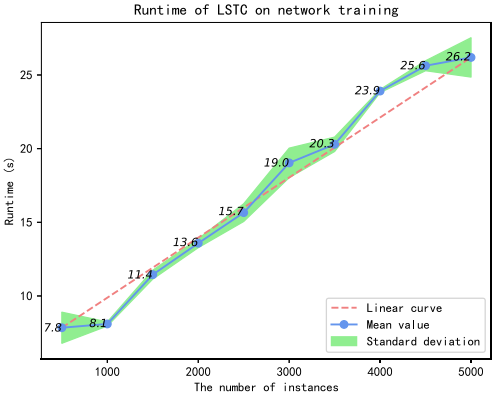
Thus the smaller the Ranking Loss, the less likely the model is to rank the labels incorrectly. Also we need to note that Ranking Loss$_i$ is not discussed when any of $|\mathcal{N}_i|$ and $|\mathcal{P}_i|$ is 0.

4.2. Runtime analysis

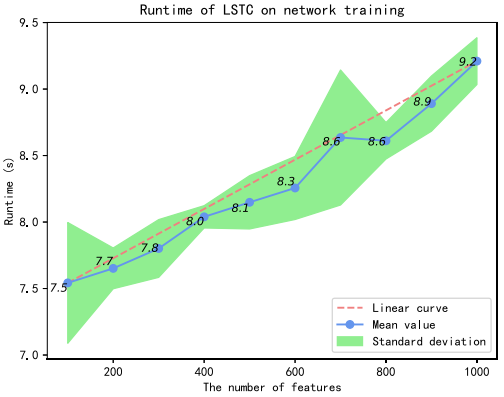
We would like to check the validity of the time complexity analysis of data conversion, as indicated by Table 2. Fig. 4 illustrates the runtime variation of the Algorithm 1 on different samples of the Social dataset. Specifically, Fig. 4(a) shows quadratic growth, while Figs. 4(b) and 4(c) show approximately linear growth. These trends of these curves are in line with the description of time complexity $O((d+q)n^2)$.

Fig. 5 presents the runtime curve of network training. Among them, the curves of Figs. 5(a) and 5(c) are more stable than curves of 5(b). Simultaneously, both of the more stable curves have larger increments than the Fig. 5(b). In fact, this is due to the deliberate design of Algorithm 1. Data conversion makes the number of features actually participating in network training have no obvious relationship with the original number. Therefore, the change in Fig. 5(b) may only be caused by the vector distance calculation with small changes and poor stability.

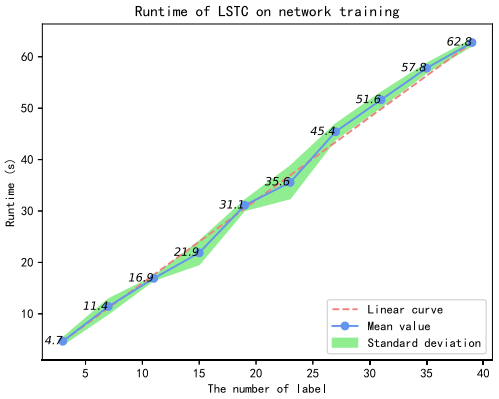
Obviously, the network training is not quite time-consuming. For example, on the biggest dataset Social we experienced, the training time is only 102 seconds, which is quite acceptable in applications.



(a) Runtime for different number of instances with $d = 100$ and $q = 5$.



(b) Runtime for different number of features with $n = 500$ and $q = 5$.



(c) Runtime for different number of labels with $n = 500$ and $d = 100$.

Fig. 5. The runtime change of network training with the increase of the number of instances, features, and labels.

4.3. Ablation studies

We undertake ablation studies to show the validity of LSTC in comparison with some alternatives.

Fig. 6 shows the ablation study between LSTC and LSTC with 2NN. The purpose of this ablation study is to explore whether the selection strategy of auxiliary labels of LSTC is effective.

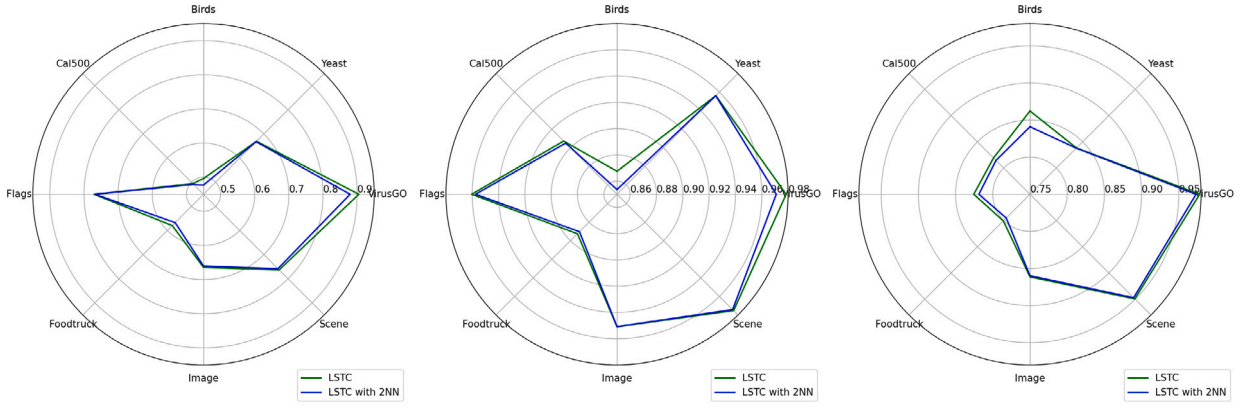
The selection strategy of LSTC is to select two similar and dissimilar labels based on Eqs. (11) and (12) to achieve label correlation and alleviate the issue of getting in local optima. On the contrary, the common approach kNN only implements the former but not the latter. Hence, we will choose 2NN (2-nearest neighbors) as ablation comparison to demonstrate that mitigating local optima is effective.

Fig. 7 shows the ablation study between LSTC and LSTC with random representative. In feature conversion, we select the most representative m_k positive/negative instances by density peak as conversion reference. In this ablation study, in order to verify the effectiveness of the density peak is effective, we will use random representative instances selection as the ablation comparison of the density peak strategy. Specifically, after obtaining $\mathbf{P}$ and $\mathbf{N}$ of the data matrix $\mathbf{X}$, m_k representative instances are randomly selected to form $\mathbf{P}_c^{(k)}$ and $\mathbf{N}_c^{(k)}$.

Fig. 8 shows the ablation study between third-order label and first-order label. The purpose of this ablation study is to explore whether it is meaningful to consider label correlations.

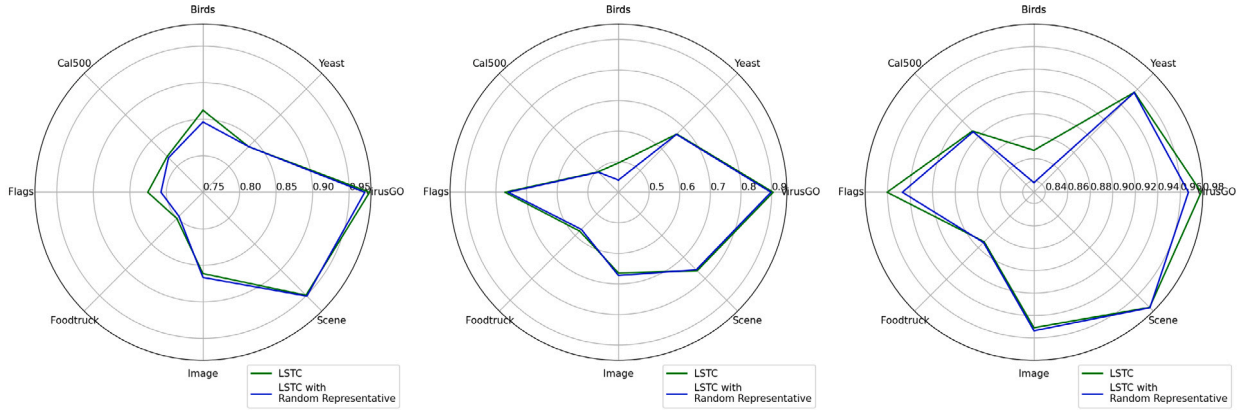
LSTC deals with third-order label strategy through two auxiliary labels and special neural network with $2(m_k + m_{c_k} + m_{f_k})$ input nodes and six output nodes. Here, if we need to use the first-order label strategy, we can make the following changes in the model training. On the one hand, we can train the model with only a single label. On the other hand, we can continue to adopt the paired output prediction network, but the input and output ports will be set to single port.

From the results in Figs. 6, 7 and 8, we observe that the ablation study that consider label correlation has a more significant boosts than the other two strategies. For example, if the first order label strategy is adopted, the mean rank of micro-AUC and NDCG



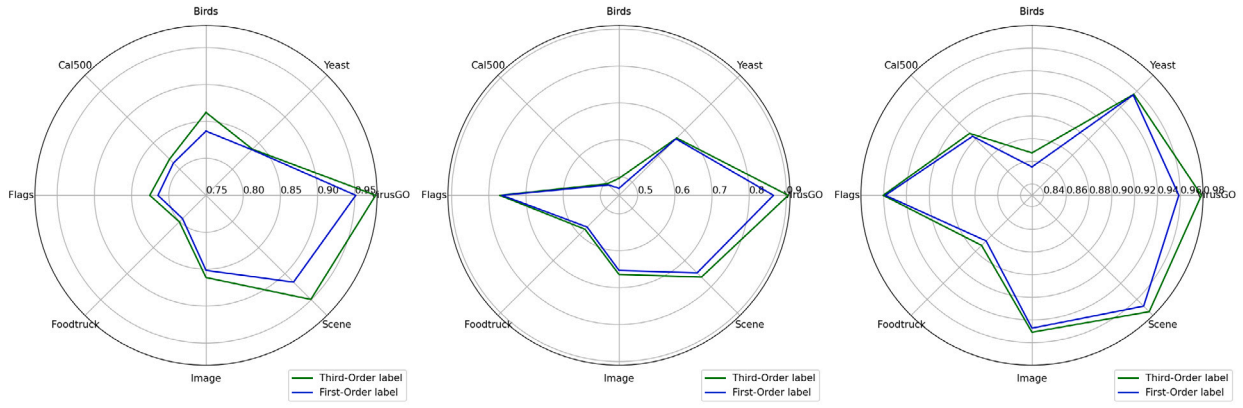
(a) Peak F_1-score (b) NDCG (c) micro-AUC

Fig. 6. Ablation study on the auxiliary label selection strategy. For brevity, only ten datasets are selected for comparison.



