Learning Context-Dependent Label Permutations for Multi-Label Classification

Jinseok Nam [1] **Young-Bum Kim** [1] **Eneldo Loza Mencía** [2] **Sunghyun Park** [1] **Ruhi Sarikaya** [1] **Johannes Fürnkranz** [2]

Abstract

A key problem in multi-label classification is to utilize dependencies among the labels. Chaining classifiers are a simple technique for addressing this problem but current algorithms all assume a fixed, static label ordering. In this work, we propose a multi-label classification approach which allows to choose a dynamic, context-dependent label ordering. Our proposed approach consists of two sub-components: a simple EM-like algorithm which bootstraps the learned model, and a more principled approach based on reinforcement learning. Our experiments on three public multi-label classification benchmarks show that our proposed dynamic label ordering approach based on reinforcement learning outperforms recurrent neural networks with fixed label ordering across both bipartition and ranking measures on all the three datasets. As a result, we obtain a powerful sequence prediction-based algorithm for multi-label classification, which is able to efficiently and explicitly exploit label dependencies.

1. Introduction

Multi-label classification (MLC) is the problem of mapping an instance to a subset of possible labels (Zhang & Zhou, 2014). *Classifier chains* (CCs) (Read et al., 2011) and their probabilistic counterpart, *probabilistic classifier chains* (PCCs) (Dembczyński et al., 2010), have proven to be a simple but powerful MLC method for exploiting label dependencies. These methods organize the set of possible labels as a pre-defined sequence and learn to predict each respective next label by including the predictions for previous labels to the input features of this training set. A drawback of CCs is that they require a fixed static ordering of labels, as has already been noted by Read et al. (2011). Many past research tried to address this problem by exploring different label ordering strategies (Li & Zhou, 2013; Sucar et al., 2014; Kumar et al., 2013; Read et al., 2014). However, projecting label dependencies into a sequential ordering is non-trivial (Malerba et al., 1997).

A research direction that have been less explored is dynamically choosing a suitable label ordering for individual instances. Most notably, da Silva et al. (2014) have shown that an ensemble of PCCs with dynamically chosen label orderings perform better than one with vanilla PCCs. Its key idea is to address the error propagation problem of chain-based multi-label prediction methods (Senge et al., 2014). If base classifiers in the beginning of the chain make errors at prediction time, those errors are propagated to predictions of the classifiers at later steps in the chain. Although building an ensemble of PCCs with different label orders depending on instances performs consistently better than PCCs with a single global label ordering, their approach is limited to selecting from a fixed set of label chains used during training time, and also it is a computationally demanding method that requires nearest neighbor search at test time. Kulessa & Loza Mencía (2018), in contrast, overcome these difficulties by employing a special kind of decision trees which allow to define its prediction objective – predicting the next label in the sequence – on the fly during test time. However, the price to pay is its limited predictive performance.

In this work, we propose two techniques for learning context-dependent label sequences, where context can be characterized by input features. Both approaches rely on formulating MLC as a sequence-prediction problem that can be solved via *recurrent neural networks* (RNNs) (Nam et al., 2017), where only relevant labels are predicted in a sequential manner as in (Doppa et al., 2014). In the first approach, we sample a most probable label sequence for a given instance using the intermediate stages of RNN during model training. The sampled target label sequence is complemented by the corresponding true label subset and then used as if the true target sequence to update RNN's parameters. The second technique performs the exploration in a more structured way using reinforcement to learn a sequence-generating policy.

[1]Amazon, Seattle, Washington, USA [2]Knowledge Engineering, TU Darmstadt, Darmstadt, Hessen, Germany. Correspondence to: Jinseok Nam <jinseo@amazon.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

We analyze both techniques empirically on datasets with different characteristics and in comparison to static baseline sequence ordering strategies. Our results demonstrate that in fact the use of context-dependent label sequences can substantially improve the predictive performance against the previous state-of-the-art baseline using RNNs with static label ordering strategies on three public benchmark datasets.

