SGM: Sequence Generation Model for Multi-Label Classification

Pengcheng Yang[1,2], Xu Sun[1,2], Wei Li[2], Shuming Ma[2], Wei Wu[2], Houfeng Wang[2]
[1]Deep Learning Lab, Beijing Institute of Big Data Research, Peking University
[2]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
{yang_pc, xusun, liweitj47, shumingma, wu.wei, wanghf}@pku.edu.cn

Abstract

Multi-label classification is an important yet challenging task in natural language processing. It is more complex than single-label classification in that the labels tend to be correlated. Existing methods tend to ignore the correlations between labels. Besides, different parts of the text can contribute differently to predicting different labels, which is not considered by existing models. In this paper, we propose to view the multi-label classification task as a sequence generation problem, and apply a sequence generation model with a novel decoder structure to solve it. Extensive experimental results show that our proposed methods outperform previous work by a substantial margin. Further analysis of experimental results demonstrates that the proposed methods not only capture the correlations between labels, but also select the most informative words automatically when predicting different labels.[1]

1 Introduction

Multi-label classification (MLC) is an important task in the field of natural language processing (NLP), which can be applied in many real-world scenarios, such as text categorization (Schapire and Singer, 2000), tag recommendation (Katakis et al., 2008), information retrieval (Gopal and Yang, 2010), and so on. The target of the MLC task is to assign multiple labels to each instance in the dataset.

Binary relevance (BR) (Boutell et al., 2004) is one of the earliest attempts to solve the MLC task by transforming the MLC task into multiple single-label classification problems. However, it neglects the correlations between labels. Classifier chains (CC) proposed by Read et al. (2011) converts the MLC task into a chain of binary classification problems to model the correlations between labels. However, it is computationally expensive for large datasets. Other methods such as ML-DT (Clare and King, 2001), Rank-SVM (Elisseeff and Weston, 2002), and ML-KNN (Zhang and Zhou, 2007) can only be used to capture the first or second order label correlations or are computationally intractable when high-order label correlations are considered.

In recent years, neural networks have achieved great success in the field of NLP. Some neural network models have also been applied in the MLC task and achieved important progress. For instance, fully connected neural network with pairwise ranking loss function is utilized in Zhang and Zhou (2006). Kurata et al. (2016) propose to perform classification using the convolutional neural network (CNN). Chen et al. (2017) use CNN and recurrent neural network (RNN) to capture the semantic information of texts. However, they either neglect the correlations between labels or do not consider differences in the contributions of textual content when predicting labels.

In this paper, inspired by the tremendous success of the sequence-to-sequence (Seq2Seq) model in machine translation (Bahdanau et al., 2014; Luong et al., 2015; Sun et al., 2017), abstractive summarization (Rush et al., 2015; Lin et al., 2018), style transfer (Shen et al., 2017; Xu et al., 2018) and other domains, we propose a sequence generation model with a novel decoder structure to solve the MLC task. The proposed sequence generation model consists of an encoder and a decoder with the attention

[1]The datasets and code are available at https://github.com/lancopku/SGM

mechanism. The decoder uses an LSTM to generate labels sequentially, and predicts the next label based on its previously predicted labels. Therefore, the proposed model can consider the correlations between labels by processing label sequence dependencies through the LSTM structure. Furthermore, the attention mechanism considers the contributions of different parts of text when the model predicts different labels. In addition, a novel decoder structure with global embedding is proposed to further improve the performance of the model by incorporating overall informative signals.

The contributions of this paper are listed as follows:

- We propose to view the MLC task as a sequence generation problem to take the correlations between labels into account.
- We propose a sequence generation model with a novel decoder structure, which not only captures the correlations between labels, but also selects the most informative words automatically when predicting different labels.
- Extensive experimental results show that our proposed methods outperform the baselines by a large margin. Further analysis demonstrates the effectiveness of the proposed methods on correlation representation.

The whole paper is organized as follows. We describe our methods in Section 2. In Section 3, we present the experiments and make analysis and discussions. Section 4 introduces the related work. Finally in Section 5 we conclude this paper and explore the future work.

2 Proposed Method

We introduce our proposed methods in detail in this section. First, we give an overview of the model in Section 2.1. Second, we explain the details of the proposed sequence generation model in Section 2.2. Finally, Section 2.3 presents our novel decoder structure.

2.1 Overview

First of all, we define some notations and describe the MLC task. Given the label space with L labels $\mathcal{L} = \{l_1, l_2, \cdots, l_L\}$, a text sequence $\boldsymbol{x}$ containing m words, the task is to assign a subset $\boldsymbol{y}$ containing n labels in the label space $\mathcal{L}$ to $\boldsymbol{x}$. Unlike traditional single-label classification where only one label is assigned to each sample, each sample in the MLC task can have multiple labels. From the perspective of sequence generation, the MLC task can be modeled as finding an optimal label sequence $\boldsymbol{y}^*$ that maximizes the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$, which is calculated as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{n} p(y_i|y_1, y_2, \cdots, y_{i-1}, \boldsymbol{x}) \tag{1}$$

An overview of our proposed model is shown in Figure 1. First, we sort the label sequence of each sample according to the frequency of the labels in the training set. High-frequency labels are placed in the front. In addition, the bos and eos symbols are added to the head and tail of the label sequence, respectively.