(a) micro-AUC (b) Peak F_1-score (c) NDCG

Fig. 7. Ablation study on representative instances selection in feature conversion.



(a) micro-AUC (b) Peak F_1-score (c) NDCG

Fig. 8. Ablation study on label correlations.

will increase from 1.50 to 3.75 and 1.58 to 3.58. Therefore, we can consider that the algorithm is significantly and reliably improved after considering the label correlation.

Although the changes in Figs. 6 and 7 are slight compared to Fig. 8, such increment is also meaningful. For example, for NDCG, a small improvement may lead to a large change in the mean rank. So if the random representative instances selection strategy is adopted, the mean rank of NDCG will still increase from 1.58 to 3.08.

4.4. Effectiveness comparison

The brief introduction and parameter settings of the comparison algorithms are as follows.

1) **BP-MLL**[4] [46] is the first multi-label neural network. It proposes a loss function for multi-label and considers label correlation exploitation. The settings are as follows: The size of the hidden layer is 20% of the input layer. epochs $= 100, \alpha = 0.05$.
2) **ML-KNN**[5] [47] is a multi-label lazy learning approach based on kNN and maximum a posteriori (MAP) principle. The settings are as follows: $k = 15$ and smooth term is set to 1.
3) **IMLLA** [44] is an improved multi-label lazy learning approach through extending label correlation and modifying nearest neighbor selection scheme based on ML-KNN. The settings are as follows: $k = 15$ and use Minkowski distance with order 1.
4) **LIFT**[6] [45] is multi-label embedding classification algorithms, also is the first algorithm to propose label-specific features through k-Means clustering. The settings are as follows: $r = 0.1$ and the SVM adopts linear scheme.
5) **LLSF**[7] [13] is a second-order MLL algorithm combining shared features of two strongly correlated labels with an improved binary classifier. The settings are as follows: $\alpha = 2^{-5}, \beta = 2^{-3}, \gamma = 0.1$.
6) **GLOCAL**[8] [48] has the ability to handle missing labels through learning a low-rank model with global and local label correlations. The settings are as follows: We cancel the setting of missing labels. $g = 3$ (the number of clusters), $k = 20$ (the latent matrix dimension), the regularization parameters $\lambda, \lambda_2, \lambda_3$ and λ_4 are set to $1, 10^{-2}, 0.125$ and 0.125, respectively.
7) **LSML**[9] [10] is augmented from the incomplete label matrix by learning a high-order label correlation to obtain label-specific features for multi-label classification with missing labels. The settings are as follows: $\lambda_1 = 10^2, \lambda_2 = 10^{-5}, \lambda_3 = 10^{-3}, \lambda_4 = 10^{-5}$.
8) **PML-NI**[10] [42] has the advantages of multi-label processing for noisy environments by introducing a noise matrix. The settings are as follows: We minimize noise here to ensure fair comparisons. $\lambda = 100, \beta = 0.5, \gamma = 0.5$.
9) **MASP**[11] [22] is a multi-label active learning through serial-parallel neural network, it can address exploitation of label correlation, missing labels and selection of query labels. The settings are as follows: We unset the missing labels and perform semi-supervised learning. The serial and parallel parts adopt the Sigmoid activation function, while the layer between them adopt the Relu activation function.
10) **LRMML**[12] [15] handles missing labels and discriminatory label correlation by using an auxiliary label matrix and low-rank label subspace constraints. The settings are as follows: We set the missing rate to 0. Meantime, we set $\lambda_T = 10^{-1}$, $\lambda_R = 2$, and $\lambda_L = 10^{-1}$, which is consistent with the optimal setting of the released code.

Table 4 lists the comparison of LSTC with different MLL algorithms under the four evaluation measures micro-AUC, macro-AUC [20], Peak F_1-score [9] and NDCG [14]. The results are obtained through 5-fold cross-validation. The higher the value, the better the effect of the algorithm.

Table 5 lists the comparison of LSTC with different MLL algorithms under the four evaluation measures Hamming Loss, One-Error, Coverage and Ranking Loss. The results are obtained through 5-fold cross-validation. The lower the value, the better the algorithm.

From Tables 4 and 5 we observe the following:

1) Overall, it can be found that the performance of LSTC generally outperforms the counterparts.
2) LSTC generally performs well in micro-AUC, macro-AUC, Peak F_1-score, NDCG and Ranking Loss measures. But relatively speaking, it performs slightly worse on the Hamming Loss, One-Error, and Coverage measures.
3) There are six datasets with outstanding performance in more than five measures. Among them, CHD49 performs well on seven evaluation measures.
4) Under certain evaluation measures, LSTC performs very well on some datasets but mediocrely on others. For example, on macro-AUC, Peak F_1-score, and NDCG, the performance of the Birds dataset of LSTC is 1.26, 1.75, and 2.57 percentage points higher than the second place, respectively. Relatively, the One-Error and Coverage measures are at a disadvantage on the Yeast dataset. Although performs very well on this dataset.

[4] http://www.lamda.nju.edu.cn/code_BPMLL.ashx.
[5] http://www.lamda.nju.edu.cn/files/MLkNN.rar.
[6] http://palm.seu.edu.cn/zhangml/Resources.htm#ijcai11.
[7] https://github.com/jiunhwang/LLSF.
[8] http://www.lamda.nju.edu.cn/code_GLOCAL.ashx.
[9] http://www.escience.cn/people/huangjun/index.html.
[10] http://www.xiemk.pro/code/PMLNIcode.zip.
[11] https://gitee.com/fansmale/masp.
[12] https://github.com/sanjayksau/lrmml-2.

Table 4
The performance of LSTC on twelve datasets under the four evaluation measures of micro-AUC, macro-AUC, Peak F_1-score, and NDCG. **Bold** indicates the best performance, and *italics* indicates the second best.