2. Multi-label Classification as Sequence Prediction

Let us consider a pair of an instance $\mathbf{x} \in \mathbb{R}^D$ and its corresponding set of T relevant labels $\boldsymbol{y} = \{y_1, y_2, \cdots, y_T\}$ where y_i are indices from a vocabulary $\mathcal{Y}$ of L labels. As we want to exploit label dependencies, one of the goals in MLC is to maximize the joint probability $P(\boldsymbol{y}|\mathbf{x})$ of labels conditioned on the given instance. However, the complexity of learning the joint probability grows exponentially in L.

To avoid this computational complexity, recent work has shown that MLC can be turned into a sequence prediction problem, which can be solved using recurrent neural networks (Wang et al., 2016; Nam et al., 2017). To that end, we can decompose the joint probability $P(\boldsymbol{y}|\mathbf{x})$ into the product of conditional probabilities

$$P(y_1, y_2, \cdots, y_T|\mathbf{x}) = \prod_{i=1}^{T} P(y_i|\boldsymbol{y}_{<i}, \mathbf{x}) \tag{1}$$

where $\boldsymbol{y}_{<i}$ denotes the set of labels that precede label y_i. Each conditional probability of predicting label k at step i can be estimated as

$$P_\theta(y_i = k|\boldsymbol{y}_{<i}, \mathbf{x}) = \frac{\exp(o_{ik})}{\sum_{j=1}^{L} \exp(o_{jk})} \tag{2}$$

where o_{ik} are the output scores that an RNN yields for node k in the output layer at step i. The outputs $\mathbf{o}_i \in \mathbb{R}^L$ are computed by an affine transformation, denoted by FC, of hidden states $\mathbf{h}_i^u$, which is represented by the concatenation of projected hidden states $\mathbf{h}_i$ and label embeddings $\mathbf{u}_{y_{i-1}} \in \mathbb{R}^d$ that summarizes the previous labels. The hidden states $\mathbf{h}_i$ can in turn be computed using *gated recurrent units* (GRUs) (Cho et al., 2014) given previous hidden states $\mathbf{h}_{i-1}$ and contexts $\tilde{\mathbf{x}}_i$ at step i as follows:

$$\begin{aligned} \mathbf{o}_i = \text{FC}\left(\mathbf{h}_i^u\right), \quad \mathbf{h}_i^u = \sigma\left(\mathbf{W}_u \mathbf{u}_{y_{i-1}} + \mathbf{W}_h \mathbf{h}_i\right), \\ \mathbf{h}_i = \text{GRU}(\mathbf{h}_{i-1}, \tilde{\mathbf{x}}_i). \end{aligned} \tag{3}$$

The contexts $\tilde{\mathbf{x}}_i$ are given as the concatenation of a label embedding $\mathbf{u}$, and a nonlinear transformation of input instances $\mathbf{x}$. Formally, it can be defined as

$$\tilde{\mathbf{x}}_i = \left[\mathbf{u}_{y_{i-1}}; \sigma(\text{FC}(\mathbf{x}))\right] \tag{4}$$

where σ denotes a nonlinear function, e.g., tanh or ReLU. The loss for a given label set prediction is measured by the sum of the cross-entropy for all conditional probabilities with respect to the target sequence. For making a prediction, we generate a sequence of labels $\hat{\boldsymbol{y}}$ using Eq. (2) iteratively until a virtual label that indicates the end of the label sequence is sampled from $P_\theta(y_i|\boldsymbol{y}_{<i}, \mathbf{x})$ (Nam et al., 2017).

