Noname manuscript No.
(will be inserted by the editor)

LSGM: Label Sequence Generation Model for Multi-Label Learning

Jin Huang[1] · **Hanyu Yu**[1] · **Jia Zhu**[1] · **Yong Tang**[1] · **Pu Dong**[1] · **Jiaqi Lun**[1]

Received: date / Accepted: date

Abstract Multi-label learning tackles the problem where each instance object is represented by a *single* feature vector and associated with a set of class labels simultaneously. For unseen example, some existing approaches generally choose to determine all possible class labels at once based on the single feature vector. However, it can enrich the input space information by considering several nearest neighbor examples similar to the unseen example. In addition, labels that are marked differently should have different label attributes. Based on the above reflections and inspired by the Sequence-To-Sequence model, we convert the multi-label learning task to the seq2seq task and propose a **L**abel **S**equence **G**eneration neural network **M**odel called **LSGM** in this paper. Briefly, the LSGM model with Encoder-Decoder structure encodes the constructed chain-type feature vector sequence and then decoded predicts the label sub-sequences having different label attributes. The experimental results and ablation studies show that the proposed model is reasonable and effective.

Keywords Multi-label learning · Sequence-To-Sequence · Label attributes · Encoder-Decoder

1 Introduction

Multi-label learning deals with instance objects having *multiple* labels simultaneously, which widely exist in real-world applications [2,5]. In detail, for the instance objects set **I**, denote $\mathbf{F} = \{f_1, f_2, ..., f_m\}$ be the m-dimensional

F. Jin Huang
E-mail: huangjin@m.scnu.edu.cn

Hanyu Yu(🖂)
E-mail: yuhanyu@m.scnu.edu.cn

[1] School of Computer, South China Normal University, Guangzhou, China

input feature space and $\mathbf{L} = \{l_1, l_2, ..., l_n\}$ be the finite label set of n possible labels. If unseen instance object has the j-th labels, the $l_j \in \mathbf{L}$ would be marked as 1, otherwise 0. Then, the task of multi-label learning (or *multi-label classification*) is to learn a mapping function $\mathbf{H} : \mathbf{F} \rightarrow 2^{\mathbf{L}}$ which means to predict a proper label set $\mathbf{L}_i$($i.e.$ $\mathbf{H}(\mathbf{F}_i)) \in \mathbf{L}$ for the instance object $\mathbf{I}_i \in \mathbf{I}$ by the feature vector $\mathbf{F}_i \in \mathbf{F}$.

During the past decades, a considerable number of approaches have been proposed to learn from multi-label data [20]. The common strategy adopted by existing approaches can be divided into two types from the comprehensive perspective of data and algorithms. One is the *problem transformation methods*, another one is the *algorithm adaptation methods* [31]. The former tackle multi-label learning problem by transforming it into other well-established learning scenarios. The latter tackle multi-label learning problem by adapting popular learning techniques to deal with multi-label data directly. Some multi-label learning methods [2,3,18], whatever are the problem transformation methods or the problem transformation methods, generally use the single input feature vector to predict the whole label set at once. However, in the algorithm adaptation methods, the ML-KNN methods [30] especially considers multiple instance objects to enrich the input space information and then uses the maximum a posterior probability to determine the set of labels for unseen instance objects. This method illustrates that increasing the number of input feature information in multi-label learning may be helpful to label prediction. And, the EncDec model [16] converts the multi-label classification to a sequence-to-sequence task and use recurrent neural networks to predict only positive labels(marked as 1). It tells us that the labels have positive or negative attributes, which maybe helpful to predict labels. Besides, as the increasing of feature vector dimension, the number of labels and instance objects, some methods [10,15,27,29] choose to use the neural network to tackle the large-scale multi-label learning task.

Based on the above analysis and inspired by the tremendous success of the Sequence-To-Sequence model [1,14] in the machine translation in recent years, we propose a **L**abel **S**equence **G**eneration neural network **M**odel, called **LSGM**, that convert the multi-label learning task to the seq2seq task of multi-label learning. The LSGM model utilizes the Encoder-Decoder structure. We construct a feature chain encoder which firstly get k nearest neighbor instance objects similar to the unseen instance object from the constructed feature vectors graph. Next, the feature vectors of k instance objects and unseen instance object are arranged into a chain-type feature vectors sequence using certain rule. And then the encoder encodes the entire feature chain into a representation vector and transmits it to the label sequence decoder. During decoding, the decoder first splits the original label set to two label sub-sequences according to the label attributes and then predicts separately. Next, using the ensemble methods to merge the predicted results of two sub-sequences and get the predicted label set for the unseen instance object lastly.

The contributions of this paper are summarized as follows:

- We propose to convert the multi-label learning task to the seq2seq task of multi-label learning and describe the formalization conversion process.
- A kind of special sorting rule is used to arrange the nearest neighbor instance objects and unseen instance object into the chain-type feature vectors sequence so that can capture the potential meaning of these instance objects effectively.
- During decoding, the global label information is proposed to alleviate the exposure bias problem and the ensemble methods are proposed to tackle the label conflict problem due to different label attributes.

The sorting rule, global label information, and ensemble methods would be validated and compared in the experiments and ablation study sections.

The rest of this paper is organized as follows. Section 2 briefly discusses existing approaches to multi-label learning. Section 3 presents the details of the proposed model LSGM. Then, section 4 reports the experiment results. Section 5 gives the ablation studies. Finally, Section 6 concludes and indicates several issues for future work.

2 Related Work

During the past decades, a lot of approaches have been proposed to learn from multi-label data. These approaches could be simply divided into two categorizations [31] from the comprehensive perspective of data and algorithms:

- *Problem transformation approaches*: This category of approaches tackles multi-label learning problem by transforming it into other well-established learning scenarios. The key factor of these approaches is to fit data to approaches.
- *Algorithm adaptation approaches*: This category of approaches tackles multi-label learning problem by adapting popular learning techniques to deal with multi-label data directly. The key factor of these approaches is to fit approaches to data.

In the former category approaches, the Binary Relevance [2] decomposes the task of multi-label learning into n independent binary classification problems, where each binary classification problem corresponds to a possible label in the label set $\mathbf{L}$. Classifier Chains [18] transforms the multi-label learning problem into a *chain* of binary classification problems, where subsequent binary classifiers in the chain is built upon the predictions of preceding ones. The Calibrated Label Ranking [7] transforms the multi-label learning problem into the label ranking problem, where ranking among labels is fulfilled by the techniques of pairwise comparison. The Binary Relevance and Classifier Chains approaches both use the single input feature vector to predict the possible label set. In the latter category approaches, the Ranking Support Vector Machine(Rank-SVM) [5] adapts maximum margin strategy to deal with multi-label learning data, where a set of linear classifiers are optimized to minimize the empirical ranking loss and enabled to handle nonlinear case with kernel tricks. And the collective

Multi-Label Classifier(CML) [8] adapts maximum entropy principle to deal with multi-label data, where correlations among labels are encoded as constraints that the resulting distribution must satisfy. The above two approaches also use the single input feature vector to predict the possible label set. Especially, the ML-KNN [30] uses multiple instance objects to enrich the input feature space and then adapts *k-nearest neighbor* techniques to deal with multi-label learning data, where maximum a posteriori rule is utilized to make prediction by reasoning with the labeling information embodied in the neighbors. This approach illustrates that increasing the number of input features in multi-label learning is helpful to label prediction.

Recent years, as the tremendous success of the Sequence-To-Sequence model [1, 14] in the machine translation in recent years, the EncDec model [16] converts the multi-label classification to a sequence-to-sequence task and use recurrent neural networks to predict positive labels for unseen instance objects. Reducing the length of the label chain might be helpful in reducing the label cascading errors, particularly at the end of the label chain (i.e. label sequence) [16]. It tells us that the positive or negative attributes of labels and shorter label sequence may be helpful to predict labels. And as the increasing of feature vector dimension, the number of labels and instance objects, some methods [29, 27, 10, 15] choose to use the neural network to tackle the large-scale multi-label learning task, where the BP-MLL [29] employs a novel error function by Back-propagation algorithm which can capture the characteristics of multi-label learning, i.e. the labels belonging to an instance object should be ranked higher than those not belonging to that instance object.