Dataset	BP-MLL'06	ML-KNN'07	IMLLA'12	LIFT'14	LLSF'15	GLOCAL'18	LSML'19	PML-NI'21	MASP'22	LRMML'22	LSTC
	micro-AUC										
Birds	.5581$_{\pm.0966}$	.7698$_{\pm.0357}$	.4367$_{\pm.0235}$	.7011$_{\pm.0288}$	.7860$_{\pm.0247}$	.5462$_{\pm.0328}$	.8204$_{\pm.0141}$	*.8537$_{\pm.0223}$*	.7592$_{\pm.0208}$	.8225$_{\pm.0156}$	**.8622$_{\pm.0131}$**
Cal500	.8009$_{\pm.0110}$	*.8149$_{\pm.0057}$*	.6793$_{\pm.0086}$	.8143$_{\pm.0039}$	.7716$_{\pm.0114}$	.8053$_{\pm.0079}$	.8142$_{\pm.0051}$	.8009$_{\pm.0078}$	.8001$_{\pm.0243}$	.7837$_{\pm.0275}$	**.8193$_{\pm.0057}$**
CHD49	.6193$_{\pm.1197}$	.7730$_{\pm.0195}$	.7721$_{\pm.0189}$	*.7882$_{\pm.0175}$*	.7687$_{\pm.0147}$	.7725$_{\pm.0118}$	.7879$_{\pm.0202}$	.7812$_{\pm.0154}$	.6598$_{\pm.0243}$	.7798$_{\pm.0243}$	**.7886$_{\pm.0207}$**
Emotion	.5750$_{\pm.0514}$	.8419$_{\pm.0326}$	.8535$_{\pm.0109}$	*.8683$_{\pm.0068}$*	.8433$_{\pm.0195}$	.8464$_{\pm.0080}$	.8573$_{\pm.0048}$	.8562$_{\pm.0116}$	.8341$_{\pm.0146}$	.8555$_{\pm.0166}$	**.8697$_{\pm.0063}$**
Flags	.5397$_{\pm.1036}$	.7236$_{\pm.0402}$	.7339$_{\pm.0095}$	.7402$_{\pm.0304}$	.7712$_{\pm.0430}$	.726$_{\pm.03590}$	.7737$_{\pm.0175}$	*.7983$_{\pm.0285}$*	.7872$_{\pm.0421}$	.7757$_{\pm.0308}$	**.8256$_{\pm.0093}$**
Foodtruck	.6897$_{\pm.0456}$	.7913$_{\pm.0305}$	.7868$_{\pm.0189}$	.7917$_{\pm.0241}$	.7980$_{\pm.0269}$	.7907$_{\pm.0216}$	*.8001$_{\pm.0241}$*	.7923$_{\pm.0229}$	.7486$_{\pm.0189}$	.8000$_{\pm.0301}$	**.8007$_{\pm.0211}$**
Image	.5152$_{\pm.0240}$	.8395$_{\pm.0090}$	.8418$_{\pm.0084}$	*.8593$_{\pm.0072}$*	.8321$_{\pm.0076}$	.8279$_{\pm.0164}$	.8270$_{\pm.0088}$	.8308$_{\pm.0113}$	.8267$_{\pm.0148}$	.8277$_{\pm.0113}$	**.8613$_{\pm.0146}$**
GpositiveGO	.7724$_{\pm.0911}$	.9879$_{\pm.0042}$	.9888$_{\pm.0058}$	.9896$_{\pm.0074}$	.9911$_{\pm.0058}$	.9853$_{\pm.0090}$	.9908$_{\pm.0029}$	.9901$_{\pm.0069}$	.8327$_{\pm.0158}$	*.9913$_{\pm.0046}$*	**.9934$_{\pm.0039}$**
Scene	.5264$_{\pm.0171}$	.9434$_{\pm.0047}$	.9435$_{\pm.0079}$	**.9504$_{\pm.0053}$**	.9139$_{\pm.0038}$	.9188$_{\pm.0080}$	.9323$_{\pm.0068}$	.9151$_{\pm.0109}$	.9394$_{\pm.0033}$	.9327$_{\pm.0041}$	*.9493$_{\pm.0061}$*
VirusGO	.8050$_{\pm.0571}$	.9492$_{\pm.0192}$	.9403$_{\pm.0199}$	.9666$_{\pm.0270}$	.9593$_{\pm.0197}$	.9331$_{\pm.0279}$	*.9846$_{\pm.0063}$*	.9796$_{\pm.0103}$	.9082$_{\pm.0248}$	**.9851$_{\pm.0081}$**	.9780$_{\pm.0177}$
WaterQuality	.5736$_{\pm.0492}$	.7415$_{\pm.0102}$	*.7507$_{\pm.0085}$*	.7484$_{\pm.0128}$	.7291$_{\pm.0073}$	.7284$_{\pm.0102}$	.7166$_{\pm.0124}$	.7253$_{\pm.0084}$	.6709$_{\pm.0104}$	.7149$_{\pm.0080}$	**.7529$_{\pm.0086}$**
Yeast	.7417$_{\pm.0461}$	**.8416$_{\pm.0113}$**	*.8406$_{\pm.0083}$*	.8351$_{\pm.0071}$	.8303$_{\pm.0018}$	.8304$_{\pm.0114}$	.8345$_{\pm.0050}$	.8312$_{\pm.0046}$	.8339$_{\pm.0085}$	.8351$_{\pm.0061}$	.8383$_{\pm.0074}$
Mean rank	10.50	5.67	6.50	4.17	6.75	7.83	4.75	5.17	8.33	5.00	1.50
	macro-AUC										
Birds	.4807$_{\pm.0354}$	.6050$_{\pm.0417}$	.3776$_{\pm.0700}$	.5528$_{\pm.0842}$	.6790$_{\pm.0354}$	.4766$_{\pm.0552}$	.7184$_{\pm.0256}$	*.7932$_{\pm.0641}$*	.7264$_{\pm.0538}$	.7749$_{\pm.0355}$	**.8058$_{\pm.0228}$**
Cal500	.4807$_{\pm.0202}$	.4944$_{\pm.0238}$	.4891$_{\pm.0103}$	.5165$_{\pm.0141}$	.5213$_{\pm.0184}$	*.5415$_{\pm.0252}$*	.5385$_{\pm.0160}$	.5201$_{\pm.0147}$	.5280$_{\pm.0244}$	.5316$_{\pm.0100}$	**.5457$_{\pm.0098}$**
CHD49	.5236$_{\pm.0567}$	.5139$_{\pm.0248}$	.5283$_{\pm.0417}$	*.6034$_{\pm.0430}$*	.5719$_{\pm.0369}$	.5798$_{\pm.0183}$	.5906$_{\pm.0135}$	.5742$_{\pm.0307}$	.5561$_{\pm.0313}$	.5842$_{\pm.0306}$	**.6126$_{\pm.0306}$**
Emotion	.5861$_{\pm.0698}$	.8257$_{\pm.0212}$	.8336$_{\pm.0192}$	*.8487$_{\pm.0195}$*	.8372$_{\pm.0151}$	.8360$_{\pm.0140}$	.8456$_{\pm.0180}$	.8333$_{\pm.0134}$	.8321$_{\pm.0162}$	.8406$_{\pm.0176}$	**.8615$_{\pm.0168}$**
Flags	.4844$_{\pm.0164}$	.5113$_{\pm.0550}$	.5360$_{\pm.0539}$	.5093$_{\pm.0629}$	.6191$_{\pm.0400}$	.5669$_{\pm.0757}$	.6442$_{\pm.0144}$	.7207$_{\pm.0538}$	*.7270$_{\pm.0383}$*	.6467$_{\pm.0405}$	**.7355$_{\pm.0186}$**
Foodtruck	.5163$_{\pm.0431}$	.5356$_{\pm.0225}$	.5685$_{\pm.0390}$	.5593$_{\pm.0352}$	.5593$_{\pm.0274}$	.5791$_{\pm.0395}$	**.5963$_{\pm.0160}$**	.5837$_{\pm.0214}$	.5581$_{\pm.0342}$	.5808$_{\pm.0295}$	*.5921$_{\pm.0345}$*
Image	.5216$_{\pm.0660}$	.8316$_{\pm.0117}$	.8395$_{\pm.0060}$	*.8565$_{\pm.0066}$*	.8261$_{\pm.0191}$	.8243$_{\pm.0111}$	.8208$_{\pm.0143}$	.8315$_{\pm.0049}$	.8323$_{\pm.0150}$	.8213$_{\pm.0111}$	**.8567$_{\pm.0124}$**
GpositiveGO	.6993$_{\pm.0848}$	.9775$_{\pm.0206}$	.9779$_{\pm.0259}$	.9881$_{\pm.0069}$	.9727$_{\pm.0241}$	.9745$_{\pm.0185}$	.9895$_{\pm.0057}$	*.9911$_{\pm.0025}$*	.9471$_{\pm.0288}$	.9857$_{\pm.0102}$	**.9918$_{\pm.0043}$**
Scene	.6160$_{\pm.0710}$	.9385$_{\pm.0072}$	.9374$_{\pm.0031}$	**.9477$_{\pm.0017}$**	.9113$_{\pm.0100}$	.9130$_{\pm.0105}$	.9284$_{\pm.0079}$	.9110$_{\pm.0061}$	.9342$_{\pm.0090}$	.9276$_{\pm.0047}$	*.9436$_{\pm.0033}$*
VirusGO	.6780$_{\pm.1193}$	.8649$_{\pm.1014}$	.9128$_{\pm.0917}$	.9381$_{\pm.0795}$	.8710$_{\pm.0708}$	.9105$_{\pm.0691}$	*.9497$_{\pm.0644}$*	.9195$_{\pm.0951}$	.8822$_{\pm.0512}$	**.9796$_{\pm.0165}$**	.9412$_{\pm.0665}$
WaterQuality	.4946$_{\pm.0483}$	.7022$_{\pm.0077}$	.7156$_{\pm.0125}$	.7158$_{\pm.0103}$	.6918$_{\pm.0148}$	.6914$_{\pm.0118}$	.6843$_{\pm.0062}$	.6894$_{\pm.0125}$	*.7207$_{\pm.0078}$*	.6833$_{\pm.0129}$	**.7229$_{\pm.0098}$**
Yeast	.5308$_{\pm.0204}$	.6879$_{\pm.0118}$	.6835$_{\pm.0102}$	.6820$_{\pm.0098}$	.6872$_{\pm.0137}$	.6915$_{\pm.0091}$	.6956$_{\pm.0122}$	.6818$_{\pm.0173}$	**.7036$_{\pm.0180}$**	*.6970$_{\pm.0115}$*	.6826$_{\pm.0151}$
Mean rank	10.75	7.58	6.75	5.08	7.00	6.42	4.42	5.83	5.58	4.75	1.92
	Peak F_1-score										
Birds	.1420$_{\pm.0294}$	.2886$_{\pm.0326}$	.1038$_{\pm.0024}$	.2312$_{\pm.0269}$	.3118$_{\pm.0194}$	.1971$_{\pm.0078}$	.3140$_{\pm.0180}$	*.4785$_{\pm.0295}$*	.2312$_{\pm.0007}$	.3195$_{\pm.0354}$	**.4960$_{\pm.0414}$**
Cal500	.4641$_{\pm.0083}$	.4810$_{\pm.0116}$	.3844$_{\pm.0075}$	.4859$_{\pm.0030}$	.4626$_{\pm.0109}$	.4856$_{\pm.0100}$	*.4907$_{\pm.0107}$*	.4846$_{\pm.0062}$	.4661$_{\pm.0261}$	.4666$_{\pm.0247}$	**.4943$_{\pm.0059}$**
CHD49	.6357$_{\pm.0605}$	.7046$_{\pm.0110}$	.7012$_{\pm.0265}$	.7078$_{\pm.0111}$	.7024$_{\pm.0302}$	.7074$_{\pm.0191}$	.7087$_{\pm.0163}$	.7056$_{\pm.0119}$	.6688$_{\pm.0192}$	*.7097$_{\pm.0241}$*	**.7157$_{\pm.0239}$**
Emotion	.5106$_{\pm.0500}$	.6882$_{\pm.0368}$	.7047$_{\pm.0118}$	*.7103$_{\pm.0065}$*	.6884$_{\pm.0294}$	.6896$_{\pm.0143}$	.7029$_{\pm.0266}$	.7004$_{\pm.0107}$	.6885$_{\pm.0221}$	.7003$_{\pm.0185}$	**.7162$_{\pm.0168}$**
Flags	.6585$_{\pm.0178}$	.7107$_{\pm.0272}$	.7122$_{\pm.0292}$	.7202$_{\pm.0210}$	.7372$_{\pm.0307}$	.6967$_{\pm.0290}$	.7343$_{\pm.0091}$	*.7514$_{\pm.0368}$*	.7447$_{\pm.0400}$	.7331$_{\pm.0249}$	**.7708$_{\pm.0118}$**
Foodtruck	.4528$_{\pm.0240}$	.5572$_{\pm.0370}$	.5623$_{\pm.0260}$	.5618$_{\pm.0415}$	.5673$_{\pm.0352}$	.5580$_{\pm.0302}$	.5660$_{\pm.0197}$	.5608$_{\pm.0359}$	.4958$_{\pm.0185}$	*.5698$_{\pm.0267}$*	**.5796$_{\pm.0214}$**
Image	.4080$_{\pm.0045}$	.6305$_{\pm.0127}$	.6327$_{\pm.0129}$	*.6581$_{\pm.0143}$*	.6319$_{\pm.0076}$	.6173$_{\pm.0224}$	.6156$_{\pm.0135}$	.6282$_{\pm.0167}$	.6364$_{\pm.0251}$	.6150$_{\pm.0152}$	**.6642$_{\pm.0125}$**
GpositiveGO	.6924$_{\pm.0730}$	.9321$_{\pm.0203}$	.9351$_{\pm.0131}$	.9387$_{\pm.0165}$	.9366$_{\pm.0255}$	.9432$_{\pm.0179}$	.9469$_{\pm.0101}$	**.9537$_{\pm.0163}$**	.6367$_{\pm.0204}$	*.9506$_{\pm.0066}$*	.9457$_{\pm.0165}$
Scene	.3190$_{\pm.0066}$	.7523$_{\pm.0177}$	.7533$_{\pm.0136}$	**.7816$_{\pm.0154}$**	.6891$_{\pm.0112}$	.6953$_{\pm.0197}$	.7240$_{\pm.0113}$	.6919$_{\pm.0176}$	.7487$_{\pm.0086}$	.7245$_{\pm.0116}$	*.7633$_{\pm.0209}$*
VirusGO	.6285$_{\pm.0483}$	.8212$_{\pm.0378}$	.8483$_{\pm.0368}$	.8743$_{\pm.0461}$	.8668$_{\pm.0402}$	.8249$_{\pm.0659}$	.9035$_{\pm.0128}$	*.9139$_{\pm.0366}$*	.8021$_{\pm.0348}$	**.9165$_{\pm.0237}$**	.9051$_{\pm.0498}$
WaterQuality	.5349$_{\pm.0097}$	.6162$_{\pm.0106}$	*.6219$_{\pm.0124}$*	.6196$_{\pm.0225}$	.6075$_{\pm.0048}$	.6055$_{\pm.0139}$	.5964$_{\pm.0205}$	.6039$_{\pm.0103}$	.3884$_{\pm.0073}$	.5964$_{\pm.0059}$	**.6241$_{\pm.0137}$**
Yeast	.5844$_{\pm.0259}$	.6714$_{\pm.0115}$	*.6715$_{\pm.0135}$*	**.6734$_{\pm.0095}$**	.6593$_{\pm.0048}$	.6646$_{\pm.0103}$	.6628$_{\pm.0134}$	.6628$_{\pm.0073}$	.6625$_{\pm.0108}$	.6643$_{\pm.0093}$	.6695$_{\pm.0102}$
Mean rank	10.58	6.83	6.08	4.00	6.83	6.92	5.17	5.08	8.00	5.00	1.75
	NDCG										
Birds	.5422$_{\pm.0239}$	.7231$_{\pm.0139}$	.5696$_{\pm.0504}$	.6501$_{\pm.0472}$	.7122$_{\pm.0126}$	.6676$_{\pm.0192}$	.6000$_{\pm.4899}$	*.8418$_{\pm.0334}$*	.8037$_{\pm.0255}$	.7331$_{\pm.0367}$	**.8675$_{\pm.0169}$**
Cal500	.8884$_{\pm.0054}$	.9018$_{\pm.0021}$	.8521$_{\pm.0036}$	.9011$_{\pm.0086}$	.8836$_{\pm.0043}$	.8992$_{\pm.0045}$	*.9044$_{\pm.0011}$*	.8981$_{\pm.0051}$	.8932$_{\pm.0139}$	.8916$_{\pm.0101}$	**.9073$_{\pm.0024}$**
CHD49	.8998$_{\pm.0133}$	.9171$_{\pm.0174}$	.9093$_{\pm.0160}$	**.9273$_{\pm.0236}$**	.9207$_{\pm.0101}$	.9194$_{\pm.0211}$	.9187$_{\pm.0119}$	*.9261$_{\pm.0086}$*	.7007$_{\pm.0334}$	.9203$_{\pm.0157}$	.9245$_{\pm.0177}$
Emotion	.7887$_{\pm.0681}$	.9363$_{\pm.0068}$	.9444$_{\pm.0084}$	**.9478$_{\pm.0063}$**	.9239$_{\pm.0123}$	.9338$_{\pm.0158}$	.9331$_{\pm.0052}$	.9365$_{\pm.0081}$	.9302$_{\pm.0085}$	.9342$_{\pm.0123}$	*.9445$_{\pm.0151}$*
Flags	.8480$_{\pm.0491}$	.9186$_{\pm.0261}$	.9189$_{\pm.0083}$	.9027$_{\pm.0240}$	.9348$_{\pm.0270}$	.9053$_{\pm.0175}$	.9396$_{\pm.0106}$	*.9552$_{\pm.0079}$*	.9524$_{\pm.0110}$	.9362$_{\pm.0227}$	**.9605$_{\pm.0070}$**
Foodtruck	.7965$_{\pm.0515}$	.8703$_{\pm.0164}$	.8708$_{\pm.0282}$	*.8909$_{\pm.0110}$*	.8837$_{\pm.0263}$	.8852$_{\pm.0160}$	.8853$_{\pm.0191}$	.8896$_{\pm.0147}$	.8401$_{\pm.0231}$	.8862$_{\pm.0191}$	**.8925$_{\pm.0193}$**
Image	.7717$_{\pm.0169}$	.9439$_{\pm.0030}$	.9449$_{\pm.0049}$	**.9526$_{\pm.0064}$**	.9307$_{\pm.0056}$	.9281$_{\pm.0052}$	.9283$_{\pm.0061}$	.9266$_{\pm.0067}$	.9351$_{\pm.0076}$	.9297$_{\pm.0067}$	*.9509$_{\pm.0057}$*
GpositiveGO	.7332$_{\pm.0700}$	.9941$_{\pm.0040}$	.9960$_{\pm.0015}$	*.9965$_{\pm.0026}$*	.9940$_{\pm.0068}$	.9899$_{\pm.0092}$	.9959$_{\pm.0018}$	.9897$_{\pm.0167}$	.9303$_{\pm.0079}$	.9956$_{\pm.0036}$	**.9974$_{\pm.0015}$**
Scene	.7423$_{\pm.0143}$	.9719$_{\pm.0047}$	.9717$_{\pm.0032}$	**.9781$_{\pm.0008}$**	.9531$_{\pm.0018}$	.9565$_{\pm.0052}$	.9649$_{\pm.0028}$	.9568$_{\pm.0033}$	.9698$_{\pm.0015}$	.9644$_{\pm.0037}$	*.9753$_{\pm.0041}$*
VirusGO	.6857$_{\pm.0628}$	.9673$_{\pm.0074}$	.9739$_{\pm.0170}$	.9680$_{\pm.0237}$	.9775$_{\pm.0074}$	.9628$_{\pm.0195}$	*.9885$_{\pm.0082}$*	.9784$_{\pm.0133}$	.9357$_{\pm.0260}$	**.9903$_{\pm.0042}$**	.9785$_{\pm.0162}$
WaterQuality	.8570$_{\pm.0133}$	.9291$_{\pm.0025}$	.9325$_{\pm.0035}$	*.9328$_{\pm.0045}$*	.9215$_{\pm.0055}$	.9204$_{\pm.0074}$	.9211$_{\pm.0044}$	.9195$_{\pm.0065}$	.8340$_{\pm.0026}$	.9211$_{\pm.0059}$	**.9332$_{\pm.0040}$**
Yeast	.8880$_{\pm.0428}$	.9554$_{\pm.0031}$	.9550$_{\pm.0053}$	*.9555$_{\pm.0017}$*	.9522$_{\pm.0024}$	.9533$_{\pm.0028}$	.9520$_{\pm.0049}$	.9519$_{\pm.0066}$	.9541$_{\pm.0036}$	.9532$_{\pm.0019}$	**.9562$_{\pm.0016}$**
Mean rank	10.67	5.50	5.92	3.42	7.00	7.42	5.92	5.75	7.42	5.50	1.58