Although the RNN architectures for MLC have shown performance improvement over traditional approaches such as PCCs that also minimize subset 0/1 loss, a key problem remains unsolved. Like all other label chain approaches, the use of RNNs relies on the fact that the joint probability of labels can, in principle, be reduced to the product of the conditional probabilities. In theory, this implies that different orders of conditional probability computations should yield the same joint probability of labels (Dembczyński et al., 2010). However, in practice, we learn conditional probabilities from limited data sampled from the true data distribution, which may lead us to different joint probability estimates depending on the order of conditional probability estimation steps. As an example, let us consider two RNN models parameterized by θ_1 and θ_2. Suppose we train θ_1, θ_2 using label orderings y_2, y_1 and y_1, y_2, respectively, then it is very likely that we would obtain

$$P_{\theta_1}(y_1|y_2, \mathbf{x}) P_{\theta_1}(y_2|\mathbf{x}) \neq P_{\theta_2}(y_2|y_1, \mathbf{x}) P_{\theta_2}(y_1|\mathbf{x}) \tag{5}$$

even though both terms are theoretically equivalent and have been learned from the same training data.

This order-dependence of classifier chains has already been observed by Read et al. (2011). The sound probabilistic analysis of the approach goes back to Dembczyński et al. (2010). Like many previous works on conventional classifier chains (Li & Zhou, 2013; Sucar et al., 2014; Kumar et al., 2013; Read et al., 2014), Nam et al. (2017) have observed that, also in the context of RNNs, the choice of label orderings has an impact on prediction results, comparing with a variety of label ordering strategies. However, the label ordering strategies proposed by Nam et al. (2017) are still static and applied globally regardless of context.

3. Generating Label Sequences Without Target Sequences

In the following, we review commonly used static strategies, a random dynamic strategy, and then propose two informed dynamic alternatives in Sections 3.2 and 3.3. We discuss the weakness of the informed dynamic approaches and combine them into a unified learning framework in Section 3.4.

3.1. Static and Random Label Permutations

A straightforward way to generate label sets in a sequential manner is to bring the labels into an arbitrary but fixed order, and to use it to sort relevant labels during training. Traditionally, CC approaches create several label chains randomly

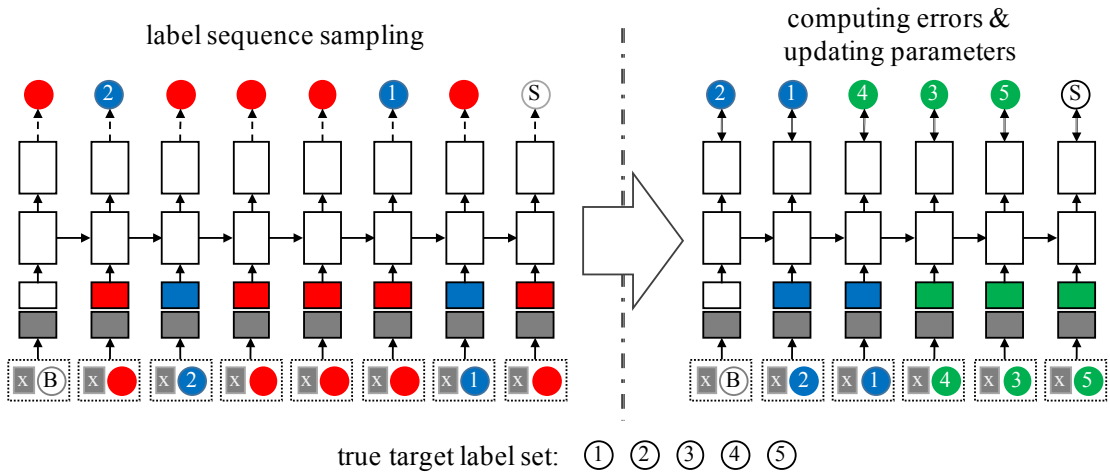

Figure 1. Label permutation learning driven by RNNs. Instead of using a static label ordering strategy, we run RNNs to stochastically generate the most probable label sequence using the current model parameters for a given instance $\mathbf{x}$ (*left*: label sequence sampling). This instance is then used to train the network using a label ordering that consists of the predicted relevant labels in the predicted order, followed by the missing labels in an arbitrary order (*right*: cross-entropy loss minimization).

and build an ensemble of CC models trained on the respective random label chains (Read et al., 2011; Dembczyński et al., 2010). We refer to this strategy as **fixed-rnd**. A more informed way to convert a set of relevant labels into a sequence is to sort the labels according to the label frequency distribution so that frequent labels come first in a label sequence, followed by less frequent ones, which we refer to as **freq2rare**. We can also sort them in reverse such that rare labels are placed in the front, referred to as **rare2freq**. Note that both of the frequency-based label permutation approaches have been compared in (Nam et al., 2017).