From another perspective, the task of multi-label learning is rather difficult due to the exponential number of possible label sets(i.e. $2^{\mathbf{L}}$, in **Introduction**) and only a single input feature vector(i.e. $\mathbf{F}_i$). To cope with this challenge, existing approaches also focus on exploiting the correlation between labels to tackle the task, which can be divided into three categories based on the order of label correlations, namely the *first-order* approaches, *second-order* approaches and *high-order* approaches [31]:

- The first-order approaches deal with multi-label learning problem by decomposing it into a number of independent binary classification problems, such as [2] and [30]. It may be less effective due to ignoring the label correlations.
- Second-order approaches consider the pairwise correlation between labels, such as [29] and [7]. But it may suffer from the fact that label correlations could go beyond second-order under real-world scenarios.
- The High-order approaches especially consider high-order relations between labels, such as [22] and [11]. That could address more realistic label correlations, while may exhibit high model complexities.

Thus, how to effectively use the label correlation is one of the key factors to make the model predict better.

3 LSGM

We introduce LSGM and its detailed implementation in this section. There are two major modules in our framework: *feature chain encoder* and *label sequence decoder*. During encoding, the model firstly constructs the chain-type feature vectors sequence on the features vectors graph and then encodes. The encoder encodes the entire feature chain into a representation vector and then transmits it to the decoder. For decoding, the LSGM model first splits the source label set to two label sub-sequences with different attributes and then decodes. Lastly, using the ensemble methods to merge the decoded results of two sub-sequences. The whole framework is described in Figure 1.

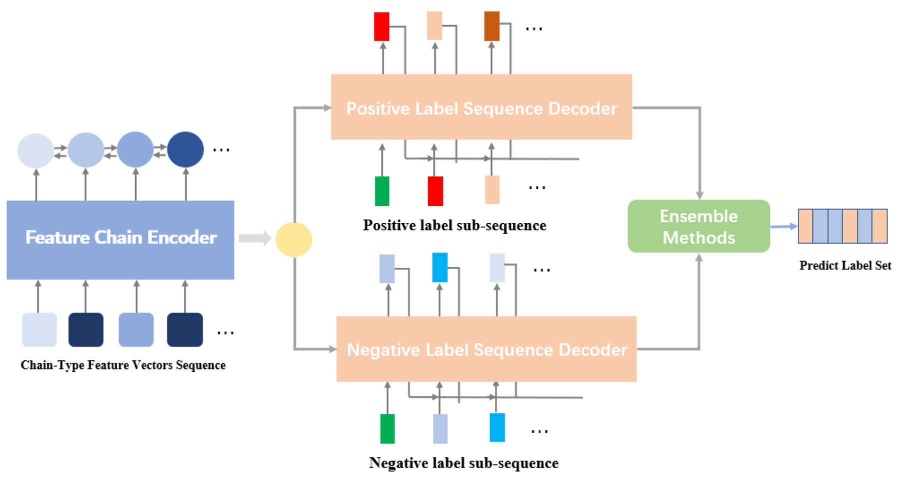

Positive Label Sequence Decoder
Feature Chain Encoder
Positive label sub-sequence
Ensemble Methods
Predict Label Set
Negative Label Sequence Decoder
Chain-Type Feature Vectors Sequence
Negative label sub-sequence

Fig. 1 The left blue blocks are the input feature vectors and blue circles are the hidden states. The right red or blue blocks represent different positive or negative labels.

3.1 Convert Task

As we describe in the **Introduction** section, the task of multi-label learning is to learn a mapping function $\mathbf{H}$, which use the feature vector $\mathbf{F}_i \in \mathbb{R}^m$ to predict a proper label set $\mathbf{L}_i \in \mathbb{R}^n$ for the unseen instance object $\mathbf{I}_i$. In the original multi-label learning task, the input space is only a single feature vector $\mathbf{F}_i$ for the instance object $\mathbf{I}_i$. However, the ML-KNN [30] considers multiple instance objects to enrich the input space information. Base on that, we can get some instance object rather than one instances object. Next, by arranging these instance objects and the unseen instance object with certain rules, we can get the *chain-type feature vectors sequence*. In the finite label set $\mathbf{L} = \{l_1, l_2, ..., l_n\}$, if the unseen instance object $\mathbf{I}_i$ has the label l_j, the mark of l_j is 1, otherwise 0. That means the labels have *positive*(marked as

1) or *negative*(marked as 0) attributes which can be used to predict labels. Especially, the label attributes had been applied in the EncDec model [16]. Thus we can get two label sub-sequences from the original label set in multi-label learning. One is the *positive label attributes label sub-sequence*, and another one is the *negative label attributes label sub-sequence*. The EncDec model only uses positive labels in multi-label classification data, because of having fewer labels to decode. However, after observing the multi-label learning data, we found that the number of positive labels is not always less than the number of negative labels. Moreover, in the experiments, we found that merge the decoded results of two label sub-sequences can be better than any single label sub-sequence decoded result.

According to the previous analysis, we can get the input feature vectors sequence and two target label sub-sequences. And the basic thought of seq2seq task is to map(encode) the input sequence to a vector of fixed dimensionality, and the decode the target sequence from the vector [19]. Based on that, we convert the multi-label learning task to the seq2seq task of multi-label learning. A visualization of the conversion process can be seen in Figure 2. We use the input feature vectors sequence to predict the two target label sub-sequences. The seq2seq task of multi-label learning can be converted and formalized as the following:

$$\begin{aligned}\mathbf{H}:\mathbf{F}_i\to\mathbf{L}_i,\ \ for\ \mathbf{I}_i &\Rightarrow \{\mathbf{F}_i,\mathbf{F}_1,\mathbf{F}_2,...\}\to\mathbf{L}_i\\ &\Rightarrow \mathbf{F}_{chain}\to\mathbf{L}_i\\ &\Rightarrow p(\mathbf{L}_i|\mathbf{F}_{chain})\\ &=\prod P(l_j/l_1,l_2,...,l_{j-1}|\mathbf{F}_{chain}).\end{aligned}$$

where $l_j\in\mathbf{L}_i$. More precisely, we use the input chain-type feature vectors sequence to predict the target positive label sub-sequence and the target negative label sub-sequence, and then merge the decoded results of two sub-sequences. Lastly, we can get the predicted label set $\mathbf{L}_i$ for the unseen instance object $\mathbf{I}_i$.

3.2 Feature Chain Encoder

Similar to Zhang's work [30], for each unseen instance object $\mathbf{I}_i$, its k nearest neighbors in the training set are firstly found. However, unlike Zhang's [30] work that directly uses the k-Nearest Neighbor (kNN) algorithm to find k nearest neighbor instance objects, we construct a weighted instance object feature graph on the training instance objects, which is like [13] and [4]. Thus we can high effectively search k nearest neighbor instance objects from this graph for the unseen instance object $\mathbf{I}_i$. Then we arrange the k nearest neighbor instance objects and unseen instance object into a chain-type order using certain rule. Next, the encoder would encode this chain-type input sequence. The whole process would be described in the following steps.

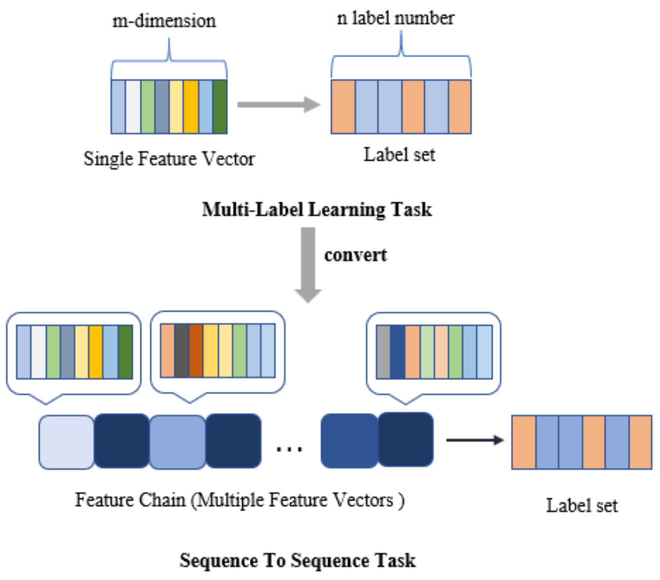

m-dimension
n label number
Single Feature Vector
Label set
Multi-Label Learning Task
convert
Feature Chain (Multiple Feature Vectors)
Label set
Sequence To Sequence Task

Fig. 2 Convert the multi-label learning task to the seq2seq task of multi-label learning.

