Order-Free Learning Alleviating Exposure Bias in Multi-Label Classification

Che-Ping Tsai, Hung-Yi Lee
Speech Processing and Machine Learning Laboratory,
National Taiwan University*
{r06922039, hungyilee}@ntu.edu.tw

Abstract

Multi-label classification (MLC) assigns multiple labels to each sample. Prior studies show that MLC can be transformed to a sequence prediction problem with a recurrent neural network (RNN) decoder to model the label dependency. However, training a RNN decoder requires a predefined order of labels, which is not directly available in the MLC specification. Besides, RNN thus trained tends to overfit the label combinations in the training set and have difficulty generating unseen label sequences. In this paper, we propose a new framework for MLC which does not rely on a predefined label order and thus alleviates exposure bias. The experimental results on three multi-label classification benchmark datasets show that our method outperforms competitive baselines by a large margin. We also find the proposed approach has a higher probability of generating label combinations not seen during training than the baseline models. The result shows that the proposed approach has better generalization capability.

Introduction

Multi-label classification (MLC) is a fundamental but challenging problem in machine learning with applications such as text categorization (Yang et al. 2018b), sound event detection (Gemmeke et al. 2017; Yu et al. 2018), and image classification (Tsai and Lee 2018). In contrast to single-label classification, multi-label predictors must not only relate labels with the corresponding instances, but also exploit the underlying label structures. Take for instance the text classification dataset RCV1 (Lewis et al. 2004), which uses a hierarchical tree structure between labels.

Recent studies show that MLC can be transformed to a sequence prediction problem by probabilistic classifier chains (PCC)(Read et al. 2011; Cheng, Hüllermeier, and Dembczynski 2010; Dembczynski, Waegeman, and Hüllermeier 2012). PCC models the joint probabilities of output labels with the use of the chain rule and predicts labels based on previously generated output labels. Furthermore, PCC can be replaced by a RNN decoder to model label correlation. Wang et al. (2016) propose the CNN-RNN architecture to capture both image-label relevance and semantic label dependency in multi-label image classification. Nam et al. (2017) and Yang et al. (2018b) show that state-of-the-art multi-label text classification results can be achieved by using a sequence-to-sequence (seq2seq) architecture to encode input text sequences and decode labels sequentially.

However, this kind of RNN-based decoder suffers from several problems. First, these models are trained using maximum likelihood estimation (MLE) on target label sequences, which relies on a predefined ordering of labels. Previous studies (Vinyals, Bengio, and Kudlur 2016; Yang et al. 2018a) show that ordering has a significant impact on the performance. This issue also appears in the PCC, It is addressed by ensemble averaging (Read et al. 2011; Cheng, Hüllermeier, and Dembczynski 2010), ensemble pruning (Li and Zhou 2013), pre-analysis of the label dependencies by Bayes nets (Sucar et al. 2014) and integrating beam search with training to determine a suitable tag ordering (Kumar et al. 2013). However, these approaches rely on training multiple models to ensemble or determine a proper order, which is computationally expensive.

Although Nam et al. (2017) and Yang et al. (2018b) compare several ordering strategies and suggest ordering positive labels by frequency directly in descending order (from frequent to rare labels), it is unnatural to impose a strict order on labels, which may break down label correlations in a chain. Furthermore, we find that this kind of model tends to overfit to label combinations and shows poor generalization ability.

Second, during training, the RNN-based models are always conditioned on correct prefixes; during inference, however, the prefixes are generated by the RNN-based model, yielding a problem known as *exposure bias* (Ranzato et al. 2015) in seq2seq learning. The error may propagate as the model might be in a part of the state space that it has not seen during training (Senge, Del Coz, and Hüllermeier 2014).

In this paper, we propose a novel learning algorithm for RNN-based decoders on multi-label classification not rely on a predefined label order. The proposed approach is inspired by optimal completion distillation (OCD) (Sabour, Chan, and Norouzi 2018), a training procedure for optimiz-

*This work was financially supported by the Ministry of Science and Technology of Taiwan.

ing seq2seq models. In this algorithm, we feed the RNN decoder generated tokens by sampling from the current model. Hence, the model may encounter different label orders and wrong prefixes during training, so the model explores more, and exposure bias is alleviated.

Another common and straightforward way to avoid the need for ordered labels in MLC is binary relevance (BR) (Tsoumakas and Katakis 2007), which decomposes MLC into multiple independent single-label binary classification problems. However, this yields a model that cannot take advantage of label co-occurrences. In this paper, we further propose helping the model to learn better by use of an auxiliary binary relevance (BR) decoder jointly trained with the RNN decoder within a multitask learning (MTL) framework.

In addition, at the inference stage, the predictions of the BR decoder can be jointly combined in the RNN decoder to improve performance further. We propose two methods to combine their probabilities. Extensive experiments show that the proposed model outperforms competitive baselines by a large margin on three multi-label classification benchmark datasets, including two text classification and one sound event classification datasets.

The contributions of this paper are as follows:

- We propose a novel training algorithm for multi-label classification which predicts labels autoregressively but does not require a predefined label order for training.
- We compare our methods with competitive baseline models on three multi-label classification datasets and demonstrate the effectiveness of the proposed models.
- We systematically analyze the problem of exposure bias and the effectiveness of scheduled sampling (Bengio et al. 2015) in multi-label classification.