A simple way for dynamically creating label sequences is to randomly permute labels in a relevant label set every time a training example is presented to the learner. In other words, each training example will have $|\boldsymbol{y}|!$ target label sequences and only one of them is given as a target sequence when it is fed into the network. We refer to the second random label ordering as **always-rnd**. Note that a dynamic strategy like **always-rnd** can only be used in RNN-like approaches which are able to predict labels in varying orders.

3.2. Model-Based Label Permutations

The key idea for obtaining a dynamic training sequence is to bootstrap the training process using the currently trained model. We thus refer to it as *model-based label permutation* (MbLP). More precisely, for a given training instance $\mathbf{x}$, we first generate a prediction $\hat{\boldsymbol{y}}$, from which we extract the sequence of predicted labels. In the beginning of the training phase, the quality of the sequences generated by the RNN will be poor, but it will improve as we train the RNN on more training data. In any case, we can use the predicted results to order the target label sets $\boldsymbol{y}$ as follows: all predicted labels $y_i \in \boldsymbol{y} \cap \hat{\boldsymbol{y}}$ are sorted in the order in which they are predicted by the RNN, and all labels from $\boldsymbol{y} \setminus \hat{\boldsymbol{y}}$ are appended in a random order. The resulting label sequence $\boldsymbol{y}_{\text{mp}}$ is then used for training the network on $\mathbf{x}$.

Figure 1 illustrates this process with an example in which we have five relevant labels, i.e., the target label set is $\boldsymbol{y} = \{1, 2, 3, 4, 5\}$. First, the RNN generates a prediction by repeatedly sampling a label from the conditional probability $P(y_i | \boldsymbol{y}_{<i}, \mathbf{x})$, using the sampled label as an input label for the next step. This sampling process is repeated until the special virtual label, denoted by *S* in the figure, is sampled, or until the length of the label sequence reaches a predefined maximum number. As can be seen in the left part of Figure 1, it may generate *false positives* (FP) (circles in red) as well as *true positives* (TP) (circles in blue, with the label numbers inscribed). Once a complete label sequence has been generated, we drop all FP from the sequence while keeping the order of the TP. In turn, the *false negatives* (FN) (circles in green) are added to the end of the truncated sequence. Since the FN are not sampled in the first step and the proper order of FN is unknown, the order of FN is determined randomly. Thus, we obtain the label sequence $\boldsymbol{y}_{\text{mp}} = \langle 2, 1, 4, 3, 5 \rangle$, which is used as the target sequence for updating the parameters of the RNN using the cross-entropy loss. Note that the number of parameters in the RNNs for MbLP does not increase compared to the RNNs that use **fixed-rnd** and **always-rnd**, but it requires another forward pass to generate the label sequence.

3.3. Reinforcement Learning of Label Permutations

Even though we cannot trivially generate a good label permutation for an example $\mathbf{x}$, we can nevertheless estimate the

quality of a given permutation using a multi-label evaluation measure, and use this as a feedback signal for learning label orderings. In *reinforcement learning* (RL), a policy $\pi(s)$ defines a probability distribution over actions from which an agent takes actions a given states s of the environment, then receives a reward r from the environment. In our setting, sampling a label from $P_\theta(y_i|\boldsymbol{y}_{<i}, \mathbf{x})$ in Eq. (1) can be seen as taking a probabilistic action following a stochastic policy $P_\theta(a_i|s_i)$ parameterized by θ. The states s_i correspond to the hidden states $\mathbf{h}_i$ of the RNN.