The text sequence $\boldsymbol{x}$ is encoded to the the hidden states, which are aggregated to a context vector $\boldsymbol{c}_t$ by the attention mechanism at time-step t. The decoder takes the context vector $\boldsymbol{c}_t$, the last hidden state $\boldsymbol{s}_{t-1}$ of the decoder and the embedding vector $g(\boldsymbol{y}_{t-1})$ as the inputs to produce the hidden state $\boldsymbol{s}_t$ at time-step t. Here $\boldsymbol{y}_{t-1}$ is the predicted probability distribution over the label space $\mathcal{L}$ at time-step $t-1$. The function g takes $\boldsymbol{y}_{t-1}$ as input and produces the embedding vector which is then passed to the decoder. Finally, the masked softmax layer is used to output the probability distribution $\boldsymbol{y}_t$.

2.2 Sequence Generation

In this subsection, we introduce the details of our proposed model. The whole sequence generation model consists of an encoder and a decoder with the attention mechanism.

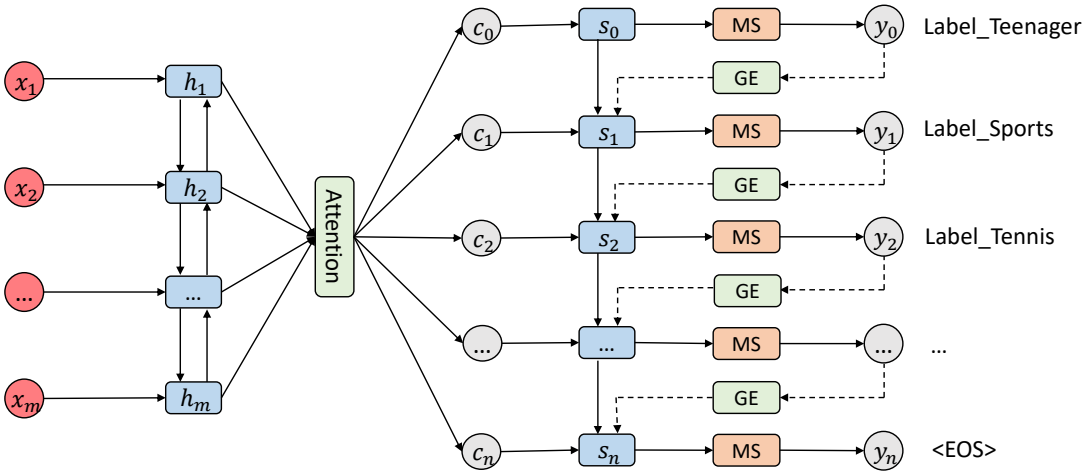

Figure 1: The overview of our proposed model. MS denotes the masked softmax layer. GE denotes the global embedding.

Encoder: Let $(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m)$ be a sentence with m words and $\boldsymbol{w}_i$ is the one-hot representation of the i-th word. We first embed $\boldsymbol{w}_i$ to a dense embedding vector $\boldsymbol{x}_i$ by an embedding matrix $\boldsymbol{E} \in \mathbb{R}^{k \times |\mathcal{V}|}$. Here $|\mathcal{V}|$ is the size of the vocabulary, and k is the dimension of the embedding vector.

We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to read the text sequence $\boldsymbol{x}$ from both directions and compute the hidden states for each word,

$$\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i) \tag{2}$$

$$\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i) \tag{3}$$

We obtain the final hidden representation of the i-th word by concatenating the hidden states from both directions, $\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i; \overleftarrow{\boldsymbol{h}}_i]$, which embodies the information of the sequence centered around the i-th word.

Attention: When the model predicts different labels, not all text words make the same contribution. The attention mechanism produces a context vector by focusing on different portions of the text sequence and aggregating the hidden representations of those informative words. Specially, the attention mechanism assigns the weight α_{ti} to the i-th word at time-step t as follows:

$$e_{ti} = \boldsymbol{v}_a^T \tanh(\boldsymbol{W}_a \boldsymbol{s}_t + \boldsymbol{U}_a \boldsymbol{h}_i) \tag{4}$$