Table 5
The performance of LSTC on twelve datasets under the four evaluation measures of Hamming Loss, One-Error, Coverage, and Ranking Loss. **Bold** indicates the best performance, and *italics* indicates the second best.

Dataset	Hamming Loss										
	BP-MLL'06	ML-KNN'07	IMLLA'12	LIFT'14	LLSF'15	GLOCAL'18	LSML'19	PML-NI'21	MASP'22	LRMML'22	LSTC
Birds	.0650$_{\pm.0205}$	*.0529$_{\pm.0062}$*	.1213$_{\pm.0174}$	.0539$_{\pm.0059}$	.0535$_{\pm.0057}$	.0979$_{\pm.0098}$	.0533$_{\pm.0046}$	.0551$_{\pm.0052}$	.0550$_{\pm.0044}$	.0534$_{\pm.0036}$	**.0480$_{\pm.0054}$**
Cal500	.1540$_{\pm.0078}$	.1382$_{\pm.0017}$	.1787$_{\pm.0028}$	*.1368$_{\pm.0034}$*	.1435$_{\pm.0022}$	.1388$_{\pm.0013}$	.1373$_{\pm.0023}$	.5683$_{\pm.3862}$	.1734$_{\pm.0162}$	.1413$_{\pm.0026}$	**.1365$_{\pm.0018}$**
CHD49	.3578$_{\pm.0431}$	.3103$_{\pm.0276}$	.3053$_{\pm.0214}$	*.2841$_{\pm.0103}$*	.3583$_{\pm.0164}$	.2974$_{\pm.0188}$	.2847$_{\pm.0125}$	.3171$_{\pm.0121}$	.3339$_{\pm.0080}$	.2877$_{\pm.0225}$	**.2823$_{\pm.0181}$**
Emotion	.3506$_{\pm.0352}$	.2097$_{\pm.0102}$	.2007$_{\pm.0146}$	**.1902$_{\pm.0148}$**	.2628$_{\pm.0050}$	.2002$_{\pm.0115}$	.1995$_{\pm.0144}$	.2010$_{\pm.0082}$	.2114$_{\pm.0097}$	.1984$_{\pm.0142}$	*.1917$_{\pm.0105}$*
Flags	.3416$_{\pm.0264}$	.3404$_{\pm.0505}$	.3327$_{\pm.0495}$	.3460$_{\pm.0130}$	.3821$_{\pm.0219}$	.3049$_{\pm.0535}$	.2959$_{\pm.0264}$	.2810$_{\pm.0194}$	*.2783$_{\pm.0280}$*	.2969$_{\pm.0281}$	**.2775$_{\pm.0141}$**
Foodtruck	.2029$_{\pm.0650}$	.1562$_{\pm.0194}$	.7521$_{\pm.0157}$	.1572$_{\pm.0180}$	.1554$_{\pm.0190}$	.7286$_{\pm.0300}$	.1627$_{\pm.0156}$	.1780$_{\pm.0192}$	.1994$_{\pm.0068}$	*.1533$_{\pm.0160}$*	**.1532$_{\pm.0105}$**
Image	.3327$_{\pm.0898}$	.1754$_{\pm.0107}$	.1756$_{\pm.0135}$	*.1594$_{\pm.0099}$*	.2264$_{\pm.0053}$	.1841$_{\pm.0054}$	.1989$_{\pm.0031}$	.1825$_{\pm.0060}$	.1754$_{\pm.0088}$	.2001$_{\pm.0063}$	**.1590$_{\pm.0071}$**
GpositiveGO	.2039$_{\pm.0296}$	.0377$_{\pm.0116}$	.0381$_{\pm.0097}$	.0347$_{\pm.0150}$	.0525$_{\pm.0071}$	.0315$_{\pm.0110}$	.0318$_{\pm.0055}$	**.0271$_{\pm.0082}$**	.0617$_{\pm.0045}$	*.0313$_{\pm.0076}$*	.0323$_{\pm.0073}$
Scene	.2244$_{\pm.0608}$	.0870$_{\pm.0062}$	.0879$_{\pm.0060}$	**.0815$_{\pm.0069}$**	.1511$_{\pm.0028}$	.1111$_{\pm.0045}$	.1048$_{\pm.0042}$	.1076$_{\pm.0077}$	.0906$_{\pm.0058}$	.1072$_{\pm.0054}$	*.0848$_{\pm.0056}$*
VirusGO	.2223$_{\pm.0546}$	.0739$_{\pm.0258}$	.0652$_{\pm.0314}$	.0549$_{\pm.0181}$	.0888$_{\pm.0209}$	.0820$_{\pm.0260}$	.0578$_{\pm.0239}$	**.0420$_{\pm.0148}$**	.0854$_{\pm.0178}$	*.0498$_{\pm.0129}$*	.0572$_{\pm.0151}$
WaterQuality	.4025$_{\pm.0368}$	.2893$_{\pm.0079}$	.6112$_{\pm.0092}$	**.2813$_{\pm.0193}$**	.3057$_{\pm.0049}$	.6288$_{\pm.0116}$	.3846$_{\pm.0164}$	.3572$_{\pm.0385}$	*.2832$_{\pm.0091}$*	.3119$_{\pm.0074}$	.2842$_{\pm.0107}$
Yeast	.2796$_{\pm.0387}$	.1974$_{\pm.0046}$	*.1972$_{\pm.0059}$*	**.1928$_{\pm.0063}$**	.2443$_{\pm.0058}$	.2000$_{\pm.0048}$	.2004$_{\pm.0023}$	.2009$_{\pm.0084}$	.2067$_{\pm.0073}$	.1999$_{\pm.0035}$	.1984$_{\pm.0058}$
Mean rank	10.00	5.08	7.17	3.33	8.42	7.08	5.08	6.25	6.92	4.58	2.17
	One-Error										
Birds	.8990$_{\pm.1070}$	.7324$_{\pm.0444}$	.7312$_{\pm.0604}$	.7386$_{\pm.0611}$	.6570$_{\pm.0729}$	.6919$_{\pm.0359}$	.6538$_{\pm.0776}$	.6514$_{\pm.0556}$	*.2853$_{\pm.0570}$*	.6793$_{\pm.0630}$	**.2589$_{\pm.0324}$**
Cal500	.2835$_{\pm.2082}$	**.1162$_{\pm.0357}$**	.4074$_{\pm.0202}$	.1509$_{\pm.0589}$	.2312$_{\pm.0425}$	.1526$_{\pm.0572}$	*.1194$_{\pm.0427}$*	.1622$_{\pm.0232}$	.1594$_{\pm.0508}$	.1989$_{\pm.0671}$	.1334$_{\pm.0235}$
CHD49	.3306$_{\pm.1530}$	*.2417$_{\pm.0682}$*	.2440$_{\pm.0613}$	.2426$_{\pm.0338}$	.2825$_{\pm.0436}$	.2702$_{\pm.0565}$	.2553$_{\pm.0307}$	.2707$_{\pm.0812}$	.3369$_{\pm.0591}$	.2699$_{\pm.0463}$	**.2360$_{\pm.0396}$**
Emotion	.6535$_{\pm.1208}$	.3019$_{\pm.0110}$	.2959$_{\pm.0598}$	*.2383$_{\pm.0598}$*	.2751$_{\pm.0507}$	.2571$_{\pm.0314}$	.2496$_{\pm.0159}$	.2561$_{\pm.0446}$	.2817$_{\pm.0245}$	.2681$_{\pm.0282}$	**.2344$_{\pm.0404}$**
Flags	.3548$_{\pm.2555}$	.2708$_{\pm.0644}$	*.2050$_{\pm.0869}$*	.2398$_{\pm.0570}$	.2618$_{\pm.0972}$	.2243$_{\pm.0791}$	.2495$_{\pm.0589}$	.2224$_{\pm.0751}$	.2883$_{\pm.0643}$	.2342$_{\pm.0349}$	**.2015$_{\pm.0469}$**
Foodtruck	.5428$_{\pm.2794}$	.2853$_{\pm.0223}$	.2977$_{\pm.0597}$	.2901$_{\pm.0647}$	.2848$_{\pm.0387}$	.2875$_{\pm.0560}$	.2754$_{\pm.0490}$	.2945$_{\pm.0494}$	.4054$_{\pm.0383}$	*.2746$_{\pm.0466}$*	**.2675$_{\pm.0611}$**
Image	.7531$_{\pm.0324}$	.3275$_{\pm.0431}$	.3264$_{\pm.0226}$	**.2732$_{\pm.0261}$**	.3190$_{\pm.0145}$	.3309$_{\pm.0212}$	.3415$_{\pm.0136}$	.3268$_{\pm.0151}$	.3190$_{\pm.0214}$	.3415$_{\pm.0146}$	*.2855$_{\pm.0180}$*
GpositiveGO	.5114$_{\pm.1449}$	.0715$_{\pm.0369}$	.0751$_{\pm.0244}$	.0636$_{\pm.0224}$	.0674$_{\pm.0227}$	.0637$_{\pm.0282}$	*.0580$_{\pm.0138}$*	.0637$_{\pm.0113}$	.1021$_{\pm.0324}$	.0636$_{\pm.0179}$	**.0539$_{\pm.0167}$**
Scene	.8196$_{\pm.0256}$	.2272$_{\pm.0113}$	.2314$_{\pm.0300}$	**.1925$_{\pm.0303}$**	.2738$_{\pm.0125}$	.2636$_{\pm.0152}$	.2301$_{\pm.0272}$	.2656$_{\pm.0189}$	.2264$_{\pm.0211}$	.2326$_{\pm.0192}$	*.2123$_{\pm.0056}$*
VirusGO	.6904$_{\pm.2085}$	.1468$_{\pm.0328}$	.1345$_{\pm.0684}$	.1159$_{\pm.0201}$	.1011$_{\pm.0402}$	.0961$_{\pm.0566}$	**.0722$_{\pm.0296}$**	.0941$_{\pm.0273}$	.2175$_{\pm.0412}$	*.0821$_{\pm.0320}$*	.0966$_{\pm.0487}$
WaterQuality	.5442$_{\pm.1713}$	.2781$_{\pm.0114}$	.2825$_{\pm.0234}$	**.2685$_{\pm.0331}$**	.3235$_{\pm.0284}$	.3118$_{\pm.0444}$	.3303$_{\pm.0211}$	.3146$_{\pm.0263}$	.2745$_{\pm.0150}$	.3273$_{\pm.0254}$	*.2689$_{\pm.0358}$*
Yeast	.5625$_{\pm.3048}$	.2345$_{\pm.0061}$	**.2214$_{\pm.0148}$**	*.2223$_{\pm.0205}$*	.2279$_{\pm.0229}$	.2227$_{\pm.0231}$	.2274$_{\pm.0143}$	.2272$_{\pm.0156}$	.2685$_{\pm.0264}$	.2292$_{\pm.0076}$	.2271$_{\pm.0291}$