A trajectory is an alternating state-action sequence $\tau = \{s_0, a_0, \cdots, s_{T-1}, a_{T-1}, s_T\}$ following $P_\theta(\tau) = \prod_{i=0}^{T-1} P(s_{i+1}|s_i, a_i)P_\theta(a_i|s_i)$ where $P(s_{i+1}|s_i, a_i)$ is a state transition distribution. The objective of learning $P_\theta(a_i|s_i)$ is to maximize the expected total reward

$$J(\theta) = \mathbb{E}_{P_\theta^\tau}[R(\tau)] = \sum_\tau P_\theta(\tau)R(\tau) \tag{6}$$

where $R(\tau) = \sum_{i=0}^{T-1} \gamma^i r_{i+1}$ is the reward for a trajectory τ. In our case, trajectories correspond to predicted label sequences, and the example-based F_1 measure and subset accuracy can be used as the reward $R(\tau)$. In order to maximize the return, we update $J(\theta)$ in the direction of its gradient $\nabla_\theta J(\theta)$, which is known as episodic REINFORCE (Williams, 1992). As the gradient generally has a high variance due to stochastic processes for the action space exploration, we reduce its variance by explicitly using the immediate rewards for each step based on the observation that future actions do not affect the current reward. In addition, a step-based baseline $b(s_i) \in \mathbb{R}$ reduces the variance even further, resulting in the gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{P_\theta^\tau}\left[\sum_{i=0}^{T-1} \nabla_\theta \log P_\theta(a_i|s_i)\left(R_i - b(s_i)\right)\right] \tag{7}$$

where $R_i = \sum_{l=i}^{T-1} \gamma^{l-i} r_l$ is the total reward from step i with discounting factor γ. In fact, the total reward is an estimate of the *action*-value function $R_i \approx Q(s_i, a_i) = \mathbb{E}_{p_\theta^\pi}\left[\sum_{l=i}^{T} \gamma^{l-i} r_l | s_i, a_i\right]$ and the baseline $b(s_i)$ is an estimate of the *state*-value function $b(s_i) \approx V(s_i) = \mathbb{E}_{P_\theta^\pi}\left[\sum_{l=i}^{T} \gamma^{l-i} r_l | s_i\right]$. Hence, we can define the shifted total reward $R_i - b(s_i)$ as an estimate of the *advantage* function $A(s_i, a_i) = Q(s_i, a_i) - V(s_i)$, which measures how much better the actual total reward $Q(s_i, a_i)$ is than the expected total reward $V(s_i)$. The unbiased estimator of the advantage function would be the *temporal difference* (TD) error $\delta(s_i) = r_{i+1} + \gamma V(s_{i+1}) - V(s_i)$, i.e., $\mathbb{E}_{P_\theta^\pi}[\delta|s, a] = A(s, a)$. Using a parameterized state-value function $V_\theta(s) \approx V(s)$, we minimize the approximated TD error $\delta(s_i) = \frac{1}{2}(r_{i+1} + \gamma V_\theta(s_{i+1}) - V_\theta(s_i))^2$ per step.

Plugging the TD error into Eq. (7), we have

$$\nabla_\theta J(\theta) \approx \frac{1}{K} \sum_{k=1}^{K} \sum_{i=0}^{T-1} \nabla_\theta \log P_\theta(a_i^{(k)}|s_i^{(k)})\delta(s_i^{(k)}) \tag{8}$$

where K denotes the number of samples to approximate the expectation of the gradients. To prevent premature solutions of Eq. (8), we also impose entropy regularization on policy $\beta P_\theta(a_i|s_i)$ that encourages exploration (Williams, 1992) where β controls the regularization strength. This formulation of policy search is known as *actor-critic* (AC) and has been used successfully in several domains. In particular, Bahdanau et al. (2017) have applied it to sequence prediction. Our architecture is rather simple in comparison, but many techniques presented in their work could also be applied in our setting.