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{m} \exp(e_{tj})} \tag{5}$$

where $\boldsymbol{W}_a$, $\boldsymbol{U}_a$, $\boldsymbol{v}_a$ are weight parameters and $\boldsymbol{s}_t$ is the current hidden state of the decoder at time-step t. For simplicity, all bias terms are omitted in this paper. The final context vector $\boldsymbol{c}_t$ which is passed to the decoder at time-step t is calculated as follows:

$$\boldsymbol{c}_t = \sum_{i=1}^{m} \alpha_{ti} \boldsymbol{h}_i \tag{6}$$

Decoder: The hidden state $\boldsymbol{s}_t$ of the decoder at time-step t is computed as follows:

$$\boldsymbol{s}_t = \text{LSTM}(\boldsymbol{s}_{t-1}, [g(\boldsymbol{y}_{t-1}); \boldsymbol{c}_{t-1}]) \tag{7}$$

where $[g(\boldsymbol{y}_{t-1}); \boldsymbol{c}_{t-1}]$ means the concatenation of the vectors $g(\boldsymbol{y}_{t-1})$ and $\boldsymbol{c}_{t-1}$. $g(\boldsymbol{y}_{t-1})$ is the embedding of the label which has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. $\boldsymbol{y}_{t-1}$ is the probability

distribution over the label space $\mathcal{L}$ at time-step $t-1$ and is computed as follows:

$$\boldsymbol{o}_t = \boldsymbol{W}_o f(\boldsymbol{W}_d \boldsymbol{s}_t + \boldsymbol{V}_d \boldsymbol{c}_t) \tag{8}$$

$$\boldsymbol{y}_t = softmax(\boldsymbol{o}_t + \boldsymbol{I}_t) \tag{9}$$

where $\boldsymbol{W}_o$, $\boldsymbol{W}_d$, and $\boldsymbol{V}_d$ are weight parameters, $\boldsymbol{I}_t \in \mathbb{R}^L$ is the mask vector that is used to prevent the decoder from predicting repeated labels, and f is a nonlinear activation function.

$$(\boldsymbol{I}_t)_i = \begin{cases} -\infty & \text{if the label } l_i \text{ has been predicted at previous } t-1 \text{ time steps.} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

At the training stage, the loss function is the cross-entropy loss function. We employ the beam search algorithm (Wiseman and Rush, 2016) to find the top-ranked prediction path at inference time. The prediction paths ending with the eos are added to the candidate path set.

2.3 Global Embedding

In the sequence generation model mentioned above, the embedding vector $g(\boldsymbol{y}_{t-1})$ in Equation (7) is the embedding of the label that has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. However, this calculation only takes advantage of the maximum value of $\boldsymbol{y}_{t-1}$ greedily. The proposed sequence generation model generates labels sequentially and predicts the next label conditioned on its previously predicted labels. 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, which is also called *exposure bias*. To a certain extent, the beam search algorithm alleviates this problem. However, it can not fundamentally solve the problem because the *exposure bias* phenomenon is likely to occur for all candidate paths. $\boldsymbol{y}_{t-1}$ represents the predicted probability distribution at time-step $t-1$, so it is obvious that all information in $\boldsymbol{y}_{t-1}$ is helpful when we predict the current label at time-step t. The *exposure bias* problem ought to be relieved by considering all informative signals contained in $\boldsymbol{y}_{t-1}$.

Based on this motivation, we propose a new decoder structure, where the embedding vector $g(\boldsymbol{y}_{t-1})$ at time-step t is capable of representing the overall information at $(t-1)$-th time step. Inspired by the idea of the adaptive gate in highway network (Srivastava et al., 2015), here we introduce our global embedding. Let $\boldsymbol{e}$ denotes the embedding of the label which has the highest probability under the distribution $\boldsymbol{y}_{t-1}$. $\bar{\boldsymbol{e}}$ is the weighted average embedding at time t, which is calculated as follows:

$$\bar{\boldsymbol{e}} = \sum_{i=1}^{L} y_{t-1}^{(i)} \boldsymbol{e}_i \tag{11}$$

where $y_{t-1}^{(i)}$ is the i-th element of $\boldsymbol{y}_{t-1}$ and $\boldsymbol{e}_i$ is the embedding vector of the i-th label. Then the proposed global embedding $g(\boldsymbol{y}_{t-1})$ passed to the decoder at time-step t is as follows:

$$g(\boldsymbol{y}_{t-1}) = (\boldsymbol{1} - \boldsymbol{H}) \odot \boldsymbol{e} + \boldsymbol{H} \odot \bar{\boldsymbol{e}} \tag{12}$$

where $\boldsymbol{H}$ is the transform gate controlling the proportion of the weighted average embedding:

$$\boldsymbol{H} = \boldsymbol{W}_1 \boldsymbol{e} + \boldsymbol{W}_2 \bar{\boldsymbol{e}} \tag{13}$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{L \times L}$ are weight matrices. The global embedding $g(\boldsymbol{y}_{t-1})$ is the optimized combination of the original embedding and the weighted average embedding by using transform gate $\boldsymbol{H}$, which can automatically determine the combination factor in each dimension. $\boldsymbol{y}_{t-1}$ contains the information of all possible labels. By considering the probability of every label, the model is capable of reducing damage caused by mispredictions made in the previous time steps. This enables the model to predict label sequences more accurately.