Related work

RNN-based multi-label classification

To free the RNN-based MLC classifier from a predefined label order, Chen et al. (2017) proposes the order-free RNN to dynamically decide a target label at each time during training by choosing the label in the target label set with the highest predicted probability; hence, the model learns a label order by itself. Although the order can be modified during training, this approach still needs an initialized label order to start the training process. We find order-free RNN shows poor generalization ability to unseen label combinations in the experiments. Also, as the model is always supplied with the correct labels, it suffers from exposure bias.

To handle both exposure bias and label order, other studies apply a reinforcement learning (RL) algorithm to MLC. He et al. (2018) apply an off-policy Q learning algorithm to multi-label image classification. Yang et al. (2018a) uses two decoders to solve multi-label text classification, one of which is trained with MLE and the other is trained with a self-critical policy gradient training algorithm. However, Q learning and policy gradients cannot easily incorporate ground truth sequence information, except via the reward function, as the model is rewarded only at the end of each episode. Indeed, He et al. (2018) does not work without pretraining on the target dataset. By contrast, we use optimal completion distillation (OCD) (Sabour, Chan, and Norouzi 2018) for MLC, which optimizes token-level log-loss, where the training is stabilized and requires neither initialization nor joint optimization with MLE.

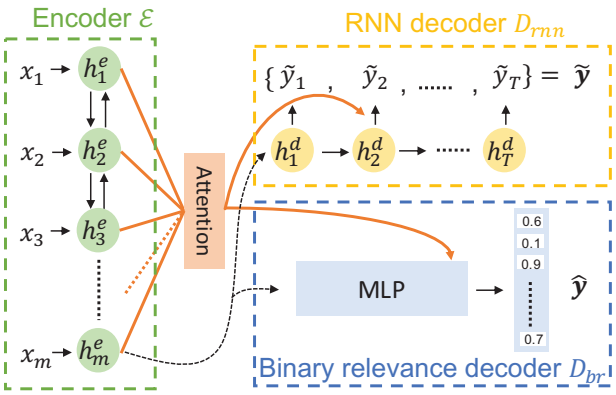

Figure 1: Overview of proposed model. The model is composed of three components: encoder $\mathcal{E}$, RNN decoder $\mathcal{D}_{rnn}$, and binary relevance decoder $\mathcal{D}_{br}$. $\tilde{\mathbf{y}} = \{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_T\}$ represents the sampled sequence from $\mathcal{D}_{rnn}$, while $\hat{y}$ is a vector representing the predicted probabilities of labels by $\mathcal{D}_{br}$.

Optimal Completion Distillation (OCD)

Our work is inspired by OCD (Sabour, Chan, and Norouzi 2018), which was first used in the context of end-to-end speech recognition in which it achieved state-of-the-art performance. In contrast to MLE, OCD algorithms encourage the model to extend all possible tokens that lead to the optimal edit distance by assigning equal probabilities to the target policy that the model learns from. We use OCD to train the RNN decoder in MLC. The OCD training details for MLC are in section *Learning for RNN decoder* $\mathcal{D}_{rnn}$. In contrast to the original OCD (Sabour, Chan, and Norouzi 2018) which optimizes the edit distance, in the proposed approach we optimize the numbers of missing and false alarm labels.

Model architecture

An overview of the proposed model is shown in Fig. 1. The model is composed of three components: encoder $\mathcal{E}$, RNN decoder $\mathcal{D}_{rnn}$, and binary relevance decoder $\mathcal{D}_{br}$. Here, multi-label text classification is considered an instance of MLC. For other types of MLC other than text classification (e.g. sound event classification), the architecture of the encoder can be changed.

Encoder $\mathcal{E}$

We employ a bidirectional LSTM as an encoder $\mathcal{E}$. The encoder reads the input text sequence $\mathbf{x} = x_1, x_2, ..., x_m$ of m words in both forward and backward directions and computes the hidden states $h_1^e, h_2^e, ...h_m^e$ for each word.

$$h_1^e, h_2^e, ...h_m^e = \text{BiLSTM}(x_1, x_2, ..., x_m) \quad (1)$$

RNN decoder $\mathcal{D}_{rnn}$

The RNN decoder seeks to predict labels sequentially. It is potentially more powerful than the binary relevance decoder because each prediction is determined based on the previous prediction: thus it implicitly learns label dependencies. We implement it using LSTMs with an attention mechanism. Hence, the encoder and RNN decoder form a seq2seq model. In particular, we set the initial hidden state of the decoder $h_0^d = h_m^e$ and calculate the hidden state h_t^d and output o_t at time t as

$$h_t^d, o_t = \text{LSTM}(h_{t-1}^d, \tilde{y}_{t-1}) \tag{2}$$

where $\tilde{y}_{t-1}$ is the predicting label at previous timestep. $\tilde{y}_t$ is estimated with $p_{rnn}(\tilde{y}_t|\tilde{\mathbf{y}}_{<t}, \mathbf{x})$ by the following equations:

$$p_{rnn}(\tilde{y}_t|\tilde{\mathbf{y}}_{<t}, \mathbf{x}) = \text{softmax}(o_t) \tag{3}$$

$$\tilde{y}_t \sim \text{softmax}(o_t + M_t), \tag{4}$$

During sampling, we add a mask vector $M_t \in \mathcal{R}^L$ (Yang et al. 2018b) to prevent the model from predicting repeated labels, where L is the number of labels in the dataset:

$$(M_t)_j = \begin{cases} -\infty & \text{if the } j\text{-th label has been predicted} \\ & \text{before step } t. \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Binary relevance decoder $\mathcal{D}_{br}$

The binary relevance (BR) decoder $\mathcal{D}_{br}$ here is an auxiliary decoder to train the encoder within the multitask learning (MTL) framework. The BR decoder predicts each label separately as a binary classifier for each label, helping the model to learn better. Another advantage of using the BR decoder is that we can consider the predictions of both the RNN and BR decoders to further improve performance.