As aforementioned, AC receives rewards for each step of the action unlike REINFORCE that only takes a reward at the end of the action sequence generation, so that it helps us avoid the credit assignment problem. In fact, several MLC evaluation measures can be considered as a reward function for AC, but most of them compare the entire set of predicted labels or label ranking with its ground truth label set. Let us denote an evaluation function, e.g., nDCG@5, which we use to calculate the goodness of generated label sequences during training as $\Phi(\hat{\boldsymbol{y}}_i, \boldsymbol{y})$, which takes a (partial) label set $\hat{\boldsymbol{y}}_i$ generated by our model in a sequential manner and a ground truth label set $\boldsymbol{y}$. Then, a reward for taking an action to construct $\hat{\boldsymbol{y}}_{i+1}$ at step $i+1$ can be calculated as follows

$$r_{i+1} = \Phi(\hat{\boldsymbol{y}}_{i+1}, \boldsymbol{y}) - \Phi(\hat{\boldsymbol{y}}_i, \boldsymbol{y}). \tag{9}$$

where $\hat{\boldsymbol{y}}_{i+1} \leftarrow \hat{\boldsymbol{y}}_i \cup \hat{y}_{i+1}$. The above reward function calculates a relative improvement of adding new predicted label $\hat{y}_{i+1}$ to the label sequence generated at the previous step $\hat{\boldsymbol{y}}_i$.

3.4. Learning a Policy with Model-Based Label Permutations

One may find that AC generates a sequence of label stochastically as the MbLP approach in Figure 1. In contrast to MbLPs, we use the target label set to calculate only the goodness of the generated label sequences and AC explores context-dependent label spaces in a way to maximize the goodness. This allows us to maximize directly evaluation measures of interest. However, it could take very long to achieve such a goal because of credit assignment problems of AC in our case. On the other hand, the MbLP approach does not learn to maximize sequence-level quality measure such as example-based F_1 as it tries to minimize the errors with respect to given target sequences at each step independently even though the sequences are generated a model itself stochastically.

Conveniently, both AC and MbLP approaches can share parameters except a set of parameters that are used to represent

the parameterized state-value function $V_\theta(s)$ in AC.[1] We combine both approaches to take advantage of the different strengths as follows

$$\mathcal{L}_{all}(\theta) = \alpha\mathcal{L}_{\text{AC}}(\theta) + (1-\alpha)\mathcal{L}_{\text{mp}}(\theta) \tag{10}$$

where $\mathcal{L}_{\text{AC}}(\theta) = -J(\theta)$ from Eq. (8), and $\mathcal{L}_{\text{mp}}(\theta)$ denotes the loss function of the MbLP approach. We refer to our approach minimizing Equation (10) as *context-dependent label permutation learning for RNN* (CLP-RNN).

4. Experimental Setup

We carried out our experiments on three multi-label datasets from the XML Extreme Classification Repository[2] and their statistics are given in Table 1. For fair comparison, we used the datasets without any further processing except feature scaling that enforces the mean and variance of each feature to be 0 and 1, respectively. We set aside 10% of the training data as the validation sets. We used the same RNN models for all the label ordering strategies including AC and MbLP on each dataset. The dimensionality of label embeddings and hidden activations of our proposed approach on all the datasets were 512 and 2048, respectively. For AC, the number of samples K in Eq. (8) set to 1, discount factor $\gamma \in \{0.1, 0.3, 0.6, 0.9, 0.99\}$ and entropy regularization parameter $\beta \in \{0.01, 0.0001\}$ were chosen based on the performance on the validation set for each dataset. We used either example-based F_1 or nDCG@5 as a reward function. For RNN training, layer normalization (Ba et al., 2016) and variational dropout (Gal & Ghahramani, 2016) were applied. In addition to the use of variational dropout for RNNs, we also applied plain dropout on input features with probability 0.2 or 0.5 when overfitting was observed. As optimization algorithm, we used Adam (Kingma & Ba, 2015) with a learning rate of 0.0001 and minibatches of size 128. In our experiments, no beam search was used when generating label sequences given instances at test time. The reward function r_{t+1} at each step $t+1$ in Eq. (9) was calculated by using either nDCG@5 or example-based F1-measure based on the performance of CLP-RNN on the validation set.