Dataset	Total Samples	Label Sets	Words/Sample	Labels/Sample
RCV1-V2	804,414	103	123.94	3.24
AAPD	55,840	54	163.42	2.41

Table 1: Summary of datasets. **Total Samples**, **Label Sets** denote the total number of samples and labels, respectively. **Words/Sample** is the average number of words per sample and **Labels/Sample** is the average number of labels per sample.

3 Experiments

In this section, we evaluate our proposed methods on two datasets. We first introduce the datasets, evaluation metrics, experimental details, and all baselines. Then, we compare our methods with the baselines. Finally, we provide the analysis and discussions of experimental results.

3.1 Datasets

Reuters Corpus Volume I (RCV1-V2)[2]: This dataset is provided by Lewis et al. (2004). It consists of over 800,000 manually categorized newswire stories made available by Reuters Ltd for research purposes. Multiple topics can be assigned to each newswire story and there are 103 topics in total.
Arxiv Academic Paper Dataset (AAPD)[3]: We build a new large dataset for the multi-label text classification. We collect the abstract and the corresponding subjects of 55,840 papers in the computer science field from the website[4]. An academic paper may have multiple subjects and there are 54 subjects in total. The target is to predict corresponding subjects of an academic paper according to the content of the abstract.

We divide each dataset into training, validation and test sets. The statistics of the two datasets are shown in Table 1.

3.2 Evaluation Metrics

Following the previous work (Zhang and Zhou, 2007; Chen et al., 2017), we adopt hamming loss and micro-F_1 score as our main evaluation metrics. Micro-precision and micro-recall are also reported to assist the analysis.

- **Hamming-loss** (Schapire and Singer, 1999) evaluates the fraction of misclassified instance-label pairs, where a relevant label is missed or an irrelevant is predicted.
- **Micro-F_1** (Manning et al., 2008) can be interpreted as a weighted average of the precision and recall. It is calculated globally by counting the total true positives, false negatives, and false positives.

3.3 Details

We extract the vocabularies from the training sets. For the RCV1-V2 dataset, the size of the vocabulary is 50,000 and out-of-vocabulary (OOV) words are replaced with unk. Each document is truncated at the length of 500 and the beam size is 5 at the inference stage. Besides, we set the word embedding size to 512. The hidden sizes of the encoder and the decoder are 256 and 512, respectively. The number of LSTM layers of encoder and decoder is 2.

For the AAPD dataset, the size of word embedding is 256. There are two LSTM layers in the encoder and its size is 256. For the decoder, there is one LSTM layer of size 512. The size of the vocabulary is 30,000 and OOV words are also replaced with unk. Each document is truncated at the length of 500. The beam size is 9 at the inference stage.

We use the Adam (Kingma and Ba, 2014) optimization method to minimize the cross-entropy loss over the training data. For the hyper-parameters of the Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 1 \times 10^{-8}$. Additionally,

[2] http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[3] https://github.com/lancopku/SGM

[4] https://arxiv.org/

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0086	0.904	0.816	0.858
CC	0.0087	0.887	0.828	0.857
LP	0.0087	0.896	0.824	0.858
CNN	0.0089	**0.922**	0.798	0.855
CNN-RNN	0.0085	0.889	0.825	0.856
SGM	0.0081	0.887	0.850	0.869
+ GE	**0.0075**	0.897	**0.860**	**0.878**

(a) Performance on the RCV1-V2 test set.

Models	HL(-)	P(+)	R(+)	F1(+)
BR	0.0316	0.644	0.648	0.646
CC	0.0306	0.657	0.651	0.654
LP	0.0312	0.662	0.608	0.634
CNN	0.0256	**0.849**	0.545	0.664
CNN-RNN	0.0278	0.718	0.618	0.664
SGM	0.0251	0.746	0.659	0.699
+ GE	**0.0245**	0.748	**0.675**	**0.710**

(b) Performance on the AAPD test set.

Table 2: Comparison between our methods and all baselines on two datasets. GE denotes the global embedding. HL, P, R, and F1 denote hamming loss, micro-precision, micro-recall, and micro-F_1, respectively. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite.

we make use of the dropout regularization (Srivastava et al., 2014) to avoid overfitting and clip the gradients (Pascanu et al., 2013) to the maximum norm of 10.0. During training, we train the model for a fixed number of epochs and monitor its performance on the validation set. Once the training is finished, we select the model with the best micro-F_1 score on the validation set as our final model and evaluate its performance on the test set.