Step-1 Instance Object Feature Graph: Let $\mathbf{G}$ be a weighted neighborhood graph on the training instance objects. Each vertex in $\mathbf{G}$ corresponds an instance object $\mathbf{I}_i$. And the edge $\mathbf{E}_{ij}$ between vertex $\mathbf{I}_i$ and vertex $\mathbf{I}_j$ means, $\mathbf{I}_i$ is the nearest neighbor of $\mathbf{I}_j$ or $\mathbf{I}_j$ is the nearest neighbor of $\mathbf{I}_i$. Using the *Euclidean Distance* [24] to calculate the weight of edge $\mathbf{E}_{ij}$. That is:

$$d(\mathbf{I}_i, \mathbf{I}_j) = d(\mathbf{I}_j, \mathbf{I}_i) = \sqrt{(\mathbf{F}_i - \mathbf{F}_j)^2} = \sqrt{\sum_{t=1}^{m} (f_t^i - f_t^j)^2}.$$

That means the weight of edge $\mathbf{E}_{ij}$ is equal the distance between feature vector $\mathbf{F}_i$ and $\mathbf{F}_j$. This is the *feature vector graph* $\mathbf{G}$ on training instance objects. We define a spare matrix $\mathbf{S}$[13] from it, indicating the similarities among neighboring instance objects:

$$\mathbf{S}_{ij} = \begin{cases} \frac{1}{J_i} exp(\frac{d(\mathbf{I}_i - \mathbf{I}_j)^2}{2\sigma^2}), & if\ \mathbf{I}_j \in \mathbf{K}_i \\ 0, & otherwise. \end{cases}$$

where $\mathbf{K}_i$ is the k nearest neighbor instance objects set of instance object $\mathbf{I}_i$. And parameter σ is empirically estimated as the average distance between instance objects. $J_i = \sum_{\mathbf{I}_j \in \mathbf{K}_i} exp(-\frac{d(\mathbf{I}_i - \mathbf{I}_j)^2}{2\sigma^2})$, thus $\sum_j \mathbf{S}_{ij} = 1$ for instance objects $\mathbf{I}_i$. In order to reduce computational cost of searching k nearest neighbor for unseen instance object $\mathbf{I}_i$, we use Ball-tree [17] to efficiently search k nearest neighbor instance objects for the unseen instance object $\mathbf{I}_i$.

Step-2 Chain-Type Feature Vector Sequence: We can get k nearest neighbor instance objects similar to the unseen instance object $\mathbf{I}_i$ from the **Step-1**. Next, we sort these instance objects by their correlation with the unseen instance object $\mathbf{I}_i$. In detail, we sort these instance in ascending order. This sort of rule is the correlation with the unseen instance object $\mathbf{I}_i$. The more similar with the unseen instance object $\mathbf{I}_i$, the more behind the k-th nearest neighbor instance object sort. Thus, we can get a k nearest neighbor instance objects sorted sequence $\{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_k\} \in \mathbb{R}^k$. And then we insert the unseen instance object $\mathbf{I}_i$ into this sequence to generate a mix instance objects sequence $\{\mathbf{I}_1, \mathbf{I}_i, \mathbf{I}_2, \mathbf{I}_i, ..., \mathbf{I}_k, \mathbf{I}_i\} \in \mathbb{R}^{k+k}$. The reasons why we sort these instance objects and unseen instance object are described in the Ablation Study section. Next, the mix instance objects sequence is converted to the corresponding mix instance objects feature vectors sequence $\{\mathbf{F}_1, \mathbf{F}_i, \mathbf{F}_2, \mathbf{F}_i, ..., \mathbf{F}_k, \mathbf{F}_i\} \in \mathbb{R}^{(k+k)\times m}$. This is the *Chain-Type Feature Vectors Sequence*. We use a bidirectional LSTM neural network [9] to build the **Feature Chain Encoder**, namely **FC-Encoder**, which can better capture the potential meaning of these instances objects from forward and back. It reads the mixed instance object feature vectors sequence and computes the hidden states for each instance objects. The two directions hidden states of the encoder at t-th moment is,

$$\overrightarrow{\mathbf{h}_t} = \overrightarrow{\textbf{FC-Encoder}}(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{F}_t)$$

$$\overleftarrow{\mathbf{h}_t} = \overleftarrow{\textbf{FC-Encoder}}(\overleftarrow{\mathbf{h}}_{t+1}, \mathbf{F}_t)$$

We obtain the final hidden state vector representation of the t-th instance object by concatenating the hidden states from both directions, $\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$, which embodies the information of the sequence centered around the t-th instance object. Thus, the **FC-Encoder** can get the potential meaning of these instance objects from this chain-type feature vectors sequence.

3.3 Label Sequence Decoder

The Bi-LSTM of encoder output a hidden state $\mathbf{h}_{k+k}$ at the last moment, which represents all potential meanings of the chain-type feature vectors sequence, marked as $\mathbf{C} \in \mathbb{R}^{1\times m}$. Then the representation vector $\mathbf{C}$ is passed to the decode. Next, we would introduce the normal decode process. In this process, we put forward to use *Global Label Information* to actively alleviate the exposure bias problem and thus introduce high-order label correlation. Then, we raise to use ensemble methods to merge the decoded results of two label sub-sequences. All processes would be described in the following steps.

Step-1 Normal Decode Label Sequence: As described in the **Convert Task** section, we can get two label sub-sequences with different label attributes from the original label set. One is the positive label sub-sequence, another one is the negative label sub-sequence. Two sub-sequences are the target sequences for decoding in the seq2seq task of multi-label learning. Thus, there are two decoders corresponding to the two label sub-sequences. One is the positive label sub-sequence decoder **LS-Decoder**$_P$, another one is the negative label sub-sequence decoder **LS-Decoder**$_N$. The decoders would generate current labels according to the representation vector **C** and the previously predicted label. Unidirectional LSTM [9] is used to decode the two target label sub-sequences. The hidden states of two decoders at t-th moment are,

$$\begin{aligned}\mathbf{s}_t^p &= \textbf{LS-Decoder}_P(\mathbf{s}_{t-1}^p, l_{t-1}^p, \mathbf{C}, \mathbf{h}_{all}),\\ \mathbf{s}_t^n &= \textbf{LS-Decoder}_N(\mathbf{s}_{t-1}^n, l_{t-1}^n, \mathbf{C}, \mathbf{h}_{all}).\end{aligned}$$

where l_{t-1}^p and l_{t-1}^n is the previously predicted positive labels and negative label. Besides the $\mathbf{h}_{all} \in \mathbb{R}^{(k+k)\times m}$ is all the hidden states of the Bi-LSTM of encoder, which is used to do attention mechanism [14].