Mean rank	10.83	6.42	6.42	4.00	7.00	5.75	4.83	5.75	7.25	6.08	2.00
	Coverage										
Birds	5.5472$_{\pm.3871}$	4.0621$_{\pm.3885}$	4.5363$_{\pm.7044}$	4.1560$_{\pm.8223}$	3.7023$_{\pm.3407}$	7.4132$_{\pm.4880}$	3.8527$_{\pm.2839}$	**2.5980$_{\pm.1689}$**	3.8729$_{\pm.5765}$	3.7535$_{\pm.4129}$	*2.8124$_{\pm.2288}$*
Cal500	137.1812$_{\pm.5428}$	**130.5605$_{\pm.0816}$**	162.9398$_{\pm.6650}$	133.3209$_{\pm.1576}$	148.0021$_{\pm.3322}$	133.7434$_{\pm.4446}$	134.2643$_{\pm.3957}$	142.3020$_{\pm.0200}$	141.2599$_{\pm.4585}$	144.6494$_{\pm.6256}$	*133.1585$_{\pm.6702}$*
CHD49	3.7588$_{\pm.7227}$	2.8464$_{\pm.0761}$	2.8359$_{\pm.0902}$	*2.6829$_{\pm.0834}$*	2.7405$_{\pm.0832}$	2.7461$_{\pm.1416}$	2.7243$_{\pm.1212}$	2.7565$_{\pm.1633}$	2.9928$_{\pm.0946}$	2.7045$_{\pm.0959}$	**2.6649$_{\pm.0733}$**
Emotion	2.7393$_{\pm.6446}$	1.8163$_{\pm.1202}$	1.7552$_{\pm.0844}$	*1.6963$_{\pm.1824}$*	1.8504$_{\pm.0766}$	1.8065$_{\pm.1495}$	1.7791$_{\pm.1175}$	1.8045$_{\pm.1297}$	1.8026$_{\pm.0321}$	1.7783$_{\pm.1521}$	**1.6951$_{\pm.0436}$**
Flags	4.1351$_{\pm.4898}$	4.0209$_{\pm.1942}$	3.9025$_{\pm.4274}$	3.9080$_{\pm.1149}$	3.8919$_{\pm.0783}$	3.8804$_{\pm.3755}$	*3.8800$_{\pm.2228}$*	3.9002$_{\pm.1856}$	3.9163$_{\pm.2753}$	3.8888$_{\pm.2817}$	**3.8308$_{\pm.1429}$**
Foodtruck	4.1526$_{\pm.7362}$	3.6950$_{\pm.3479}$	3.8176$_{\pm.1905}$	3.6620$_{\pm.7342}$	3.6525$_{\pm.2674}$	3.7588$_{\pm.6372}$	*3.5320$_{\pm.2847}$*	3.6805$_{\pm.4746}$	4.5958$_{\pm.2690}$	**3.5306$_{\pm.412}$**	3.6456$_{\pm.4342}$
Image	2.1799$_{\pm.0642}$	0.9686$_{\pm.0745}$	0.9675$_{\pm.0771}$	*0.8704$_{\pm.0763}$*	0.9585$_{\pm.0366}$	0.9729$_{\pm.0488}$	0.9975$_{\pm.0744}$	0.9476$_{\pm.0215}$	0.9880$_{\pm.0418}$	0.9925$_{\pm.0550}$	**0.8640$_{\pm.0757}$**
GpositiveGO	0.8537$_{\pm.3514}$	0.1024$_{\pm.0442}$	0.1176$_{\pm.0581}$	0.0982$_{\pm.0392}$	0.0924$_{\pm.0414}$	0.0954$_{\pm.0438}$	0.0847$_{\pm.0323}$	**0.0753$_{\pm.0253}$**	0.1773$_{\pm.0130}$	0.0829$_{\pm.0180}$	*0.0825$_{\pm.0211}$*
Scene	2.5753$_{\pm.1478}$	0.4601$_{\pm.0441}$	0.4772$_{\pm.0244}$	**0.3989$_{\pm.0403}$**	0.5929$_{\pm.0291}$	0.5637$_{\pm.0639}$	0.4836$_{\pm.0593}$	0.5899$_{\pm.0450}$	0.4873$_{\pm.0538}$	0.4919$_{\pm.0379}$	*0.4400$_{\pm.0367}$*
VirusGO	1.5607$_{\pm.6952}$	0.5435$_{\pm.2693}$	0.5094$_{\pm.2999}$	0.4065$_{\pm.0850}$	0.4451$_{\pm.1501}$	0.3736$_{\pm.1241}$	**0.3491$_{\pm.0748}$**	*0.3529$_{\pm.1231}$*	0.6573$_{\pm.1605}$	0.3714$_{\pm.0886}$	0.3823$_{\pm.1422}$
WaterQuality	10.7404$_{\pm.3190}$	8.7302$_{\pm.2293}$	8.6854$_{\pm.2502}$	8.7352$_{\pm.0747}$	8.9632$_{\pm.3357}$	8.9705$_{\pm.1453}$	9.1009$_{\pm.0905}$	8.9931$_{\pm.2503}$	*8.6642$_{\pm.1443}$*	9.1179$_{\pm.1371}$	**8.6292$_{\pm.2445}$**
Yeast	7.9741$_{\pm.5760}$	*6.3198$_{\pm.1826}$*	**6.2823$_{\pm.0773}$**	6.3298$_{\pm.1620}$	6.4693$_{\pm.0892}$	6.3798$_{\pm.1238}$	6.3508$_{\pm.2888}$	6.4436$_{\pm.1366}$	6.3847$_{\pm.2280}$	6.3473$_{\pm.1218}$	6.3244$_{\pm.0737}$
Mean rank	10.42	6.25	6.42	4.42	6.58	6.58	4.83	5.58	7.75	5.17	2.00
	Ranking Loss										
Birds	.3952$_{\pm.0279}$	.2985$_{\pm.0230}$	.3655$_{\pm.0432}$	.3038$_{\pm.0131}$	.2609$_{\pm.0233}$	.2855$_{\pm.0171}$	.2664$_{\pm.0260}$	**.1608$_{\pm.0211}$**	.3283$_{\pm.0303}$	.2604$_{\pm.0262}$	*.1876$_{\pm.0305}$*
Cal500	.1935$_{\pm.0075}$	.1817$_{\pm.0091}$	.3175$_{\pm.0084}$	.1826$_{\pm.0032}$	.2232$_{\pm.0116}$	.1877$_{\pm.0012}$	.1825$_{\pm.0045}$	.2004$_{\pm.0054}$	*.1811$_{\pm.0057}$*	.2142$_{\pm.0272}$	**.1792$_{\pm.0034}$**
CHD49	.3123$_{\pm.1042}$	.2226$_{\pm.0324}$	.2213$_{\pm.0317}$	.2009$_{\pm.0083}$	.2204$_{\pm.0156}$	.2222$_{\pm.0330}$	*.2006$_{\pm.0229}$*	.2171$_{\pm.0278}$	.2818$_{\pm.0231}$	.2075$_{\pm.0278}$	**.1987$_{\pm.0174}$**
Emotion	.3807$_{\pm.1417}$	.1717$_{\pm.0054}$	.1624$_{\pm.0269}$	*.1426$_{\pm.0298}$*	.1694$_{\pm.0190}$	.1635$_{\pm.0231}$	.1556$_{\pm.0099}$	.1571$_{\pm.0153}$	.1880$_{\pm.0230}$	.1585$_{\pm.0205}$	**.1422$_{\pm.0182}$**
Flags	.2577$_{\pm.0598}$	.2454$_{\pm.0328}$	.2419$_{\pm.0551}$	.2375$_{\pm.0195}$	.2298$_{\pm.0347}$	.2324$_{\pm.0524}$	*.2232$_{\pm.0235}$*	.2370$_{\pm.0336}$	.2520$_{\pm.0092}$	.2239$_{\pm.0349}$	**.2165$_{\pm.0301}$**
Foodtruck	.2039$_{\pm.0619}$	.1549$_{\pm.0216}$	.1681$_{\pm.0091}$	.1556$_{\pm.0197}$	.1539$_{\pm.0220}$	.1593$_{\pm.0167}$	**.1501$_{\pm.0236}$**	.1533$_{\pm.0232}$	.2249$_{\pm.0244}$	*.1510$_{\pm.0248}$*	.1532$_{\pm.0213}$
Image	.4888$_{\pm.0200}$	.1744$_{\pm.0200}$	.1753$_{\pm.0130}$	*.1502$_{\pm.0149}$*	.1720$_{\pm.0093}$	.1749$_{\pm.0123}$	.1777$_{\pm.0070}$	.1755$_{\pm.0149}$	.1898$_{\pm.0163}$	.1794$_{\pm.0096}$	**.1493$_{\pm.0052}$**
GpositiveGO	.2826$_{\pm.1164}$	.0314$_{\pm.0160}$	.0360$_{\pm.0176}$	.0302$_{\pm.0121}$	*.0244$_{\pm.0117}$*	.0294$_{\pm.0136}$	.0255$_{\pm.0047}$	.0250$_{\pm.0056}$	.0523$_{\pm.0141}$	.0249$_{\pm.0071}$	**.0236$_{\pm.0091}$**
Scene	.4943$_{\pm.0307}$	.0747$_{\pm.0061}$	.0779$_{\pm.0068}$	**.0628$_{\pm.0077}$**	.1010$_{\pm.0039}$	.0953$_{\pm.0113}$	.0811$_{\pm.0132}$	.0978$_{\pm.0073}$	.0829$_{\pm.0057}$	.0813$_{\pm.0057}$	*.0710$_{\pm.0052}$*
VirusGO	.2673$_{\pm.1353}$	.0612$_{\pm.0288}$	.0555$_{\pm.0387}$	.0366$_{\pm.0118}$	.0412$_{\pm.0195}$	*.0275$_{\pm.0191}$*	**.0234$_{\pm.0098}$**	.0295$_{\pm.0119}$	.0960$_{\pm.0367}$	.0298$_{\pm.0137}$	.0301$_{\pm.0193}$
WaterQuality	.3958$_{\pm.0082}$	.2520$_{\pm.0126}$	.2498$_{\pm.0143}$	.2473$_{\pm.0122}$	.2680$_{\pm.0084}$	.2698$_{\pm.0110}$	.2811$_{\pm.0097}$	.2727$_{\pm.0095}$	*.2469$_{\pm.0087}$*	.2820$_{\pm.0092}$	**.2411$_{\pm.0090}$**
Yeast	.2797$_{\pm.0737}$	.1671$_{\pm.0117}$	.1677$_{\pm.0061}$	**.1646$_{\pm.0075}$**	.1732$_{\pm.0062}$	.1693$_{\pm.0087}$	.1703$_{\pm.0047}$	.1727$_{\pm.0101}$	.1790$_{\pm.0115}$	.1702$_{\pm.0076}$	*.1657$_{\pm.0065}$*
Mean rank	10.58	6.25	7.17	4.33	6.17	6.08	4.33	5.58	8.42	5.33	1.75