3.4 Baselines

We compare our proposed methods with the following baselines:

- **Binary Relevance (BR)** (Boutell et al., 2004) transforms the MLC task into multiple single-label classification problems by ignoring the correlations between labels.
- **Classifier Chains (CC)** (Read et al., 2011) transforms the MLC task into a chain of binary classification problems and takes high-order label correlations into consideration.
- **Label Powerset (LP)** (Tsoumakas and Katakis, 2006) transforms a multi-label problem to a multi-class problem with one multi-class classifier trained on all unique label combinations.
- **CNN** (Kim, 2014) uses multiple convolution kernels to extract text features, which are then inputted to the linear transformation layer followed by a sigmoid function to output the probability distribution over the label space. The multi-label soft margin loss is optimized.
- **CNN-RNN** (Chen et al., 2017) utilizes CNN and RNN to capture both the global and local textual semantics and model the label correlations.

Following the previous work (Chen et al., 2017), we adopt the linear SVM as the base classifier in BR, CC and LP. We implement BR, CC and LP by means of Scikit-Multilearn (Szymański, 2017), an open-source library for the MLC task. We tune hyper-parameters of all baseline algorithms on the validation set based on the micro-F_1 score. In addition, training strategies mentioned in Zhang and Wallace (2015) are used to tune hyper-parameters for the baselines CNN and CNN-RNN.

3.5 Results

For the purpose of simplicity, we denote the proposed sequence generation model as **SGM**. We report the evaluation results of our methods and all baselines on the test sets.

The experimental results of our methods and the baselines on dataset RCV1-V2 are shown in Table 2a. Results show that our proposed methods give the best performance in the main evaluation metrics. Our proposed SGM model using global embedding achieves a reduction of 12.79% hamming-loss and an improvement of 2.33% micro-F_1 score over the most commonly used baseline BR. Besides, our methods outperform other traditional deep-learning models by a large margin. For instance, the proposed SGM model with global embedding achieves a reduction of 15.73% hamming-loss and an improvement

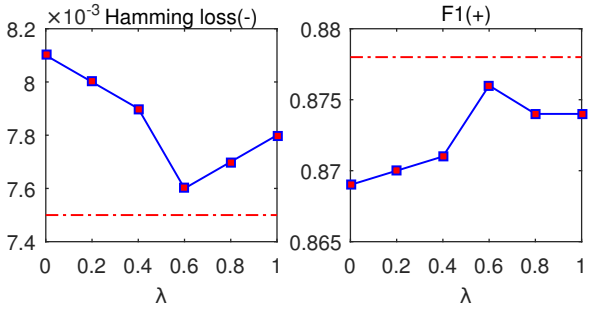

Figure 2: The performance of the SGM model when using different λ. The red dotted line represents the results of using the adaptive gate. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite.

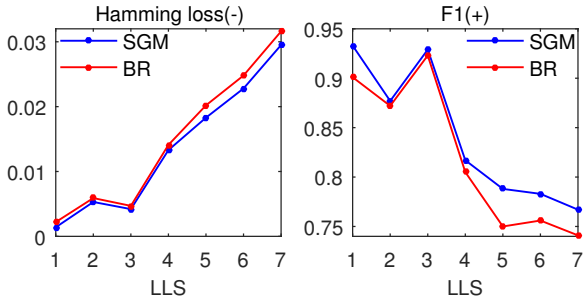

Figure 3: The performance of the SGM model on different subsets of the RCV1-V2 test set. LLS represents the length of label sequence of each sample in the subset. The explanations of symbol "+" and "-" can be found in Figure 2.

of 2.69% micro-F_1 score over the traditional CNN model. Even without the global embedding, our proposed SGM model is still able to outperform all baselines.

In addition, the SGM model is significantly improved by using global embedding. The SGM model with global embedding achieves a reduction of 7.41% hamming loss and an improvement of 1.04% micro-F_1 score on the test set compared with the model without global embedding.

Table 2b presents the results of the proposed methods and the baselines on the AAPD test set. Similar to the experimental results on the RCV1-V2 test set, our proposed methods still outperform all baselines by a large margin in main evaluation metrics. This further confirms that our methods have significant advantages over previous work on large datasets. Besides, the proposed SGM achieves a reduction of 2.39% hamming loss and an improvement of 1.57% micro-F_1 score on the test set by using global embedding. This further testifies that the global embedding is capable of helping the model to predict label sequences more accurately.

3.6 Analysis and Discussion

Here we perform further analysis on the model and experimental results. We report the evaluation results in terms of hamming loss and micro-F_1 score.