In particular, we feed the final hidden state of encoder h_m^e to a DNN with a final prediction layer of size L with sigmoid activation functions to predict the probabilities of each label. To take into account vanishing gradients for long input sequences, we add another attention module. In particular, we calculated the context vector c^{br} in the attention mechanism with the output of fully-connected layers $\text{MLP}(h_m^e)$ and then compute probabilities $p_{br}(\tilde{y}|\mathbf{x})$ as

$$p_{br}(\hat{y}|\mathbf{x}) = \text{sigmoid}(W_{br}[\text{MLP}(h_m^e); c^{br}]), \tag{6}$$

where W_{br} is the matrix of weight parameters and $[\text{MLP}(h_m^e); c^{br}]$ indicates the concatenation of $\text{MLP}(h_m^e)$ and c^{br}.

Order-Free Training

In this section, we derive the training objective for the RNN decoder , the BR decoder , and the multitask learning objective.

Learning for RNN decoder $\mathcal{D}_{rnn}$

RNN decoder learning as RL To reduce exposure bias and free the model from relying on a predefined label order, we never train on ground truth target sequences. Instead, we approach the MLC problem from an RL perspective. The model here plays the role of an *agent* whose action a_t is the current generated label at time t and whose *state* s_t is the output labels $\tilde{\mathbf{y}}_{<t}$ before time t. The policy $\pi(s)$ is a probability distribution over actions a given states s. Once the process is ended with an *end-of-sentence* token, the agent is given a *reward* R.

In our approach, reward R is defined as

$$R(\mathbf{y}^*, \tilde{\mathbf{y}}) = -|\{\mathbf{y}^*\} \setminus \{\tilde{\mathbf{y}}\}| - |\{\tilde{\mathbf{y}}\} \setminus \{\mathbf{y}^*\}|, \tag{7}$$

where $\mathbf{y}^*$ and $\tilde{\mathbf{y}}$ are the ground truth labels and the sequence of labels generated by the RNN decoder, and $B \setminus A$ is the relative complement of set A in set B.

The first and second term of reward $R(\mathbf{y}^*, \tilde{\mathbf{y}})$ are the number of labels that were not predicted and the number of misclassified labels, respectively.

Optimal completion distillation However, typical RL algorithms, such as Q learning and policy gradients, cannot easily incorporate ground truth sequence information except via the reward function. Here we introduce optimal Q-values, which evaluates each action a_t at each time t.

Optimal Q-values $Q^*(s, a)$ represents the maximum total reward the agent can acquire after taking action a at state s via subsequently conducting the optimal action sequence. Optimal Q-values at time t can be expressed as

$$Q^*(\tilde{\mathbf{y}}_{<t}, a) = \max_{\mathbf{y}_{opt} \in \mathcal{Y}} R(\mathbf{y}^*, [\tilde{\mathbf{y}}_{<t}, a, \mathbf{y}_{opt}]). \tag{8}$$

where $[\tilde{\mathbf{y}}_{<t}, a, \mathbf{y}_{opt}]$ is a complete sequence, which is the concatenation of token sequence generated before time t, action at time t and optimal subsequent action sequence $\mathbf{y}_{opt}$.

Optimal policy at time t can be calculated by taking a softmax over optimal Q-values of all the possible actions. Formally,

$$\pi^*(a|\tilde{\mathbf{y}}_{<t}) = \frac{\exp(Q^*(\tilde{\mathbf{y}}_{<t}, a)/\tau)}{\sum_{a'} \exp(Q^*(\tilde{\mathbf{y}}_{<t}, a')/\tau)}, \tag{9}$$

where $\tau \geq 0$ is a temperature parameter. If τ is close to 0, π^* is a hard target. Table 1 shows an example illustrating the optimal policy in OCD training procedure.

Given a dataset $\{\mathbf{x}, \mathbf{y}^*\}$, we first draw generated sequences from the RNN decoder $\tilde{\mathbf{y}} \sim p_{rnn}(\cdot|\mathbf{x})$ by sampling. The loss function $\mathcal{L}_{OCD}$ can be obtained via calculating KL divergence between the optimal policy and the model's generated distribution over labels at every time step t,

$$\begin{aligned} \mathcal{L}_{OCD} &= \mathbb{E}_{(\mathbf{x}, \mathbf{y}^*) \sim data} \mathbb{E}_{\tilde{\mathbf{y}} \sim p_{rnn}(\cdot|\mathbf{x})} [\\ &\sum_{t=1}^{|\tilde{\mathbf{y}}|} \text{KL}(\pi^*(\cdot|\tilde{\mathbf{y}}_{<t}) || p_{rnn}(\cdot|\tilde{\mathbf{y}}_{<t}, \mathbf{x}))]. \end{aligned} \tag{10}$$

The above equation means we "distill" knowledge from optimal policies obtained by completing with optimal action sequences to RNN decoder p_{rnn}, so RNN decoder can have similar behaviour as optimal policy π^*.