5) In LSTC and MASP, the loss value of the One-Error measure in the Birds dataset is around 0.25. This loss is a staggering 60% less than the third place. We speculate that this effect may be due to the unique network models employed by MASP and LSTC. Then this model is well suited to certain features of the Birds dataset.
6) By looking at the gap between the first and second place for each measure, LSTC has a clear advantage on F_1-score, macro-AUC, and One-Error. While for micro-AUC, NDCG, Hamming Loss and Ranking Loss the difference in effect between different datasets is slightly smaller.

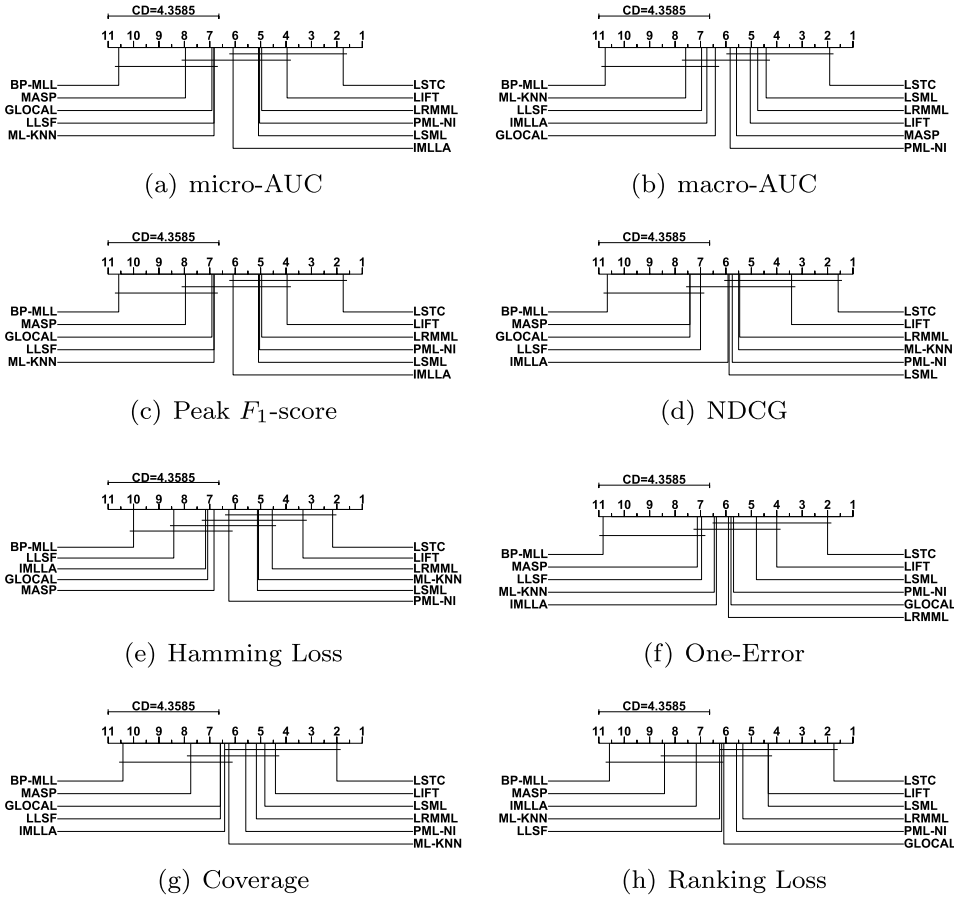


Fig. 9. Comparison of LSTC against ten comparing algorithms with the Bonferroni-Dunn test. CD = 3.4681 at significance level $\alpha = 0.05$.