3.6.1 Exploration of Global Embedding

As is shown in Table 2, global embedding can significantly improve the performance of the model. The global embedding $g(\boldsymbol{y}_{t-1})$ at time-step t takes advantage of all information of possible labels contained in $\boldsymbol{y}_{t-1}$, so it is able to enrich the source information when the model predicts the current label, which leads to the performance of the model significantly improved. The global embedding is the combination of original embedding $\boldsymbol{e}$ and the weighted average embedding $\bar{\boldsymbol{e}}$ by using the transform gate $\boldsymbol{H}$. Here we conduct experiments on the RCV1-V2 dataset to explore how the performance of our model is affected by the proportion between two kinds of embeddings. In the exploratory experiment, the final embedding vector at time-step t is calculated as follows:

$$g(\boldsymbol{y}_{t-1}) = (1-\lambda) * \boldsymbol{e} + \lambda * \bar{\boldsymbol{e}} \tag{14}$$

The proportion between two kinds of embeddings is controlled by coefficient λ. $\lambda = 0$ denotes the proposed SGM model without global embedding. The proportion of weighted average embedding increases when we increase λ. The experimental results using different λ values in the decoder are shown in Figure 2.

As is shown in Figure 2, the performance of the model varies when different λ is used. Overall, the model using the adaptive gate performs the best, which achieves the best results in both hamming loss and micro-F_1. The models with $\lambda \neq 0$ outperform the model with $\lambda = 0$, which shows that the weighted average embedding contains richer information, leading to the improvement in the performance

Models	HL(-)	F1(+)
SGM	0.0081	0.869
w/o mask	0.0083(↓ 2.47%)	0.866(↓ 0.35%)
w/o sorting	0.0084(↓ 3.70%)	0.858(↓ 1.27%)

(a) Ablation study for the SGM model.

Models	HL(-)	F1(+)
SGM + GE	0.0075	0.878
w/o mask	0.0078(↓ 4.00%)	0.873(↓ 0.57%)
w/o sorting	0.0083(↓ 10.67%)	0.859(↓ 2.16%)

(b) Ablation study for SGM model with global embedding.

Table 3: Ablation study on the RCV1-V2 test set. GE denotes the global embedding. HL and F1 denote hamming loss and micro-F_1, respectively. The symbol "+" indicates that the higher the value is, the better the model performs. The symbol "-" is the opposite. ↑ means that the performance of the model improves and ↓ is the opposite.

of the model. Without using the adaptive gate, the performance of the model improves at first and then deteriorates as λ increases. It reveals the reason why the model with the adaptive gate performs the best: the adaptive gate can automatically determine the most appropriate λ value according to the actual condition.

3.6.2 The Impact of Mask and Sorting

Our proposed methods are developed based on traditional Seq2Seq models. However, the mask module is added to the proposed methods, which is used to prevent the models from predicting repeated labels. In addition, we sort the label sequence of each sample according to the frequency of appearance of labels in the training set. In order to explore the impact of the mask module and sorting, we conduct ablation experiments on the RCV1-V2 dataset. The experimental results are shown in Table 3. "*w/o mask*" means that we do not perform mask operation and "*w/o sorting*" means that we randomly shuffle the label sequence in order to perturb its original order.

As is shown in Table 3, the performance decline of the SGM model with global embedding is more significant compared with that of the SGM model without global embedding. In addition, the decline in the performance of the two models is more significant when we randomly shuffle the label sequence of the sample compared with removing mask module. The label cardinality of the RCV1-V2 dataset is small, so our proposed methods are less prone to predicting repeated labels. This explains the reason why experimental results indicate that the mask module has little impact on the models' performance. In addition, the proposed models are trained using the maximum likelihood estimation method and the cross-entropy loss function, which requires humans to predefine the order of the output labels. Therefore, the sorting of labels is very important for the models' performance. Besides, the performance of both models declines when we do not use the mask module. This shows that the performance of the model can be improved by using the mask operation.

3.6.3 Error Analysis

In the experiment, we find that the performance of all methods deteriorates when the length of the label sequence increases (for simplicity, we denote the length of the label sequence as *LLS*). In order to explore the influence of the value of the *LLS*, we divide the test set into different subsets based on different *LLS*. Figure 3 shows the performance of the SGM model and the most commonly used baseline BR on different subsets of the RCV1-V2 test set. As is shown in Figure 3, generally, the performance of both models deteriorates as the *LLS* increases. This shows that when the label sequence of the sample is particularly long, it is difficult to accurately predict all labels. Because more information is needed when the model predicts more labels. It is easy to ignore some true labels whose feature information is insufficient.

However, as is shown in Figure 3, the proposed SGM model outperforms BR with any value of *LLS*, and the advantages of our model are more significant when *LLS* is large. The traditional BR method predicts all labels at once only based on the sample input. Therefore, it tends to ignore some true labels whose feature information contained in the sample is insufficient. The SGM model generates labels sequentially, and predicts the next label based on its previously predicted labels. Therefore, even if the sample contains less information of some true labels, the SGM model is capable of generating these true labels by considering relevant labels that have been predicted.

• Generating descriptions for videos has many applications including human robot interaction.
• Many methods for image captioning rely on pre-trained object classifier CNN and Long Short Term Memory recurrent networks.
• How to learn robust visual classifiers from the weak annotations of the sentence descriptions.