In contrast to MLE, OCD encourages the model to extend all the possible targets resulting in the same evaluation metric score. Therefore, the OCD objective focuses on all labels that were not predicted and assigns them equal probabilities.

Once all the target labels are successfully generated, the objective guides the model to produce the end-of-sentence token with probability 1.

Since the OCD targets depend only on the tokens generated previously, we do not need a human-defined label order to train the RNN decoder. The label order is instead automatically determined at each time step. In addition, we always train on sequences generated from the current model, thus alleviating exposure bias.

Note that we can substitute the reward function (Eq. 7) with other example-based test metrics such as the example-based F1 score, but these lead to the same OCD targets as the rewards of all the target labels are the same.

Time t	OCD targets	Optimal policy π^* ($\tau \to 0$)	Prediction $\tilde{y}_t$
0	A, B, D	$[\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3}, 0]$	B
1	A, D	$[\frac{1}{2}, 0, 0, \frac{1}{2}, 0]$	C
2	A, D	$[\frac{1}{2}, 0, 0, \frac{1}{2}, 0]$	A
3	D	$[0, 0, 0, 1, 0]$	D
4	<eos>	$[0, 0, 0, 0, 1]$	<eos>

Table 1: A training example of optimal completion distillation, where there are 4 kinds of labels (A, B, C, D) and an end-of-sentence token (<eos>). Labels A, B, and D are the targets of this instance and the vectors of π^* represent probabilities for labels A, B, C, D and <eos>, respectively. We set $\tau \to 0$ here, so the optimal policy only encourages labels with the highest optimal Q values. For example, at time 1, since the model has predicted correct token B at time 0, there are two optimal extended tokens {A, D}, which result in a total reward of 0 (Eq. 7) when combined with proper suffixes. Then we sample from the current policy and predict the incorrect token C, which leads to a decreased optimal possible reward of -1 (Eq. 7) at time 2.

Learning for binary relevance decoder $\mathcal{D}_{br}$

For the binary relevance decoder, given the ground truth $\mathbf{y}^*_{vec} = [y^*_{vec,1}, y^*_{vec,2} ..., y^*_{vec,L}]^T \in \{0,1\}^L$ in binary format, we use binary cross-entropy loss as the objective:

$$\mathcal{L}_{logistic} = \mathbb{E}_{(\mathbf{x}, \mathbf{y}^*_{vec}) \sim data} [\sum_{i=1}^{L} y^*_{vec,i} log\, \hat{y}_i + (1 - y^*_{vec,i}) log(1 - \hat{y}_i)], \quad (11)$$

where $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2 ..., \hat{y}_L]^T$, which is a vector of length L representing the predicted probability of each label.

Multitask Learning (MTL)

The objective of MTL is

$$\mathcal{L}_{MTL} = \mathcal{L}_{OCD} + \lambda \mathcal{L}_{logistic}, \quad (12)$$

where λ is a weight.

Decoder Integration

In this section, we seek to utilize both the RNN and BR decoders to find the optimal hypothesis $\mathcal{H} = \{l_1, l_2, ..., l_T\}$, which consists of $T-1$ predicted labels $\{l_1, l_2, ..., l_{T-1}\}$, where l_T is the end-of-sentence token indicating that the decoding process of the RNN decoder has ended.

For the BR decoder, the outputs after a sigmoid activation $p_{br}(y_l = 1|\mathbf{x}) = \hat{y}_l$ are designed to estimate the posterior probabilities of each label l. Therefore, the theoretically optimal threshold for converting the probability to a binary value should be 0.5, which is equivalent to finding the optimal hypothesis $\mathcal{H}$ that maximizes the Eq. 13 below, which is the product of the probabilities of all the labels.

$$P_{br}(\mathcal{H}) = \prod_{l \in \mathcal{H}} p_{br}(y_l = 1|\mathbf{x}) \times \prod_{l \notin \mathcal{H}} p_{br}(y_l = 0|\mathbf{x}) \quad (13)$$

For the RNN decoder, a typical inference step is performed with a beam search to solve Eq. 14 (Yang et al. 2018b; Chen et al. 2017). Given input x, the probability of the predicted hypothesis path $\mathcal{H}$ is

$$P_{path}(\mathcal{H}) = \prod_{i=1}^{i=T} p_{rnn}(l_i|\mathbf{x}, l_1, ...l_{i-1}). \quad (14)$$

To combine the predictions of the RNN and BR decoders, we simply take the product of Eq. 13 and Eq. 14 to yield the final objective function Eq. 15:

$$\hat{\mathcal{H}} = \arg\max_{\mathcal{H}} \{P_{path}(\mathcal{H}) \times P_{br}(\mathcal{H})\}. \quad (15)$$

Nonetheless, the equation is not easy to solve because the RNN decoder outputs the probability of selecting a particular label at each time step while the BR decoder produces the unconditional probabilities of all the labels at once. To incorporate the BR probabilities of labels in the score, we provide two different decoding strategies to find the best hypothesis $\hat{\mathcal{H}}$.