Step-2 Global Label Information: In the above **Step-1**, the prediction of the current label is based on the last predicted label. In other words, the current label depends on the last label greedily. Therefore, it is likely that we would get a succession of wrong label predictions in the following time steps if the prediction is wrong at time-step t [25]. This is an inherent attribute of the auto-regressive nature of the seq2seq task, which is known as the *exposure bias* [26]. The *Beam Search* [6] technology is used to passively alleviate this problem in the testing infer phase of the decoder. But in our work, we propose to actively alleviate this exposure bias problem during the training phase of the decoder by superimposing labels that had been predicted. In this case, the decoder can perceive more labels information, instead of only one single label. Moreover, it can introduce the high-order label correlation by superimposing the already predicted labels, which make the decoder learn more label correlations. We call that is the *Global Label Information* which would help the decoder reduce the dependence on the last label and can capture more label correlations. After adding the global label information, the hidden state of the decoders in the t-th moment is,

$$\begin{aligned}\mathbf{s}_t^p &= \textbf{LS-Decoder}_P(\mathbf{s}_{t-1}^p, (l_1^p + l_2^p + ... + l_{t-1}^p), \mathbf{C}, \mathbf{h}_{all})\\ &= \textbf{LS-Decoder}_P(\mathbf{s}_{t-1}^p, \sum_{v=1}^{t-1} l_v^p, \mathbf{C}, \mathbf{h}_{all})\\ \mathbf{s}_t^n &= \textbf{LS-Decoder}_N(\mathbf{s}_{t-1}^n, (l_1^n + l_2^n + ... + l_{t-1}^n), \mathbf{C}, \mathbf{h}_{all})\\ &= \textbf{LS-Decoder}_N(\mathbf{s}_{t-1}^n, \sum_{v=1}^{t-1} l_v^n, \mathbf{C}, \mathbf{h}_{all})\end{aligned}$$

where the $\sum_{v=1}^{t-1} l_v^p$ and $\sum_{v=1}^{t-1} l_v^n$ means to superimpose all predicted labels before t-1 moment(includes the t-1 moment). The comparison of the LSGM model with or without *Global Label Information* are shown in the Ablation Study section.

Step-3 Merge Sub-Sequence Decoder Results: In the above **Step-1** and **Step-2**, the original label set is split into two label sub-sequences with different label attributes. Thus there are two decoding target sub-sequences and corresponding two decoding results. In multi-label classification data, the EncDec model [16] merely utilizes positive labels as target decoding sequence, which is less than the number of negative labels. Because it is easier for the EncDec model to learn the correlation between both sequences(i.e. the input sequence and the target sequence). However, in the multi-label learning data, the number of negative labels is not always less than the number of positive labels (That would be explained in the Experiments and Ablation Study). Especially, the attribute of positive labels sub-sequence is just opposite to the attribute of negative label sub-sequences. We can use this attribute to ensemble the decoded results of the decoder **LS-Decoder**$_P$ and **LS-Decoder**$_N$.

The hidden state $\mathbf{s}_t^p \in \mathbb{R}^m$ and $\mathbf{s}_t^n \in \mathbb{R}^m$ of the two decoders contain the potential label predicted results. Next, leveraging the *softmax function*, the hidden states $\mathbf{s}_t^p$ and $\mathbf{s}_t^n$ are converted to the conditional probability sets of the original label set at the t-th moment. That is

$$\mathbf{P}_t^p = softmax(\mathbf{W}_s^p \mathbf{s}_t^p + \mathbf{b}_s^p)$$

$$\mathbf{P}_t^n = softmax(\mathbf{W}_s^n \mathbf{s}_t^n + \mathbf{b}_s^n)$$

where $\mathbf{W}_s^p \in \mathbb{R}^{m\times n}$ and $\mathbf{W}_s^n \in \mathbb{R}^{m\times n}$ are the weight matrix, and $\mathbf{b}_s^p \in \mathbf{R}^{m\times 1}$ and $\mathbf{b}_s^n \in \mathbf{R}^{m\times 1}$ are the bias vector. The serial numbers of the maximum probability in the label probability set $\mathbf{P}_t^p$ and $\mathbf{P}_t^n$ are the serial numbers of prediction label namely. That is,

$$P_i^p(l_i/l_1, l_2, ..., l_{i-1}|\mathbf{F}_{chain}) = max(\mathbf{P}_t^p)$$

$$P_j^n(l_j/l_1, l_2, ..., l_{j-1}|\mathbf{F}_{chain}) = max(\mathbf{P}_t^n)$$

The conditional probability P_i^p is the probability that the decoder **LS-Decoder**$_p$ predicts that the i-th label of label set $\mathbf{L}$ is a positive label. Similarly, the conditional probability P_j^n is the probability that the decoder **LS-Decoder**$_n$ predicts that the j-th label of label set $\mathbf{L}$ is a negative label. We call the conditional probability P_i^p and P_i^n is the *generation probability.* Because it represents the possibility of the decoder generates the i-th (or j-th) label from the input chain-type feature vectors sequence. In general, some models having seq2seq task directly output the corresponding token, namely the label in the seq2seq task of multi-label learning. However, the i-th label and j-th label may denote the same label in some cases. It means, a label is predicted as a positive label by decoder **LS-Decoder**$_p$ and at the same time, this label is also predicted as a negative label by decoder **LS-Decoder**$_n$. This is a label prediction conflict caused by two decoders with different label attributes. Thus, we should use certain methods to deal with that.

Suppose there are six labels $\{l_1, l_2, l_3, l_4, l_5, l_6\}$ of the unseen instance object $\mathbf{I}_i$, which are need to be predicted as positive(0) or negative(0) label. Assume the positive label sub-sequence decoder **LS-Decoder**$_p$ predicts that the label $\{l_2, l_3, l_5\}$ are positive labels, and the negative label sub-sequence decoder

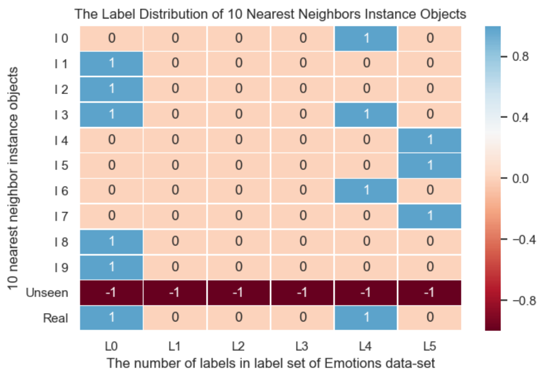
LS-Decoder$_n$ predicts that the label $\{l_1, l_2, l_4\}$ are negative labels. Then, there would be four cases,

$$\{l_1, l_2, l_3, l_4, l_5, l_6\} = \begin{cases} l_1, l_4 : -; \\ \quad l_2 : *; \\ l_3, l_5 : +; \\ \quad l_6 : null. \end{cases}$$

where the symbol "+"("−") represents that these labels are predicted as positive(negative) labels, and the symbol "$*$" means that both cases(positives label and negative labels) exist at the same time. It is contradictory and needs to be resolved. The symbol "*null*" indicates the current labels are not predicted by any decoders. That is also a contradictory and needs to be tackled.

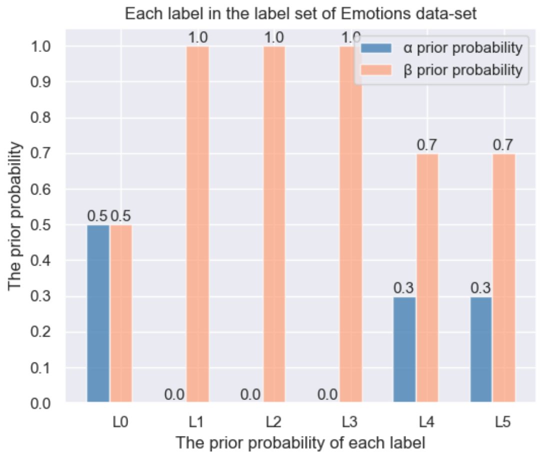
Before tackling these contradictions, we first introduce the *label prior probability.* In the ML-KNN [30], the model firstly identified k nearest neighbors in the training set for each unseen instance object $\mathbf{I}_i$, and then a membership counting vector is defined that is used to do frequency counting. From the frequency counting, the prior probability of each label of the unseen instance object $\mathbf{I}_i$ can be obtained. The prior probability means, in the k nearest neighbor instance object set of this unseen instance object $\mathbf{I}_i$, how many instance objects contain this label(marked as 1), and vice versa how many instance objects do not contain this label(marked as 0). Following their work, we get the k nearest neighbor from the **Step-1** of **Feature Chain Encoder** and then calculate the prior probability of each label. However, the subtle difference from their work is that we set the *positive label prior probability* α and the *negative label prior probability* β. α_i^v represents the number of instance objects that have the label $l_v \in \mathbf{L}_i$ in the k nearest neighbor instance objects for unseen instance object $\mathbf{I}_i$. β_i^v represents the number of instance objects that no the label l_v in the k nearest neighbor instance objects for unseen instance object $\mathbf{I}_i$. About the introduction of prior probability is shown in Figure 3.