7) LSTC ranks second across all evaluation measures on Scene datasets, while LIFT consistently ranks first. This reflects the superior performance of the feature embedding idea on this dataset. Meanwhile, it also reflects that the improvement based on LIFT in this paper is not sensitive to this dataset.

Fig. 9 illustrates the CD diagrams on each evaluation measure (# comparing algorithms $k = 11$, # dataset $N = 12$, $q_\alpha = 3.102$). We can observe that LSTC always maintains the best average rank. Meanwhile, in the Fig. 9(a), there is no evidence of a statistical difference among LSTC, LIFT, LRMML, PML-NI, LSML and IMLLA. However, they outperform the other five algorithms. Similarly, this performance also exists in other evaluation measures, but the algorithms that perform better are different.

4.5. Limitation

Although LSTC performs well on non-text-domain datasets, but faced with text-domain datasets, LSTC may no longer have a significant advantage according to Table 6. Because it is only to understand the limitations of LSTC on text-domain datasets, Table 6 only compares three of the eight evaluation measures.

The reasons are as follows.

1) The data matrix of the datasets on text-domain are generally sparse, LSTC does not perform well on some sparse datasets such as VirusGO in Tables 4 and 5.
2) Meanwhile, the label correlation of datasets on text-domain is not obvious. LSTC lacks to deal with the misjudgment of some independent labels caused by over-consideration of label correlation [36].
3) Considering label correlation through neural network may be prone to overfitting and getting stuck local optima. From Table 6, it can be found that the BP-MLL and MASP algorithms using neural network generally do not perform well.

At the same time, from Table 6, we also found that LSML and LIFT have excellent results when dealing with datasets on text-domain.

Table 6
The performance of LSTC on text-domain datasets under the three evaluation measures of micro-AUC, Peak F_1-score and NDCG. **Bold** indicates the best performance, and *italics* indicates the second best.

Dataset	micro-AUC										
	BP-MLL'06	ML-KNN'07	IMLLA'12	LIFT'14	LLSF'15	GLOCAL'18	LSML'19	PML-NI'21	MASP'22	LRMML'22	LSTC
Art	$.8263_{\pm.0062}$	$.8271_{\pm.0030}$	$.8017_{\pm.0023}$	$\mathbf{.8620_{\pm.0034}}$	$.7798_{\pm.0092}$	$.7859_{\pm.0071}$	$.8254_{\pm.0032}$	$.8362_{\pm.0058}$	$.7941_{\pm.0241}$	$\mathit{.8401_{\pm.0065}}$	$.8266_{\pm.0046}$
Business	$.9243_{\pm.0095}$	$\mathit{.9508_{\pm.0042}}$	$.9284_{\pm.0051}$	$\mathbf{.9598_{\pm.0039}}$	$.9109_{\pm.0055}$	$.9118_{\pm.0052}$	$.9341_{\pm.0056}$	$.9438_{\pm.0054}$	$.8797_{\pm.0186}$	$.9456_{\pm.0037}$	$.9376_{\pm.0075}$
Enron	$.8884_{\pm.0076}$	$.8994_{\pm.0024}$	$.8781_{\pm.0045}$	$\mathbf{.9137_{\pm.0084}}$	$.8345_{\pm.0080}$	$.7955_{\pm.0097}$	$.9024_{\pm.0079}$	$.8356_{\pm.0049}$	$.8181_{\pm.0101}$	$\mathit{.9083_{\pm.0044}}$	$.8907_{\pm.0030}$
Recreation	$.7795_{\pm.0366}$	$.7912_{\pm.0069}$	$.7813_{\pm.0081}$	$\mathbf{.8618_{\pm.0034}}$	$.7857_{\pm.0079}$	$.7927_{\pm.0097}$	$.8176_{\pm.0032}$	$.8351_{\pm.0082}$	$.7351_{\pm.0111}$	$\mathit{.8373_{\pm.0046}}$	$.7922_{\pm.0071}$
Social	$.9052_{\pm.0049}$	$\mathit{.9367_{\pm.0016}}$	$.8928_{\pm.0088}$	$\mathbf{.9437_{\pm.0035}}$	$.8725_{\pm.0070}$	$.8722_{\pm.0050}$	$.9051_{\pm.0059}$	$.9111_{\pm.0040}$	$.7608_{\pm.0100}$	$.9163_{\pm.0039}$	$.9065_{\pm.0069}$
Mean rank	7.20	3.80	7.80	1.00	9.40	9.00	5.40	4.40	10.40	2.40	5.20
	Peak F_1-score										
Art	$.3394_{\pm.0198}$	$.3374_{\pm.0075}$	$.3490_{\pm.0052}$	$.4476_{\pm.0183}$	$.3948_{\pm.0106}$	$.3965_{\pm.0145}$	$\mathit{.4609_{\pm.0036}}$	$.4600_{\pm.0097}$	$.3498_{\pm.0292}$	$\mathbf{.4657_{\pm.0111}}$	$.3603_{\pm.0108}$
Business	$.6353_{\pm.0585}$	$.7255_{\pm.0084}$	$.7231_{\pm.0056}$	$.7452_{\pm.0128}$	$.7153_{\pm.0096}$	$.7257_{\pm.0054}$	$.7459_{\pm.0085}$	$\mathit{.7461_{\pm.0099}}$	$.7093_{\pm.0114}$	$\mathbf{.7469_{\pm.0050}}$	$.7129_{\pm.0101}$
Enron	$.5149_{\pm.0215}$	$.5350_{\pm.0106}$	$.5575_{\pm.0082}$	$.6122_{\pm.0128}$	$.5119_{\pm.0161}$	$.4774_{\pm.0208}$	$\mathbf{.6225_{\pm.0170}}$	$.5273_{\pm.0150}$	$.5276_{\pm.0183}$	$\mathit{.6196_{\pm.0049}}$	$.5423_{\pm.0101}$
Recreation	$.2853_{\pm.0399}$	$.2929_{\pm.0102}$	$.3107_{\pm.0089}$	$.4766_{\pm.0051}$	$.3992_{\pm.0118}$	$.3990_{\pm.0134}$	$\mathbf{.4791_{\pm.0074}}$	$.4711_{\pm.0177}$	$.4020_{\pm.0175}$	$\mathit{.4777_{\pm.0083}}$	$.2989_{\pm.0096}$
Social	$.4329_{\pm.0173}$	$.6212_{\pm.0087}$	$.6199_{\pm.0198}$	$.6508_{\pm.0141}$	$.6096_{\pm.0211}$	$.6195_{\pm.0114}$	$\mathbf{.6525_{\pm.0115}}$	$.6468_{\pm.0055}$	$.5690_{\pm.0153}$	$\mathit{.6513_{\pm.0053}}$	$.5131_{\pm.0110}$
Mean rank	10.40	7.60	6.80	3.40	7.60	7.00	1.60	4.20	7.80	1.60	8.00
	NDCG										
Art	$.8099_{\pm.0203}$	$.8467_{\pm.0047}$	$.8458_{\pm.0066}$	$.8906_{\pm.0041}$	$.8622_{\pm.0047}$	$.8653_{\pm.0026}$	$\mathbf{.8912_{\pm.0027}}$	$.8893_{\pm.0050}$	$.8488_{\pm.0079}$	$\mathit{.8911_{\pm.0044}}$	$.8508_{\pm.0020}$
Business	$.9244_{\pm.0048}$	$.9585_{\pm.0026}$	$.9559_{\pm.0021}$	$\mathbf{.9650_{\pm.0006}}$	$.9474_{\pm.0035}$	$.9498_{\pm.0026}$	$.9641_{\pm.0013}$	$.9641_{\pm.0018}$	$.9400_{\pm.0046}$	$\mathit{.9643_{\pm.0013}}$	$.9560_{\pm.0031}$
Enron	$.8881_{\pm.0135}$	$.8992_{\pm.0058}$	$.9004_{\pm.0069}$	$.9126_{\pm.0029}$	$.8822_{\pm.0050}$	$.8654_{\pm.0109}$	$\mathbf{.9236_{\pm.0068}}$	$.8802_{\pm.0044}$	$.8815_{\pm.0078}$	$\mathit{.9235_{\pm.0027}}$	$.9032_{\pm.0015}$
Recreation	$.7882_{\pm.0351}$	$.8324_{\pm.0051}$	$.8369_{\pm.0044}$	$\mathbf{.8971_{\pm.0037}}$	$.8651_{\pm.0034}$	$.8685_{\pm.0056}$	$.8941_{\pm.0040}$	$.8922_{\pm.0050}$	$.8520_{\pm.0067}$	$\mathit{.8946_{\pm.0036}}$	$.8179_{\pm.0061}$
Social	$.8345_{\pm.0083}$	$.9346_{\pm.0020}$	$.9267_{\pm.0029}$	$\mathbf{.9415_{\pm.0042}}$	$.9121_{\pm.0049}$	$.9139_{\pm.0034}$	$\mathit{.9378_{\pm.0028}}$	$.9348_{\pm.0029}$	$.8780_{\pm.0048}$	$.9373_{\pm.0021}$	$.8995_{\pm.0093}$
Mean rank	10.20	6.80	7.20	1.80	7.40	7.20	2.20	5.20	8.80	2.20	7.20

LSML introduces a high-order label correlation control matrix **C** and manifold regularizer [2] to adjust the correlation, so that it can be stabilized within a reasonable range.

LIFT is the first algorithm that adopts the idea of label-specific features. Also, it is the source of the embedding idea of LSTC. But to reflect label correlation, we use neural network to train the model instead of LIFT using SVM classifier. As for why LIFT works well, on the one hand, the first-order multi-label algorithm obviously does not fall into overfitting due to label correlation. On the other hand, compared to neural network, SVM classifiers focus on global optima rather than local optima. However, the sparsity and weak label correlation of text-domain datasets are prone to getting stuck in local optima. Therefore, although LIFT is a first-order multi-label algorithm, it still outperforms most algorithms that consider label correlation on text-domain datasets.

At the same time, we realize that label correlation is unsuitable for all multi-label applications. In the field of personality psychology research, correlations between personality traits are low [23]. Almost all personality trait identification methods assume no dependencies among labels. Therefore, MLL algorithms that exploit label correlation like LSTC may not perform well on such datasets. These flaws may be a possible commonality among such MLL algorithms.