Logistic Rescoring

In this method, we first obtain a set of complete hypotheses using beam search only with the RNN decoder, and rescore each hypothesis $\mathcal{H}$ using the probabilities produced by the BR decoder with Eq. 13. Finally, we select as the final prediction the hypothesis $\mathcal{H}$ with the highest $P_{br}(\mathcal{H}_{best})$.

Logistic Joint Decoding

This method is one-pass decoding. We conduct a beam search according to the following equation (see the derivation in Appendix).

$$P_{joint}(\mathcal{H}) = \prod_{i=1}^{i=T} p_{rnn}(l_i|\mathbf{x}, l_1, ...l_{i-1}) \frac{p_{br}(y_{l_i} = 1|\mathbf{x})}{p_{br}(y_{l_i} = 0|\mathbf{x})} \quad (16)$$

Note that we manually set the probability of the end-of-sentence token l_T for binary relevance $p_{br}(y_{l_T} = 0|\mathbf{x}) = p_{br}(y_{l_T} = 1|\mathbf{x}) = 0.5$, since it does not exist in the outputs of the BR decoder.

Models		maF1	miF1	ebF1	ACC	HA	Average
(a) Seq2set (simp.) (Yang et al. 2018a)		-	0.705	-	-	0.9753	-
(b) Seq2set (Yang et al. 2018a)		-	0.698	-	-	0.9751	-
(c) SGM+GE (Yang et al. 2018b)		-	0.710	-	-	0.9755	-
Baselines							
(d) BR		0.523	0.694	0.695	0.368	0.9741	0.651
(e) BR++		0.521	0.700	0.703	0.390	0.9750	0.658
(f) Seq2seq		0.511	0.695	0.707	**0.421**	0.9743	0.662
(g) Seq2seq + SS		0.541	0.703	0.713	0.406	0.9742	0.667
(h) Order-free RNN		0.539	0.696	0.708	0.413	0.9742	0.666
(i) Order-free RNN + SS		0.548	0.699	0.709	0.416	0.9743	0.669
Proposed methods							
(j) OCD		0.541	0.707	0.723	0.403	0.9740	0.670
OCD + MTL	(k) RNN dec.	0.578	0.711	0.727	0.391	0.9742	0.676
	(l) BR dec.	0.562	0.711	0.718	0.382	**0.9760**	0.670
	(m) Logistic rescore	**0.585**	**0.720**	**0.736**	0.395	0.9749	**0.682**
	(n) Logistic joint dec.	0.580	0.719	0.731	0.399	0.9753	0.681

Table 2: Performance on AAPD

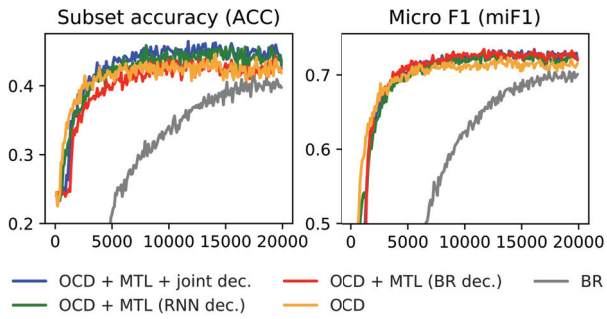

Figure 2: Performance of BR, OCD and MTL models on AAPD validation set with different decoding strategies during training. The x-axis denotes the number of updates; we use different scales on the y-axis for each measure.

Models		maF1	miF1	ebF1	ACC	HA	Average
SVM (Debole et al. 2005)		0.468	0.787	-	-	-	-
EncDec (Nam et al. 2017)		0.457	0.855	0.891	0.828	0.996	0.805
Baselines							
BR		0.442	0.861	0.878	0.817	0.9964	0.799
BR++		0.407	0.852	0.861	0.812	0.9962	0.786
Seq2seq		0.465	0.862	0.895	0.834	0.9965	0.811
Seq2seq+SS		0.464	0.856	0.895	0.834	0.9965	0.809
Order-free RNN		0.445	0.862	0.901	0.835	0.9963	0.806
Order-free RNN + SS		0.452	0.859	0.896	0.836	0.9962	0.808
Proposed methods							
OCD		0.458	0.872	0.903	0.839	0.9966	0.814
OCD + MTL	RNN dec.	0.475	0.874	**0.905**	**0.844**	0.9966	0.819
	BR dec.	0.459	**0.877**	0.898	0.835	0.9966	0.813
	Logistic rescore	0.477	0.875	0.903	0.842	**0.9967**	0.819
	Logistic joint dec.	**0.490**	0.874	0.904	0.843	**0.9967**	**0.822**

Table 3: Performance comparisons on Reuters-21578

Experimental Setup

We validate our proposed model on two multi-label text classification datasets, which are AAPD (Yang et al. 2018b) and Reuters-21758, and a sound event classification dataset, which is Audio set (Gemmeke et al. 2017) proposed by Google. They span a wide variety in terms of the number of samples, the number of labels, and the number of words per sample. Due to space limit, we put an extra experiment in text classification on RCV1-V2 (Lewis et al. 2004), data statistics, experimental settings and the descriptions of five evaluation metrics in Appendix.