Next, we would use the generation probability and prior probability of labels to merge the results of two decoders. Firstly, the decoded results of the two decoders are adjusted for merging. Suppose the unseen instance object $\mathbf{I}_i$ has the un-predicted label set $\{l_1, l_2, ..., l_n\}$. The decoder **LS-Decoder**$_p$ predicts label $\{l_2, l_3, ...\} \in \mathbb{R}_1^m (m_1 \leq m)$ as positive labels. Next, the positive label set $\{l_2, l_3, ...\}$ is extended as the positive full label generation probability set $P_{set} = \{None, g_2, g_3, ...\} \in \mathbb{R}^n$. The g_2(or g_3) is the generation probability of label l_2(or g_3) by the decoder. On the contrary, the $None$ represents that the decoder doesn't predict this label as a positive label. Thus, using this adjust methods, we can get the positive full label generation probability set $P_{set} \in \mathbb{R}^n$ and the negative full label generation probability set $N_{set} \in \mathbb{R}^n$. Then the positive prior probability α and negative prior probability β is defined for the un-predicted label set. Setting the $\alpha_{set} = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ are the positive prior probability for the un-predict labels set $\{l_1, l_2, ..., l_n\}$. α_j represents that the percentage of having the label l_j in the k nearest neighbor instance objects of the unseen instance object $\mathbf{I}_i$. Thus, we can get the positive prior probability set $\alpha_{set} \in \mathbb{R}^n$ and negative prior probability set $\beta_{set} \in \mathbb{R}^n$. At the same time,



	L0	L1	L2	L3	L4	L5
I 0	0	0	0	0	1	0
I 1	1	0	0	0	0	0
I 2	1	0	0	0	0	0
I 3	1	0	0	0	1	0
I 4	0	0	0	0	0	1
I 5	0	0	0	0	0	1
I 6	0	0	0	0	1	0
I 7	0	0	0	0	0	1
I 8	1	0	0	0	0	0
I 9	1	0	0	0	0	0
Unseen	-1	-1	-1	-1	-1	-1
Real	1	0	0	0	1	0

(a) This is the heat-map of the label set of instance objects. The top 10 lines of this heat-map are the 10 instance objects which are similar to the current instance object. The 11-th line is the current instance object that we assume its label set is unseen(marked as -1, color is red). The 12-th line is the real label set of this current instance object.



(b) This is the corresponding statistical histogram of α prior probability and β prior probability of the top 10 instance objects of left sub-figure.

Fig. 3 The subfigure A is the label distribution heat-map of 12 instance objects and the subfigure B is the corresponding label prior probability histogram map of some instance objects. The blue color represents that instance object has this label(marked as 1) and the orange color represents that instance object has not the label(marked as 0). The red color represents the label of this instance is unseen(marked as -1).

the positive coefficient p and negative coefficient q are introduced to assist the process of merging. The positive coefficient p indicates the possibility of simultaneously considering two decoding results to generate a positive label. In contrary, the negative coefficient q indicates the possibility of generating a negative label. The pseudo-code of **Algorithms 1** below is the ensemble method.

Algorithm 1 The ensemble methods for merging the results of two decoders.

```
Input: $P_{set}$, $N_{set}$, $\alpha_{set}$, $\beta_{set}$, $p$, $q$, $\{l_1, l_2, ..., l_n\}$.
 1: for $i \in [1, n]$ do
 2:     set $p = 0$ and $q = 0$;
 3:     $if\ P_{set}[i]\ =\ None:$
 4:         $q\ +=\ 1$
 5:     $else:$
 6:         $p\ +=\ P_{set}[i]$
 7:     $if\ N_{set}[i]\ =\ None:$
 8:         $p\ +=\ 1$
 9:     $else:$
10:         $q\ +=\ N_{set}[i]$
11:     $if\ p \times \alpha_{set}[i] > q \times \beta_{set}[i]:$
12:         $l_i\ =\ 1$
13:     $else:$
14:         $l_i\ =\ 0$
15: end for
Output: $\{l_1, l_2, ..., l_n\}\ =\ LGSM(P_{set}, N_{set}, \alpha_{set}, \beta_{set})$.
```

4 Experiments

In this section, the LSGM model is evaluated at four real-world multi-label learning data-sets. The data-sets, evaluation metrics, experimental details, and all baselines are introduced firstly. Then, the experimental results are provided. Lastly, the LSGM model is compared and analyzed with the baselines.

4.1 Data-sets

To thoroughly evaluate the performance of the proposed LSGM model, a total of four real-world multi-label learning data-sets are studied in this paper. The statistics of the four data-sets are shown in Table 1 which summarizes the detailed characteristics of those data-sets used in experiments.

4.2 Evaluation Metrics

Following the previous work [31], five widely-used multi-label learning metrics are employed for performance evaluation accordingly including:

– ***Hamming Loss***: Hamming Loss evaluates how many times an instance-label pair is misclassified, i.e. a label not belonging to the instance is predicted or a label belonging to the instance is not predicted.
– ***One-error***: One-errorevaluates how many times the top-ranked label is not in the set of proper labels of the instance.
– ***Coverage***: Coverage evaluates how far we need, on the average, to go down the list of labels in order to cover all the proper labels of the instance.

Table 1 Characteristics of the experimental data-sets. The *Instances* represents the number of instance objects, the *Features* represents the dimension of the feature vector and the *Labels* represents the number of labels. The Instances represents the number of instance objects, the Features represents the dimension of the feature vector and the Labels represents the number of labels. The *Label Cardinality* counts the average number of relevant labels per example; the *Label Density* normalizes label cardinality by the number of class labels; the *Label Distinct* sets counts the number of distinct sets of relevant labels existing in training data set[20, 21].

	Scene	**Yeast**	**Emotions**	**Image**
Domain	Image	Biology	Music	Image
Instances	2407	2417	593	2000
Features	294	103	72	294
Labels	6	14	6	5
Label Cardinality	1.074	4.237	1.869	1.236
Label Density	0.179	0.303	0.311	0.010
Label Distinct	15	198	27	20

- ***Ranking Loss***: Ranking Loss evaluates the average fraction of label pairs that are reversely ordered for the instance.
- ***Average Precision***: Average Precision evaluates the average fraction of relevant labels ranked higher than a particular label.

For hamming loss, one-error, coverage and ranking loss, the smaller the metric values the better the performance. For average precision, the greater the metric values the better the performance. The details of these metrics can be found in [31].

4.3 Experiments Details

On each data-set, ten-fold cross-validation is conducted. The mean value and standard deviation of each evaluation criterion are recorded. In detail, in each cross-validation, 90% of data-set is used for training, 2% of data-set is used for validating and 8% of data-set for test infer. The number of Bi-LSTM layers of the encoder and the number of LSTM layers of the decoder are both 2. We use the Adam[12] optimization method to minimize the Negative Log-Likelihood loss(NLLLoss) function over the training data.

For the hyper-parameters of the Adam optimizer, we set the learning rate α=1e-5 and others are default. Additionally, we make use of the dropout regularization to avoid over-fitting, set as 0.1. During training, we train the model as a fixed number of epochs 100 and monitor its performance. Once the training is finished, we select the model with the best average precision score on the validation set as our final model and evaluate its performance on the test set.

In order to better illustrate the effectiveness of the proposed ensemble methods in the LSGM model, we compare three LSGM models with different structures. The *LSGM-P* means that the decoder of this model is only the

LS-Decoder$_P$ and the *LSGM-N* means that the decoder of this model is only the **LS-Decoder**$_N$. Especially, the *LSGM-P* can be acted as the EncDec model [16]. They both use LSTM neural network to predict the positive labels. Besides, the *LSGM-M* means that the decoders of this model are the **LS-Decoder**$_P$ and **LS-Decoder**$_N$. It uses the ensemble methods to merge the results of two decoders. The model is implemented in Python, compiled by Pytorch version 1.2.0 under Windows. These experiments are executed on a server with 16GB memory and a GTX-1080 GPU.