(a) Visual analysis when the SGM model predicts "*CV*".

• Generating descriptions for videos has many applications including human robot interaction.
• Many methods for image captioning rely on pre-trained object classifier CNN and Long Short Term Memory recurrent networks.
• How to learn robust visual classifiers from the weak annotations of the sentence descriptions.

(b) Visual analysis when the SGM model predicts "*CL*".

Table 4: An example abstract in the AAPD dataset, from which we extract three informative sentences. This abstract is assigned two labels: "*CV*" and "*CL*". They denote computer vision and computational language, respectively.

Reference	**BR**	**SGM**	**SGM + GE**
CCAT, **C15, C152**, C41, C411	CCAT, C15, C13	CCAT, **C15, C152**	CCAT, **C15, C152**, C41, C411
CCAT, GCAT, ECAT, C31, GDIP, C13, C21, **E51, E512**	CCAT, GCAT, GDIP, E51	CCAT, ECAT, GDIP, **E51, E512**	CCAT, GCAT, ECAT, C31, GDIP, **E51, E512**, C312
GCAT, ECAT, **G15, G154, G151, G155**	GCAT, ECAT, GENV, G15	GCAT, ECAT, E21, **G15, G154, G156**	GCAT, ECAT, E21, **G15, G154, G155**

Table 5: Several examples of the generated label sequences on the RCV1-V2 dataset. The red bold labels in each example indicate that they are highly correlated.

3.6.4 Visualization of Attention

When the model predicts different labels, there exist differences in the contributions of different words. The SGM model is able to select the most informative words by utilizing the attention mechanism. The visualization of the attention layer is shown in Table 4. According to Table 4, when the SGM model predicts the label "*CV*", it can automatically assign larger weights to more informative words, like `image`, `visual`, `captioning`, and so on. For the label "*CL*", the selected informative words are `sentence`, `memory`, `recurrent`, etc. This shows that our proposed models are able to consider the differences in the contributions of textual content when predicting different labels and select the most informative words automatically.

3.6.5 Case Study

We give several examples of the generated label sequences on the RCV1-V2 dataset in Table 5, where we compare the proposed methods with the most commonly used baseline BR. The red bold labels in each example indicate that they are highly correlated. For instance, the correlation coefficient between E51 and E512 is 0.7664. Therefore, these highly correlated labels are likely to appear together in the predicted label sequence. The BR algorithm fails to capture this label correlation, leaving many true labels unpredicted. However, our proposed methods accurately predict almost all highly correlated true labels. The proposed SGM captures the correlations between labels by utilizing LSTM to generate labels sequentially. Therefore, for some true labels whose feature information is insufficient, the proposed SGM is still able to generate them by considering relevant labels that have been predicted. In addition, label sequences that are more accurate are predicted by using global embedding. The SGM model with global embedding predicts more true labels compared with the SGM model without global embedding. The reason is that the source information is further enriched by incorporating overall informative signals in the probability distribution $\boldsymbol{y}_{t-1}$ when the model predicts the label at time-step t. Enriched information makes global embedding more smooth, which enables the model to reduce damage caused by mispredictions made in the previous time steps.

4 Related Work

The MLC task studies the problem where multiple labels are assigned to each sample. There are four main types of methods for the MLC task: problem transformation methods, algorithm adaptation methods, ensemble methods, and neural network models.

Problem transformation methods map the MLC task into multiple single-label learning tasks. Binary relevance (BR) (Boutell et al., 2004) decomposes the MLC task into independent binary classification problems by ignoring the correlations between labels. In order to model label correlations, label power-set (LP) (Tsoumakas and Katakis, 2006) transforms a multi-label problem to a multi-class problem with a classifier trained on all unique label combinations. Classifier chains (CC) (Read et al., 2011) transforms the MLC task into a chain of binary classification problems, where subsequent binary classifiers in the chain are built upon the predictions of preceding ones. However, the computational efficiency and performance of these methods are challenged by applications with a large number of labels and samples.

Algorithm adaptation methods extend specific learning algorithms to handle multi-label data directly. Clare and King (2001) construct decision tree based on multi-label entropy to perform classification. Elisseeff and Weston (2002) optimize the empirical ranking loss by using maximum margin strategy and kernel tricks. Collective multi-label classifier (CML) (Ghamrawi and McCallum, 2005) adopts maximum entropy principle to deal with multi-label data by encoding label correlations as constraint conditions. Zhang and Zhou (2007) adopt k-nearest neighbor techniques to deal with multi-label data. Fürnkranz et al. (2008) make ranking among labels by utilizing pairwise comparison. Li et al. (2015) propose a novel joint learning algorithm that allows the feedbacks to be propagated from the classifiers for latter labels to the classifier for the current label. Most methods, however, can only be used to capture the first or second order label correlations or are computationally intractable in considering high-order label correlations.