Evaluation Metrics

Multi-label classification can be evaluated with multiple metrics, which capture different aspects of the problem. We follow Nam et al. (2017) in using five different metrics: subset accuracy (ACC), Hamming accuracy (HA), example-based F1 (ebF1), macro-averaged F1 (maF1), and micro-averaged F1 (miF1).

Baselines

We compare our methods with the following baselines. For a fair comparison, the architecture of all the encoders are the same except for BR++: the RNN decoders for Seq2seq, Order-free RNN, and the proposed approaches are the same.

- **Binary Relevance (BR)** is the model trained with logistic loss (Eq. 11), and consists of an encoder and a BR decoder.
- **Binary Relevance++ (BR++)** is a model with a larger encoder but with the same training algorithm as BR. Because the MTL model has more parameters than BR, for fair comparison, we increase the number of layers in the encoder RNN so that the number of parameters is approximately equal to the MTL model. [1]
- **Seq2seq** (Nam et al. 2017) is composed of an RNN encoder and an RNN decoder with an attention mechanism. The model is trained with MLE. The target label sequences are ordered from frequent to rare, which yields better performance (Nam et al. 2017; Yang et al. 2018b). [2]
- **Order-free RNN** (Chen et al. 2017) uses an algorithm for multi-label image classification to train the RNN decoder without predefined orders but suffers from exposure bias.

Since scheduled sampling also tackles the problem of exposure bias, we also compare the performance of seq2seq and order-free RNN with scheduled sampling (SS), which are **Seq2seq + SS** and **Order-free RNN + SS**. The detailed discussion of the effectiveness of scheduled sampling is in section *Disussion.*

Results and Discussion

In the following, we show results of the baseline models and the proposed method on three text datasets. For MTL models, we show the results of the four kinds of different decoding strategies described in section *Decoder Integration*. For a simple comparison, we also compute averages of the

[1] Since Yu et al. (2018) have tested different architectures of BR models on Audio set, we do not use BR++ as a baseline.

[2] For Audio set, the architecture of the encoder is described in Appendix and is not based on RNN. For consistency, we denoted it as Seq2seq.

Models		maF1	miF1	ebF1	ACC	HA	Average
Baselines							
BR		0.349	0.480	0.416	0.086	**0.9957**	0.465
Seq2seq		0.345	0.448	0.421	**0.140**	0.9942	0.470
Seq2seq + SS		0.340	0.448	0.419	0.137	0.9943	0.468
Order-free RNN		0.310	0.438	0.410	0.096	0.9940	0.450
Order-free RNN + SS		0.310	0.437	0.408	0.095	0.9947	0.449
Proposed methods							
OCD		0.353	0.465	0.435	0.117	0.9941	0.473
OCD + MTL	RNN dec.	0.359	0.466	0.438	0.115	0.9940	0.474
	BR dec.	0.353	0.485	0.420	0.075	0.9950	0.466
	Logistic rescore	**0.378**	0.487	**0.456**	0.096	0.9940	0.482
	Logistic joint dec.	0.377	**0.488**	0.454	0.119	0.9945	**0.487**

Table 4: Performance comparisons on Audio set.

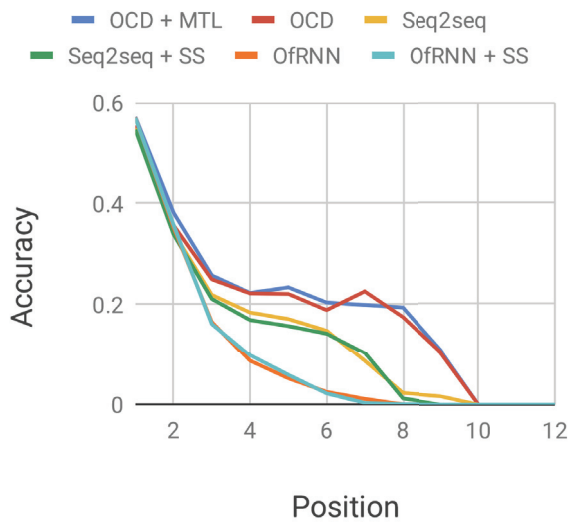

Figure 3: Position-wise accuracy of different models at each time step on Audio set. OCD+MTL was decoded by logistic joint decoding. Note that the length of the longest generated(reference) label sequence is 12.

five metrics as a reference. Note that bold texts in Table 2-4 mean that the highest performance in each measure.

Experiments on AAPD

The experimental results on the AAPD dataset are shown in Table 2. We see that different models are skilled at different metrics. For example, RNN decoder based models, i.e. Seq2seq in row (f) and Order-free RNN in row (h), perform well on ACC, whereas BR and BR++ have better results in terms of HA but show clear weaknesses in predicting rare labels (cf. especially maF1). However, OCD in row (j) performs better than all the baselines (row (d)–(i)) (0.672 on average),[3] especially in miF1 (0.707) and ebF1 (0.737), which verifies the power of the proposed training algorithm.

For MTL, we report the performance for the decoding strategies from section *Decoder Integration*. The first two decoding methods (rows (k),(l)) consider only the predictions of one decoder, while the other two (rows (m),(n)) combine the predictions using different decoding strategies.

[3]Except for ACC: the reason is given in the following discussion.