4.4 Baselines

We compare our proposed **LSGM** with the following baselines:

- **ML-KNN**: ML-KNN [30] is a first-oder algorithm where for each unknown instance its k nearest neighbors in the training set are firstly identified, and then the maximum a posteriori (MAP) principle is utilized to determine the label set for that.
- **BP-MLL**: BP-MLL [29] is second-order algorithm and utilizes a fully-connected neural network and a pairwise ranking loss function to deal with the MLL task.
- **CC**: Classifier chains [18] is a high-order algorithm which transforms the MLL task into a chain of binary classification problems, where subsequent binary classifiers in the chain are built upon the predictions of preceding ones.
- **BR**: Binary Relevance [2] is a first-order approach which decomposes MLL problem into binary classification problems, one for each class label.
- **LIFT**: Label specIfic FeaTures[28] is one of the representative multi-label learning approaches based on label-specific features, which works by utilizing clustering techniques to explore the underlying structures of feature space. It is a first-order algorithm.

For BP-MLL, the number of hidden neurons is set to be 20% of the input dimensionality and the maximum number of training epochs is set to be 100. For ML-KNN, the number of nearest neighbors considered is set to be 10. The experimental data of the comparison algorithms comes from [28].

4.5 Results and Analysis

All experimental results are shown in Table 2. From the results of three LSGM model, all of the evaluation metrics of the *LSGM-M* are better than the *LSGM-P* and *LSGM-N.* This illustrates the effectiveness of the proposed ensemble methods from the experimental results. From the perspective of various evaluation metrics, *LSGM-M* is always better than the other two decoders in the range of 1% to 9%. Figure 4 is about the average precision of every algorithms on 4 data-set. From the histogram and error bars in Figure

4, we can see that the model performance of *LSGM-M* is better and more stable than the other two. Besides, the performance of *LSGM-N* is not always worse than the performance of *LSGM-P*. Especially in the *Image* data-set, the former is better than the latter and closes to the *LSGM-M*. Under receiving the same encoder output, the decoded result of **LS-Decoder**$_N$ is better than the decoded results of decoder **LS-Decoder**$_P$. Based on the property of the seq2seq task that the shorter the decoding length, the better the decoding effect in the case of the same other run conditions, this indirectly indicates that the number of negative labels is not always more than the number of positive labels.

Both as the algorithms that use neural networks to tackle multi-label learning tasks, our proposed *LSGM-M* model has all 20 evaluation metrics outperformed the *BP-MLL* algorithms among 20 evaluation metrics(4 data-sets $\times$ 5 criteria). Both as the algorithms that use the high-order label correlation, the *LSGM-M* is superior to **CC** in the 90% compared evaluation metrics. Across all the 20 configurations, *LSGM-M* ranks in *1st* place among the eight comparing algorithms at 50% cases, in *2nd* place at 25% cases. Especially, in the **Ranking Loss** evaluation metric, the *LSGM-M* is superior to other all algorithms. From the Figure 4, the performance of *LSGM-M* is closed to the current best algorithms **LIFT**, even better than the latter about 0.3% in the **Scene** and **Yeast** data-sets. These results clearly validate the better effectiveness of our approach to the other well established multi-label learning algorithms.

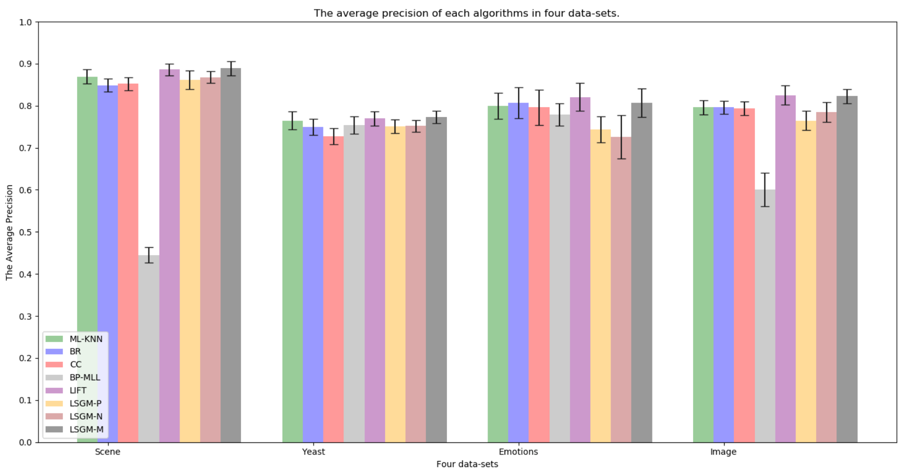


Fig. 4 The Average Precision of 8 algorithms in 4 different data-sets. The black line at the top of each color bars are the error range.

Table 2 Experimental results of each comparing algorithm (mean±std. deviation) on the four data-sets. The k is the number of nearest neighbors for unseen instance object. The k value may be different for different data-sets. For each evaluation criterion, $\Downarrow$ indicates the smaller the better while $\Uparrow$ indicates the larger the better. The **H-L** represents the **Hamming Loss**, **O-E** represents the **One-error**, the **Cov** represents the **Coverage**, the **R-L** represents the **Ranking Loss** and the **A-P** represents the **Average Precision**. The Eva. represents the Evaluation Metrics. Furthermore, the best performance among the comparing algorithms is highlighted in boldface.

Eva.	Methods	Scene(k=10)	Yeast(k=11)	Emotions(k=9)	Image(k=8)
H-L $\Downarrow$	ML-KNN	0.084±0.008	**0.195±0.011**	0.194±0.013	0.170±0.008
	BR	0.104±0.006	0.199±0.010	0.199±0.022	0.176±0.007
	CC	0.096±0.010	0.208±0.010	0.192±0.027	0.180±0.015
	BP-MLL	0.282±0.014	0.205±0.009	0.219±0.021	0.253±0.024
	LIFT	**0.077±0.009**	0.197±0.002	**0.188±0.021**	**0.156±0.017**
	LSGM-P	0.102±0.009	0.203±0.014	0.229±0.026	0.194±0.022
	LSGM-N	0.098±0.009	0.213±0.017	0.259±0.051	0.181±0.014
	LSGM-M	0.080±0.007	**0.195±0.012**	0.204±0.026	**0.156±0.010**
O-E $\Downarrow$	ML-KNN	0.219±0.029	0.228±0.029	0.263±0.067	0.320±0.026
	BR	0.250±0.027	0.230±0.023	0.253±0.070	0.314±0.021
	CC	0.226±0.034	**0.176±0.022**	**0.216±0.085**	0.289±0.026
	BP-MLL	0.821±0.031	0.235±0.031	0.318±0.057	0.600±0.079
	LIFT	0.196±0.026	0.221±0.020	0.243±0.074	**0.266±0.037**
	LSGM-P	0.221±0.031	0.283±0.028	0.269±0.068	0.327±0.055
	LSGM-N	0.212±0.030	0.297±0.064	0.319±0.070	0.314±0.039
	LSGM-M	**0.194±0.028**	0.254±0.051	0.273±0.084	0.270±0.027
Cov $\Downarrow$	ML-KNN	0.078±0.010	0.447±0.014	0.300±0.019	0.194±0.020
	BR	0.089±0.009	0.514±0.018	0.295±0.027	0.189±0.021
	CC	0.091±0.008	0.516±0.015	0.322±0.022	0.199±0.020
	BP-MLL	0.374±0.024	0.456±0.019	0.300±0.022	0.343±0.029
	LIFT	**0.065±0.007**	0.452±0.015	**0.281±0.022**	**0.168±0.019**
	LSGM-P	0.099±0.012	0.477±0.024	0.294±0.040	0.204±0.023
	LSGM-N	0.096±0.014	0.458±0.021	0.304±0.043	0.189±0.020
	LSGM-M	0.072±0.010	**0.446±0.022**	0.288±0.035	**0.168±0.017**
R-L $\Downarrow$	ML-KNN	0.076±0.012	0.166±0.015	0.163±0.022	0.175±0.019
	BR	0.089±0.011	0.200±0.013	0.156±0.034	0.169±0.019
	CC	0.135±0.013	0.285±0.022	0.233±0.040	0.245±0.024
	BP-MLL	0.434±0.026	0.171±0.015	0.173±0.020	0.366±0.037
	LIFT	0.062±0.008	0.163±0.011	0.144±0.024	0.143±0.010
	LSGM-P	0.074±0.007	0.169±0.007	0.168±0.018	0.233±0.013
	LSGM-N	0.073±0.006	0.158±0.007	0.194±0.055	0.189±0.011
	LSGM-M	**0.058±0.005**	**0.146±0.006**	**0.137±0.017**	**0.137±0.005**
A-P $\Uparrow$	ML-KNN	0.869±0.017	0.765±0.021	0.799±0.031	0.792±0.017
	BR	0.849±0.016	0.749±0.019	0.807±0.037	0.796±0.015
	CC	0.852±0.016	0.728±0.019	0.796±0.042	0.794±0.016
	BP-MLL	0.445±0.018	0.754±0.020	0.779±0.227	0.601±0.040
	LIFT	0.886±0.014	0.770±0.017	**0.821±0.033**	**0.825±0.023**
	LSGM-P	0.862±0.022	0.751±0.016	0.743±0.031	0.765±0.023
	LSGM-N	0.868±0.014	0.752±0.014	0.726±0.052	0.785±0.023
	LSGM-M	**0.889±0.017**	**0.773±0.015**	0.807±0.034	0.823±0.017