4.6. Discussion

We can now answer the questions proposed at the beginning of this section.

1) According to Fig. 4, the runtime of Algorithm 1 is in line with our time complexity analysis. In this way, the feature conversion is not the bottleneck of the LSTC algorithm. Meanwhile, Fig. 5 also verifies that the network training is not quite time-consuming.
2) According to Figs. 6, 8, and 7, all components of LSTC are appropriate.
3) According to Tables 4, 5 and Fig. 9, LSTC outperforms state-of-the-art algorithms on non-text-domain data. However, this does not indicate that LSTC is really superior than these algorithms. The reason is that LSTC only handles the basic MLL problem, while these algorithms may be able to handle missing data, missing labels, active learning, etc.
4) According to Table 6, LSTC is inappropriate for text-domain data. This is an important limitation.

5. Conclusions and further works

In this study, we proposed the LSTC algorithm with new feature conversion and label correlation learning techniques. The algorithm outperformed state-of-the-art algorithms on datasets from various domains other than text. Ablation studies also validated the superiority of our detailed techniques. Naturally, there are still some topics that require further investigation.

1) Adjustment for text-domain data. We have analyzed some possible reasons why LSTC is not suitable for this type of data. Consequently, we should adjust the algorithm, most likely the distance calculation strategy, to handle them more effectively.
2) Adaptation to missing labels, missing values, and active learning scenarios. This study is quite fundamental without considering these important issues in real world applications. Therefore, we will extend the algorithm to deal with these complicated situations.
3) Dealing with extreme multi-label learning (XML) [17]. XML is characterized by millions of instances, thousands to millions of features and labels, very high proportion (> 99.9%) of missing values, and very low (< 0.1%) label sparsity. It has attracted much research interests.[13] Hence we are also planning to enter this challenging and promising field.

CRediT authorship contribution statement

Xing-Yi Zhang: Methodology, Investigation, Software, Writing - original draft, Experimentation. **Fan Min:** Supervision, Methodology, Writing - review & editing. **Guojie Song:** Supervision, Writing - review & editing. **Hong Yu:** Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work is supported by Central Government Funds of Guiding Local Scientific and Technological Development (No. 2021ZYD0003), the National Natural Science Foundation of China (Nos. 62136002, 41674141), and Nanchong Municipal Government-Universities Scientific Cooperation Project (No. SXHZ045, 19XHZ0030).

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: VLDB, 1994, pp. 487–499.
[2] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[3] M.R. Boutell, J. Luo, X. Shen, C.M. Brown, Learning multi-label scene classification, Pattern Recognit. 37 (2004) 1757–1771.
[4] W.-C. Chang, H.-F. Yu, K. Zhong, Y. Yang, I.S. Dhillon, Taming pretrained transformers for extreme multi-label text classification, in: SIGKDD, 2020, pp. 3163–3171.
[5] Z. Cheng, Z. Zeng, Joint label-specific features and label correlation for multi-label learning with missing label, Appl. Intell. 50 (2020) 4029–4049.
[6] N. Chinchor, MUC-4 evaluation metrics, in: MUC, 1992.
[7] E. Gibaja, S. Ventura, A tutorial on multilabel learning, ACM Comput. Surv. 47 (2015) 1–38.
[8] H. Haripriya, C. Prathibhamol, Y.R. Pai, M.S. Sandeep, A.M. Sankar, S.N. Veerla, P. Nedungadi, Multi label prediction using association rule generation and simple k-means, in: ICCTICT, 2016, pp. 159–163.
[9] H. Huang, H. Xu, X. Wang, W. Silamu, Maximum f1-score discriminative training criterion for automatic mispronunciation detection, IEEE/ACM Trans. Audio Speech Lang. Process. 23 (2015) 787–797.
[10] J. Huang, F. Qin, X. Zheng, Z.-K. Cheng, Z.-X. Yuan, W.-G. Zhang, Q.-M. Huang, Improving multi-label classification with missing labels by learning label-specific features, Inf. Sci. 492 (2019) 124–146.
[11] S.-J. Huang, Z.-H. Zhou, Multi-label learning by exploiting label correlations locally, in: AAAI, 2012, pp. 949–955.
[12] X.-Y. Jia, S.-S. Zhu, W.-W. Li, Joint label-specific features and correlation information for multi-label learning, J. Comput. Sci. Technol. 35 (2020) 247–258.
[13] H. Jun, G. Li, Q. Huang, X. Wu, Learning label specific features for multi-label classification, in: ICDM, 2015, pp. 181–190.
[14] J. Kekäläinen, K. Järvelin, Using graded relevance assessments in IR evaluation, J. Am. Soc. Inf. Sci. Technol. 53 (2002) 1120–1129.
[15] S. Kumar, R. Rastogi, Low rank label subspace transformation for multi-label learning with missing labels, Inf. Sci. 596 (2022) 53–72.
[16] Y. Li, Y. Song, J. Luo, Improving pairwise ranking for multi-label image classification, in: CVPR, 2017, pp. 3617–3625.
[17] J. Liu, W.-C. Chang, Y. Wu, Y. Yang, Deep learning for extreme multi-label text classification, in: SIGIR, 2017, pp. 115–124.
[18] Y. Liu, K. Wen, Q. Gao, X. Gao, F. Nie, SVM based multi-label learning with missing labels for image annotation, Pattern Recognit. 78 (2018) 307–317.
[19] Z. Ma, S. Chen, Expand globally, shrink locally: discriminant multi-label learning with missing labels, Pattern Recognit. 111 (2021) 107675.
[20] S.J. Mason, N.E. Graham, Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: statistical significance and interpretation, Q. J. R. Meteorol. Soc. 128 (2002) 2145–2166.
[21] A.K. McCallum, Multi-label text classification with a mixture model trained by EM, in: AAAI Workshop on Text Learning, 1999.
[22] X.-Y. Min, K. Qian, B.-W. Zhang, G. Song, F. Min, Multi-label active learning through serial–parallel neural networks, Knowl.-Based Syst. 251 (2022) 109226.
[23] N.K. Mishra, A. Singh, P.K. Singh, Multi-label personality trait identification from text, Multimed. Tools Appl. 81 (2022) 1–17.
[24] N.K. Mishra, P.K. Singh, Feature construction and smote-based imbalance handling for multi-label learning, Inf. Sci. 563 (2021) 342–357.
[25] J.M. Moyano, E.L. Gibaja, K.J. Cios, S. Ventura, Review of ensembles of multi-label classifiers: models, experimental study and prospects, Inf. Fusion 44 (2018) 33–45.
[26] J. Nam, J. Kim, E.L. Mencía, I. Gurevych, J. Fürnkranz, Large-scale multi-label text classification revisiting neural networks, in: ECML, 2014, pp. 437–452.

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

[27] G. Nasierding, G. Tsoumakas, A.Z. Kouzani, Clustering based multi-label classification for image annotation and retrieval, in: IEEE SMC, 2009, pp. 4514–4519.
[28] T.T. Nguyen, M.T. Dang, A.V. Luong, A.W.-C. Liew, T. Liang, J. McCall, Multi-label classification via incremental clustering on an evolving data stream, Pattern Recognit. 95 (2019) 96–113.
[29] F. Provost, R. Kohavi, Guest editors' introduction: on applied research in machine learning, Mach. Learn. 30 (1998) 127–132.
[30] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, T. Mei, H.-J. Zhang, Correlative multi-label video annotation, in: ACM MM, 2007, pp. 17–26.
[31] K. Qian, X.-Y. Min, Y. Cheng, F. Min, Weight matrix sharing for multi-label learning, Pattern Recognit. 136 (2023) 109156.
[32] W. Qian, J. Huang, Y. Wang, W. Shu, Mutual information-based label distribution feature selection for multi-label learning, Knowl.-Based Syst. 195 (2020) 105684.
[33] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492–1496.
[34] S. Shu, F. Lv, Y. Yan, L. Li, S. He, J. He, Incorporating multiple cluster centers for multi-label learning, Inf. Sci. 590 (2022) 60–73.
[35] K.A. Spackman, Signal detection theory: valuable tools for evaluating inductive learning, in: IWML, 1989, pp. 160–163.
[36] F. Sun, J. Tang, H. Li, G.-J. Qi, T.S. Huang, Multi-label image categorization with sparse factor representation, IEEE Trans. Image Process. 23 (2014) 1028–1037.
[37] L. Sun, T. Yin, W. Ding, Y. Qian, J. Xu, Feature selection with missing labels using multilabel fuzzy neighborhood rough sets and maximum relevance minimum redundancy, IEEE Trans. Fuzzy Syst. 30 (2021) 1197–1211.
[38] A. Tan, J. Liang, W.-Z. Wu, J. Zhang, Semi-supervised partial multi-label classification via consistency learning, Pattern Recognit. 131 (2022) 108839.
[39] G. Tsoumakas, I. Katakis, I. Vlahavas, Random k-labelsets for multilabel classification, IEEE Trans. Knowl. Data Eng. 23 (2010) 1079–1089.
[40] M. Wang, F. Min, Z.-H. Zhang, Y.-X. Wu, Active learning through density clustering, Expert Syst. Appl. 85 (2017) 305–317.
[41] R. Wang, S. Kwong, X. Wang, Y. Jia, Active k-labelsets ensemble for multi-label classification, Pattern Recognit. 109 (2021) 107583.
[42] M.-K. Xie, S.-J. Huang, Partial multi-label learning with noisy label identification, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2022) 1–12.
[43] S. Xu, X. Yang, H. Yu, D.-J. Yu, J. Yang, E.C. Tsang, Multi-label learning with label-specific feature reduction, Knowl.-Based Syst. 104 (2016) 52–61.
[44] M.-L. Zhang, An improved multi-label lazy learning approach, J. Comput. Res. Dev. 49 (2012) 2271–2282.
[45] M.-L. Zhang, L. Wu, LIFT: multi-label learning with label-specific features, IEEE Trans. Pattern Anal. Mach. Intell. 37 (2014) 107–120.
[46] M.-L. Zhang, Z.-H. Zhou, Multi-label neural networks with applications to functional genomics and text categorization, IEEE Trans. Knowl. Data Eng. 18 (2006) 1338–1351.
[47] M.-L. Zhang, Z.-H. Zhou, ML-KNN: a lazy learning approach to multi-label learning, Pattern Recognit. 40 (2007) 2038–2048.
[48] Y. Zhu, J.T. Kwok, Z.-H. Zhou, Multi-label learning with global and local label correlation, IEEE Trans. Knowl. Data Eng. 30 (2018) 1081–1094.