	Seen test set		Unseen test set	
Models	miF1	ebF1	miF1	ebF1
Seq2seq	0.730	0.749	0.508	0.503
Seq2seq + SS	0.736	0.754	0.517	0.515
Order-free RNN	0.732	0.746	0.496	0.494
Order-free RNN + SS	0.724	0.740	0.520	0.517
OCD (correct prefix)	0.726	0.741	0.513	0.515
OCD	**0.746**	**0.771**	**0.521**	**0.530**

Table 5: Performance comparison on resplited AAPD, whose test set contains 2000 samples whose label sets occur in the training set (Seen test set) and 2000 samples are not (Unseen test set). OCD (correct prefix) means we only sample correct labels in the training phase.

With MTL, we see the performance is improved across all the metrics except for ACC (row (j) v.s. row (k)). In addition, joint decoding methods (row(m),(n)) achieve the best performance, and outperform previous works (row(a),(b),(c)). Interestingly, BR is also improved significantly with MTL (row (d) v.s. row(l)), the encoder of which may implicitly learn the label dependencies through the RNN decoder, which the original BR (row (d)) ignores.

Fig. 2 shows the validation ACC and miF1 curves for OCD, BR, and MTL with three decoding methods. BR performs the worst and converges the slowest. Nonetheless, with the help of MTL, BR converges much faster and better. Also, MTL helps to improve the performance of the OCD model.

Experiments on Reuters-21758

In comparison with AAPD, Reuters-21758 is a relatively small dataset. The average number of labels per sample is only 1.24 and over 80% of the samples have only one label, making for a relative simple dataset.

Table 3 shows the results of the methods. These results demonstrate again the superiority of OCD and the performance gains afforded by the MTL framework. Since there are over 80% of test samples only have one label in this corpus, to truly know the effect of proposed approaches to multi-label classification, we also provide results only on test samples with more than one label in the Section *Analysis of Reuters-21758* in Appendix.

Experiments on Audio set

Table 4 shows the performance of each model. In this experiment, all models have similar performance in HA. Surprisingly, BR is a competitive baseline model and performs especially well in miF1. Seq2seq achieves the best performance in terms of ACC, which is the same as the observation on AAPD. Overall, OCD performs better than all the baseline and MTL indeed improves the performance. OCD outperforms other RNN decoder-based models in maF1, miF1 and ebF1 and performs worse than BR only in terms of miF1.

Models	Case 1	Case 2	Case 3
Ground truth	**cs.it**, **math.it**, cs.ds	cs.lg, stat.ml, math.st, stat.th	**cs.it**, **math.it**, cs.ds, cs.dc
Seq2seq	**cs.it**, **math.it**, cs.ni	**cs.it**, **math.it**, math.st, stat.th	**cs.it**, **math.it**, cs.ni
Order-free RNN	**math.it**, **cs.it**	**math.it**, **cs.it**, stat.th, math.st	**math.it**, **cs.it**, cs.ni
OCD	**math.it**, **cs.it**, cs.ds	stat.ml, stat.th, cs.lg, math.st	**cs.it**, **math.it**
OCD + MTL + joint dec.	**cs.it**, **math.it**	math.st, stat.th, stat.ml, cs.lg, **cs.it**, **math.it**	**math.it**, **cs.it**

Table 6: Examples of generated label sequences on AAPD from different models

	Reference	OCD	Seq2seq	Seq2seq + SS	OfRNN	OfRNN + SS
AAPD						
S_{test}	392	302	214	293	251	259
$S_{test-train}$	43	30	1	3	1	4
Reuters-21758						
S_{test}	210	159	135	140	139	144
$S_{test-train}$	94	37	15	16	23	26
Audio set						
S_{test}	6300	4787	1974	1781	1806	1789
$S_{test-train}$	2445	1943	4	8	263	237

Table 7: Number of different generated (or reference) label combinations (S_{test}), and the number of generated (or reference) label combinations that do not occur in the training label sets ($S_{test-train}$) on three datasets.

Discussion

Propagation of errors Fig. 3 shows position-wise accuracy of different models at each time step of RNN decoder on Audio set. The accuracy is calculated by checking whether or not model's generated labels are in reference label sets and then averaging the errors at each time step. If a generated label sequence is less than the corresponding target label sequence, the unpredicted part of the sequence is considered wrong.

We can see that accuracy of all models decreases dramatically along x-axis. Because the labels are generated sequentially, the models would condition on wrong generated prefix label sequence in the test stage. This problem may be amplified when the generated sequence is longer because of accumulation of errors. Compared with the baseline models, OCD+MTL and OCD perform better after position 2, which demonstrates that they are more robust against error propagation, or exposure bias.

Effectiveness of scheduled sampling and OCD To demonstrate the effectiveness of scheduled sampling and OCD when dealing with exposure bias, we evaluate the performance of models when tackling with unseen label combinations, where models encounter unseen situations and the problem of exposure bias may be more severe.

In this experiment, since there are only 43 samples with unseen label combinations in original test set of AAPD, we resplited the AAPD dataset: 47840 samples in training set, 4000 samples for validation set and test set, respectively. Both validation set and test set contain 2000 samples whose label sets occur in the training set and 2000 samples are not.