5 Ablation Studies

In this section, we perform ablation experiments over three facets of LSGM in order to better explain, which is the value of k nearest neighbors, the type of feature chain and the effect of Global Label Information.

5.1 The Value of k Nearest Neighbor

In the above **Feature Chain Encoder** section, the model searches for k nearest neighbors instance objects similar to the unseen instance object $\mathbf{I}_i$ from the graph $\mathbf{G}$ and matrix $\mathbf{S}$. Thus, how many nearest neighbor instance objects need to be searched becomes a problem needing to explore. Too little neighbor instance objects may not be sufficient to describe the feature information around the unseen instance object. Too many nearest neighbor instance objects can cause too high computational complexity because the length of feature chain is twice as long as the k value. Besides, the longer the feature chain does not mean that the effect of the model is better, because of adding too much noise data or interfering the *chain-type link* among the feature vector. In addition, the k value is also related to the prior probability. Based on the above analysis, we set this ablation study about the value of k nearest neighbors.

Figure 5 is the result of ablation study. In the experiments, we choose the **Emotions** data-set, *Global Label Information*, and the ensemble methods(i.e. the *LSGM-M model*). In the process of changing the value of k from 1 to 7, the performance of *LSGM-M* is gradually improved. Next, when the value of k is 8, the performance of the model suddenly drops. But when the value of k is 9, the model achieves the best performance. Then, as the k value is getting bigger, the performance of the model is getting worse.

The above ablation study on k nearest neighbors basically verified our hypothesis that the performance of the model is related to the value of k. The k values required for different data-sets are different. From the experimental comparison, the *LSGM-M model* is generally optimal at k equal to about 10.

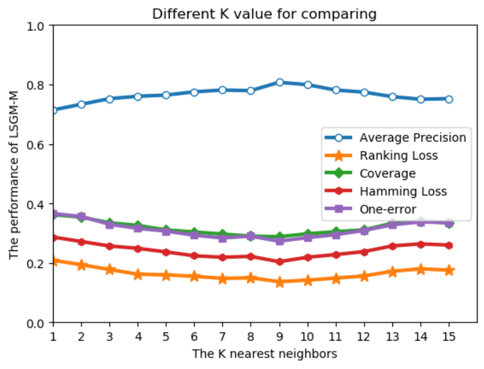
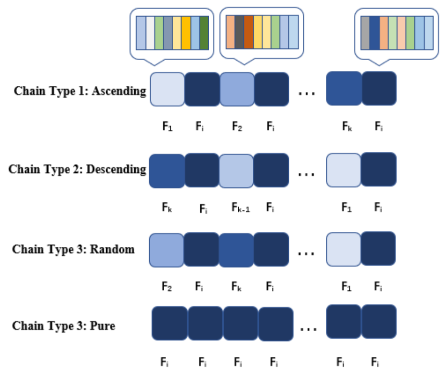
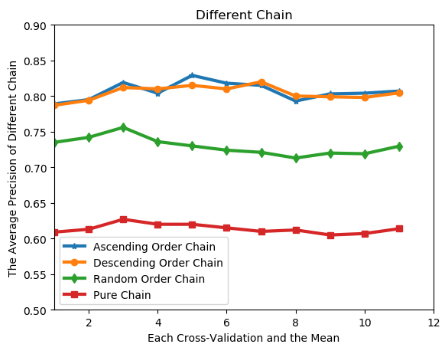


Fig. 5 The performance of LSGM-M with different k value in Emotions data.



(a) They are the *ascending* order feature chain, *descending* feature chain, *random* order feature chain, and *pure* feature chain. The different color blocks represent the different feature vectors.



(b) The is the corresponding line charts of LSGM-M model which uses four different types of feature chain in Emotions data-set. The **Average Precision** of model in every cross-validations and the mean value are recorded.

Fig. 6 The left sub-figure is the visualization of different feature chain. The right sub-figure is the average precision line chart of LSGM-M model with different feature chains in Emotions data-set.

5.2 The Type of Feature Chain

In the above **Step-2 Chain-Type Feature Vectors Sequence** of **Feature Chain Encoder**, we get the mix instance objects sorting sequence $\{\mathbf{I}_1, \mathbf{I}_i, \mathbf{I}_2, ..., \mathbf{I}_k, \mathbf{I}_i\}$. The k instance objects $\{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_k\}$ are the k nearest neighbor of the unseen instance object $\mathbf{I}_i$. The k instance objects are sorted in ascending order. The more similar to the unseen instance object $\mathbf{I}_i$, the more behind the k-th nearest neighbor instance object sort. And then mix instance objects sequence is converted to the mix instance objects feature vectors sequence $\{\mathbf{F}_1, \mathbf{F}_i, \mathbf{F}_2, \mathbf{F}_i, ..., \mathbf{F}_k, \mathbf{F}_i\}$.

We choose the ascending order to sort the k nearest neighbor instance objects and the construct the chain-type feature vectors sequence. Thus, there is a question that which methods should be used to layout the k nearest neighbor instance objects. In order to explore the order of these nearest neighbor instances objects, we set 3 cases to sort the k nearest neighbor instance object, one is the *ascending* order $\{\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_k\}$, one is the *descending* order $\{\mathbf{I}_k, \mathbf{I}_{k-1}, ..., \mathbf{I}_1\}$ and another one is the *random* order $\{\mathbf{I}_2, \mathbf{I}_k, ..., \mathbf{I}_1\}$. In particular, in order to explore whether needs to join the nearest neighbor instance objects to construct the feature chain, we set up a *pure* instance object sequence $\{\mathbf{I}_i, \mathbf{I}_i, ..., \mathbf{I}_i\}$, that is, the feature chain is only constructed by the unseen instance $\mathbf{I}_i$. Because there is no nearest neighbor instance object, the label positive or negative prior probabilities of this case are both 1. Then the unseen instance object $\mathbf{I}_i$ is inserted into the four types chain. Thus, the length of four types of instance object sequence are $(k + k)$. The visualization of the four chains is shown as the (a) sub-figure of Figure 6.

In the (b) sub-figure of Figure 6, the performance of the descending order chain and the ascending order chain is similar. This is logical because the encoder uses a bidirectional LSTM and captures both the forward and backward information of feature chain, which is equivalent to the ascending and descending order of the feature chain. However, the performance of random order chain is worse than the former because it is maybe hurt the the close or alienated correlation between the k nearest neighbor instance objects and unseen instance object. In this case, it maybe hard for the encoder to understand the potential meaning of the k nearest neighbor instance objects and the unseen instance object. The *pure* chain performs the worst. This illustrates the necessity to add the nearest neighbor instance objects for encoding and the importance of the prior probability to the model.