Among ensemble methods, Tsoumakas et al. (2011) break the initial set of labels into a number of small random subsets and employ the LP algorithm to train a corresponding classifier. Szymański et al. (2016) propose to construct a label co-occurrence graph and perform community detection to partition the label set.

In recent years, some neural network models have also been used for the MLC task. Zhang and Zhou (2006) propose the BP-MLL that utilizes a fully-connected neural network and a pairwise ranking loss function. Nam et al. (2013) propose a neural network using cross-entropy loss instead of ranking loss. Benites and Sapozhnikova (2015) increase classification speed by adding an extra ART layer for clustering. Kurata et al. (2016) utilize word embeddings based on CNN to capture label correlations. Chen et al. (2017) propose to represent semantic information of text and model high-order label correlations by combining CNN with RNN. Baker and Korhonen (2017) initialize the final hidden layer with rows that map to co-occurrence of labels based on the CNN architecture to improve the performance of the model. Ma et al. (2018) propose to use the multi-label classification algorithm for machine translation to handle the situation where a sentence can be translated into more than one correct sentences.

5 Conclusions and Future Work

In this paper, we propose to view the multi-label classification task as a sequence generation problem to model the correlations between labels. A sequence generation model with a novel decoder structure is proposed to improve the performance of classification. Extensive experimental results show that the proposed methods outperform the baselines by a substantial margin. Further analysis of experimental results demonstrates that our proposed methods not only capture the correlations between labels, but also select the most informative words automatically when predicting different labels.

As analyzed in Section 3.6.3, when a large number of labels are assigned to a sample, how to predict all these true labels accurately is an intractable problem. Our proposed methods alleviate this problem to some extent, but more effective solutions need to be further explored in the future.

6 Acknowledgements

This work is supported in part by National Natural Science Foundation of China (No. 61673028, No. 61333018) and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Baker and Korhonen2017] Simon Baker and Anna Korhonen. 2017. Initializing neural networks for hierarchical multi-label text classification. In *BioNLP*.

[Benites and Sapozhnikova2015] Fernando Benites and Elena Sapozhnikova. 2015. Haram: a hierarchical aram neural network for large-scale text classification. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 847–854. IEEE.

[Boutell et al.2004] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. 2004. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771.

[Chen et al.2017] Guibin Chen, Deheng Ye, Zhenchang Xing, Jieshan Chen, and Erik Cambria. 2017. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 2377–2383.

[Clare and King2001] Amanda Clare and Ross D King. 2001. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer.

[Elisseeff and Weston2002] André Elisseeff and Jason Weston. 2002. A kernel method for multi-labelled classification. In *Advances in neural information processing systems*, pages 681–687.

[Fürnkranz et al.2008] Johannes Fürnkranz, Eyke Hüllermeier, Eneldo Loza Mencía, and Klaus Brinker. 2008. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153.

[Ghamrawi and McCallum2005] Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200. ACM.

[Gopal and Yang2010] Siddharth Gopal and Yiming Yang. 2010. Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 315–322. ACM.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Katakis et al.2008] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

[Kingma and Ba2014] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Kurata et al.2016] Gakuto Kurata, Bing Xiang, and Bowen Zhou. 2016. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 521–526.

[Lewis et al.2004] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

[Li et al.2015] Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 835–839.

[Lin et al.2018] Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. In *ACL 2018*.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

[Ma et al.2018] Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *ACL 2018*.

[Manning et al.2008] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

[Nam et al.2013] Jinseok Nam, Jungi Kim, Iryna Gurevych, and Johannes Fürnkranz. 2013. Large-scale multi-label text classification - revisiting neural networks. *CoRR*, abs/1312.5419.

[Pascanu et al.2013] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

[Read et al.2011] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Machine learning*, 85(3):333.

[Rush et al.2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 379–389.

[Schapire and Singer1999] Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

[Schapire and Singer2000] Robert E Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.

[Shen et al.2017] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. *CoRR*, abs/1705.09655.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

[Srivastava et al.2015] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

[Sun et al.2017] Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017. Label embedding network: Learning label representation for soft training of deep networks. *CoRR*, abs/1710.10393.

[Szymański et al.2016] Piotr Szymański, Tomasz Kajdanowicz, and Kristian Kersting. 2016. How is a data-driven approach better than random choice in label space division for multi-label classification? *Entropy*, 18(8):282.

[Szymański2017] Piotr Szymański. 2017. A scikit-based python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460*.

[Tsoumakas and Katakis2006] Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3).

[Tsoumakas et al.2011] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2011. Random k-labelsets for multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 23(7):1079–1089.

[Wiseman and Rush2016] Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *CoRR*, abs/1606.02960.

[Xu et al.2018] Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *ACL 2018*.

[Zhang and Wallace2015] Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820.

[Zhang and Zhou2006] Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351.

[Zhang and Zhou2007] Min-Ling Zhang and Zhi-Hua Zhou. 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048.