Table 5 shows the results on resplited AAPD. OCD (correct prefix) means we only sample correct labels in the training phase, so this model has not encountered wrong prefix during training. Clearly, all models perform worse on unseen test set. We can see that SS improves the performance significantly on the unseen test set for both seq2seq and order free RNN. Additionally, OCD with correct prefix, which suffers from the exposure bias, performs worse in both case than OCD. They all demonstrate that sampling wrong labels from predicted distribution helps models become more robust when encountering rare situation.

SS for MLC has a potential drawback. The input labels of RNN decoders obtained by sampling could be labels which do not conform to the predefined order. This may mislead the model. However, there is no label ordering in OCD, so this problem does not exist.

On both seen and unseen test set, OCD performs the best since OCD not only alleviates exposure bias but also does not need predefined order.

Problem of overfitting Table 7 shows number of different generated label combinations (S_{test}), and the number of generated label combinations that do not occur in the training label sets ($S_{test-train}$) on three datasets. Seq2seq and OfRNN produce fewer kinds of label combinations on AAPD and Reuters-21758. As they tend to "remember" label combinations, the generated label sets are most alike, indicating a poor generalization ability to unseen label combinations. Because seq2seq is conservative and only generates label combinations it has seen in the training set, it achieves the highest ACC in Tables 2 and 4. For models with SS, they produce more kinds of label combinations, except for Audio set. OCD produces the most unseen label combinations on three datasets, since it encounters different label permutations during training.

Case study Table 6 shows examples of generated label sequences using different models on AAPD. Note labels *cs.it* and *math.it* in the three cases: Seq2seq produces label sequences only from frequent to rare, which is the same as the ground truth order, while order-free RNN learns the order implicitly. In contrast, OCD generates label sequences with flexible orders because it encounters different label permutations in the sampling process during training.

Conclusion

In this paper, we propose a new framework for multi-label classification based on optimal completion distillation and multitask learning. Extensive experimental results show that our method outperforms competitive baselines by a large margin. Furthermore, we systematically analyze exposure bias and the effectiveness of scheduled sampling.

References

Bengio, S.; Vinyals, O.; Jaitly, N.; and Shazeer, N. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, 1171–1179.

Chen, S.-F.; Chen, Y.-C.; Yeh, C.-K.; and Wang, Y.-C. F. 2017. Order-free rnn with visual attention for multi-label classification. *arXiv preprint arXiv:1707.05495*.

Cheng, W.; Hüllermeier, E.; and Dembczynski, K. J. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 279–286.

Debole et al., F. 2005. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and technology* 56(6):584–596.

Dembczynski, K.; Waegeman, W.; and Hüllermeier, E. 2012. An analysis of chaining in multi-label classification. In *ECAI*, 294–299.

Gemmeke, J. F.; Ellis, D. P.; Freedman, D.; Jansen, A.; Lawrence, W.; Moore, R. C.; Plakal, M.; and Ritter, M. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 776–780. IEEE.

He, S.; Xu, C.; Guo, T.; Xu, C.; and Tao, D. 2018. Reinforced multi-label image classification by exploring curriculum. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Kumar, A.; Vembu, S.; Menon, A. K.; and Elkan, C. 2013. Beam search algorithms for multilabel learning. *Machine learning* 92(1):65–89.

Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397.

Li, N., and Zhou, Z.-H. 2013. Selective ensemble of classifier chains. In *International Workshop on Multiple Classifier Systems*, 146–156. Springer.

Nam, J.; Mencía, E. L.; Kim, H. J.; and Fürnkranz, J. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in neural information processing systems*, 5413–5423.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2011. Classifier chains for multi-label classification. *Machine learning* 85(3):333.

Sabour, S.; Chan, W.; and Norouzi, M. 2018. Optimal completion distillation for sequence learning. *arXiv preprint arXiv:1810.01398*.

Senge, R.; Del Coz, J. J.; and Hüllermeier, E. 2014. On the problem of error propagation in classifier chains for multi-label classification. In *Data Analysis, Machine Learning and Knowledge Discovery*. Springer. 163–170.

Sucar, L. E.; Bielza, C.; Morales, E. F.; Hernandez-Leal, P.; Zaragoza, J. H.; and Larrañaga, P. 2014. Multi-label classification with bayesian network-based chain classifiers. *Pattern Recognition Letters* 41:14–22.

Tsai, C.-P., and Lee, H.-Y. 2018. Adversarial learning of label dependency: A novel framework for multi-class classification. *arXiv preprint arXiv:1811.04689*.

Tsoumakas, G., and Katakis, I. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)* 3(3):1–13.

Vinyals, O.; Bengio, S.; and Kudlur, M. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Wang, J.; Yang, Y.; Mao, J.; Huang, Z.; Huang, C.; and Xu, W. 2016. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2285–2294.

Yang, P.; Ma, S.; Zhang, Y.; Lin, J.; Su, Q.; and Sun, X. 2018a. A deep reinforced sequence-to-set model for multi-label text classification. *arXiv preprint arXiv:1809.03118*.

Yang, P.; Sun, X.; Li, W.; Ma, S.; Wu, W.; and Wang, H. 2018b. SGM: sequence generation model for multi-label classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3915–3926.

Yu, C.; Barsim, K. S.; Kong, Q.; and Yang, B. 2018. Multi-level attention model for weakly supervised audio classification. *arXiv preprint arXiv:1803.02353*.