5.3 The Global Label Information

In the above **Step-2 Global Label Information** of **Label Sequence Decoder** section, we put forward to actively alleviate the exposure bias problem of seq2seq task by superimposing the labels which had been predicted. That is(take the **LS-Decoder**$_P$ as example),

$$\begin{aligned}&\mathbf{s}_t^p = \textbf{LS-Decoder}_p(\mathbf{s}_{t-1}^p, \sum_{v=1}^{t-1} l_v^p, \mathbf{C}, \mathbf{h}_{all})\\&\Rightarrow P_i^p(l_1, l_2, ..., l_{i-1}|\mathbf{F}_{chain})\end{aligned}$$

where the $\sum_{v=1}^{t-1} l_v^p$ is corresponding to the $\{l_1, l_2, ..., l_{i-1}\}$ and the $\Rightarrow$ represents the intermediate process, which is omitted for simplicity. That means at the t-th moment, the decoder calculates the currently hidden state $\mathbf{s}_t^p$ by superimposing the labels $\{l_1, l_2, ..., l_{t-1}\}$ which had been predicted. And then the **LS-Decoder**$_p$ uses the hidden state $\mathbf{s}_t^p$ to predict that the i-th label of the label set $\mathbf{L}$ is a positive label. The model can perceive more labels of information by superimposing the predicted labels. In the process of superimposing, we think that the high-order label correlation is considered into the model, which help the model predict more precision.

Figure 7 shows the results of the model with or without *Global Label Information*(GLI). In the **Average Precision** evaluate metrics and the 11 points(include each cross-validation and the last mean value), the red line(with GLI) is higher(better) than the green line(without GLI) at 81.8% cases. In the **Hamming Loss** evaluate metrics, the red line is lower(better) than the green line at 90.9% cases. Lastly, the mean values(include the **Average Precision** and **Hamming Loss**) of the model with GLI are both better than the model without GLI. This shows the validity of our proposed *Global Label Information.*

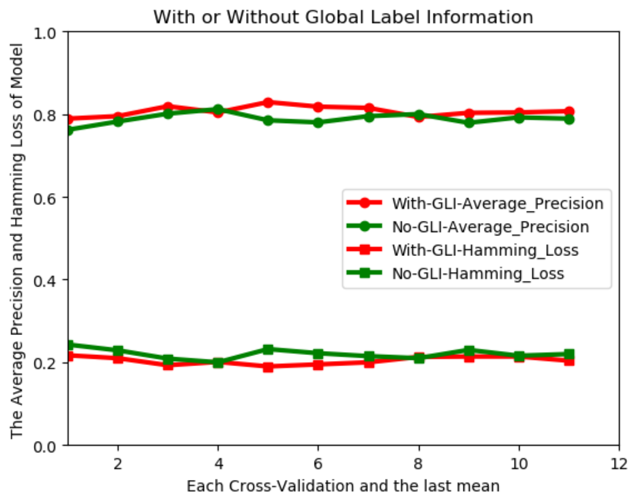


Fig. 7 The performance of LSGM-M with or without Global Label Information in Emotions data-set.

6 Conclusion and Future Work

In this paper, we convert the multi-label learning task to the seq2seq task of multi-label learning and give formal transformation process. The feature chain encoder of proposed LSGM model arranges the multiple feature vectors and unseen instance object into the chain-type feature vectors sequence with a special rule. The sequence is encoded as a representation vector and transmitted to label sequence decoder. Next, the model splits the label set to two label sub-sequence by different label attributes and then merges the decoded prediction results of two sub-sequences using ensemble methods. During decoding, the Global Label Information is proposed to actively alleviate the exposure bias problem. The extensive experimental results and ablation studies show that the converting process is reasonable and the various parts of the LSGM model are effective.

In the ablation study, different sort orders of feature vectors have different effects on the model predictions. In future work, it is necessary to further explore the more efficient sort order of feature vectors. In addition, we want to try to use the *Transformer* [23] instead of the LSTM as the feature extractor.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Boutell, M.R., Luo, J., Shen, X., Brown, C.M.: Learning multi-label scene classification. Pattern recognition **37**(9), 1757–1771 (2004)
3. Clare, A., King, R.D.: Knowledge discovery in multi-label phenotype data. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 42–53. Springer (2001)

4. Dong, H.C., Li, Y.F., Zhou, Z.H.: Learning from semi-supervised weak-label data. AAAI (2018)
5. Elisseeff, A., Weston, J.: A kernel method for multi-labelled classification. In: Advances in neural information processing systems, pp. 681–687 (2002)
6. Freitag, M., Al-Onaizan, Y.: Beam search strategies for neural machine translation. arXiv preprint arXiv:1702.01806 (2017)
7. Fürnkranz, J., Hüllermeier, E., Mencía, E.L., Brinker, K.: Multilabel classification via calibrated label ranking. Machine learning **73**(2), 133–153 (2008)
8. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: Proceedings of the 14th ACM international conference on Information and knowledge management, pp. 195–200. ACM (2005)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
10. Huang, Y., Wang, W., Wang, L., Tan, T.: Multi-task deep neural network for multi-label learning. In: 2013 IEEE International Conference on Image Processing, pp. 2897–2900. IEEE (2013)
11. Ji, S., Tang, L., Yu, S., Ye, J.: Extracting shared subspace for multi-label classification. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 381–389. ACM (2008)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Kong, X., Ng, M.K., Zhou, Z.H.: Transductive multilabel learning via label set propagation. IEEE Transactions on Knowledge and Data Engineering **25**(3), 704–719 (2013)
14. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
15. Nam, J., Kim, J., Mencía, E.L., Gurevych, I., Fürnkranz, J.: Large-scale multi-label text classificationrevisiting neural networks. In: Joint european conference on machine learning and knowledge discovery in databases, pp. 437–452. Springer (2014)
16. Nam, J., Mencía, E.L., Kim, H.J., Fürnkranz, J.: Maximizing subset accuracy with recurrent neural networks in multi-label classification. In: Advances in neural information processing systems, pp. 5413–5423 (2017)
17. Omohundro, S.M.: Five balltree construction algorithms. International Computer Science Institute Berkeley (1989)
18. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine learning **85**(3), 333 (2011)
19. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp. 3104–3112 (2014)
20. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Data mining and knowledge discovery handbook, pp. 667–685. Springer (2009)
21. Tsoumakas, G., Katakis, I., Vlahavas, I.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering **23**(7), 1079–1089 (2011)
22. Tsoumakas, G., Vlahavas, I.: Random k-labelsets: An ensemble method for multilabel classification. In: European conference on machine learning, pp. 406–417. Springer (2007)
23. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems, pp. 5998–6008 (2017)
24. Yang, B., Sun, J.T., Wang, T., Chen, Z.: Effective multi-label active learning for text classification. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 917–926. ACM (2009)
25. Yang, P., Sun, X., Li, W., Ma, S., Wu, W., Wang, H.: Sgm: sequence generation model for multi-label classification. In: Proceedings of the 27th International Conference on Computational Linguistics, pp. 3915–3926 (2018)
26. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: AAAI, pp. 2852–2858 (2017)
27. Zhang, M.L.: M l-rbf: Rbf neural networks for multi-label learning. Neural Processing Letters **29**(2), 61–74 (2009)
28. Zhang, M.L., Wu, L.: Lift: Multi-label learning with label-specific features. IEEE transactions on pattern analysis and machine intelligence **37**(1), 107–120 (2015)

29. Zhang, M.L., Zhou, Z.H.: Multilabel neural networks with applications to functional genomics and text categorization. IEEE transactions on Knowledge and Data Engineering **18**(10), 1338–1351 (2006)
30. Zhang, M.L., Zhou, Z.H.: Ml-knn: A lazy learning approach to multi-label learning. Pattern recognition **40**(7), 2038–2048 (2007)
31. Zhang, M.L., Zhou, Z.H.: A review on multi-label learning algorithms. IEEE transactions on knowledge and data engineering **26**(8), 1819–1837 (2014)