Evaluation Measures The predictive accuracy of MLC algorithms can be evaluated under different aspects. *Bipartition measures* for instance evaluate the ability to predict sets of labels, whereas *ranking measures* assess MLC approaches which produce rankings of labels instead. We extend our proposed methods so that they continue to predict labels even though seeing the sequence stop label in order to allow the comparison to label ranking approaches. *Example-based* bipartition measures compare the target set $\boldsymbol{y}$ to the predicted set $\hat{\boldsymbol{y}}$ for each test example. For instance, subset accuracy requires that the sets are perfectly hit and is hence computed as $\mathbb{I}[\boldsymbol{y} = \hat{\boldsymbol{y}}]$ with $\mathbb{I}$ as indicator function. Example-based F-Measure is less strict since it also considers partial matches and is therefore often used instead: $2|\boldsymbol{y}\cap\hat{\boldsymbol{y}}|/(|\boldsymbol{y}|+|\hat{\boldsymbol{y}}|)$. *Label-based* measures compute TP, FP and FN separately for each label y_j (indicated by index j) and combine these counts by giving each match equal weight as in *micro-averaged* F_1 ($\sum_{j=1}^{L} 2\text{TP}_j / (\sum_{j=1}^{L} 2\text{TP}_j + \text{FP}_j + \text{FN}_j)$), or by equally considering frequent and infrequent labels as in *macro-averaged* F_1 ($\frac{1}{L}\sum_{j=1}^{L} 2\text{TP}_j/(2\text{TP}_j + \text{FP}_j + \text{FN}_j)$). Let us consider $\hat{\boldsymbol{y}}_{\leq k}$ as the set of the first k predicted labels for instance $\boldsymbol{x}_i$ and $\hat{\boldsymbol{y}}_i$ as the i-th label. The ranking measure Prec@$k = |\boldsymbol{y}\cap\hat{\boldsymbol{y}}_{\leq k}|/|\hat{\boldsymbol{y}}_{\leq k}|$ corresponds to the average precision of a classifier always predicting k labels. The normalized discounted cumulative gain (nDCG@k) additionally weights predictions at the top of the ranking higher and is computed as $\sum_{i=1}^{k} \frac{\mathbb{I}[\hat{\boldsymbol{y}}_i \in \boldsymbol{y}]}{\log(i+1)} / (\sum_{i=1}^{\min(k,|\boldsymbol{y}|)} \frac{1}{\log(1+i)})$.

Table 1. Summary of datasets with number of training documents (N_{tr}), test documents (N_{ts}), features (D), labels (L), and avg. label cardinality (C).

DATASET	N_{tr}	N_{ts}	D	L	C
Mediamill	30 993	12 914	120	101	4.38
Delicious	12 920	3185	500	983	19.03
EURLex-4K	15 539	3809	5000	3993	5.31

We report our experimental results in terms of evaluation measures that are not commonly considered in the multi-label classification literature except for the macro F-measure. According to Nam et al. (2017), subset accuracy might be less useful when evaluating datasets with higher label cardinality, and Prabhu & Varma (2014) have also pointed out the importance of predicting a few positive labels correctly when the label space is large. Therefore, our proposed method is evaluated in terms of both bipartition measures, i.e., example- and label-based macro F-measures, and ranking measures, i.e., Prec@k and nDCG@k, which are widely used in large-scale multi-label dataset evaluation.

5. Experiments

A key aspect in our experimental evaluation was to demonstrate that using dynamic, context-dependent label sequences improves over using static orderings w.r.t. predictive performance. Furthermore, we wanted to investigate under which circumstances and to what degree our proposed methods supported the sequence models in finding appropriate sequences. Hence, we will lay a special focus on analyzing the learning progress itself on three datasets with selected characteristics.

[1] The size of the parameters for $V_\theta(s)$ is negligible compared to the entire network parameters in our case.

[2] http://manikvarma.org/downloads/XC/XMLRepository.html accessed 2019-01-12.

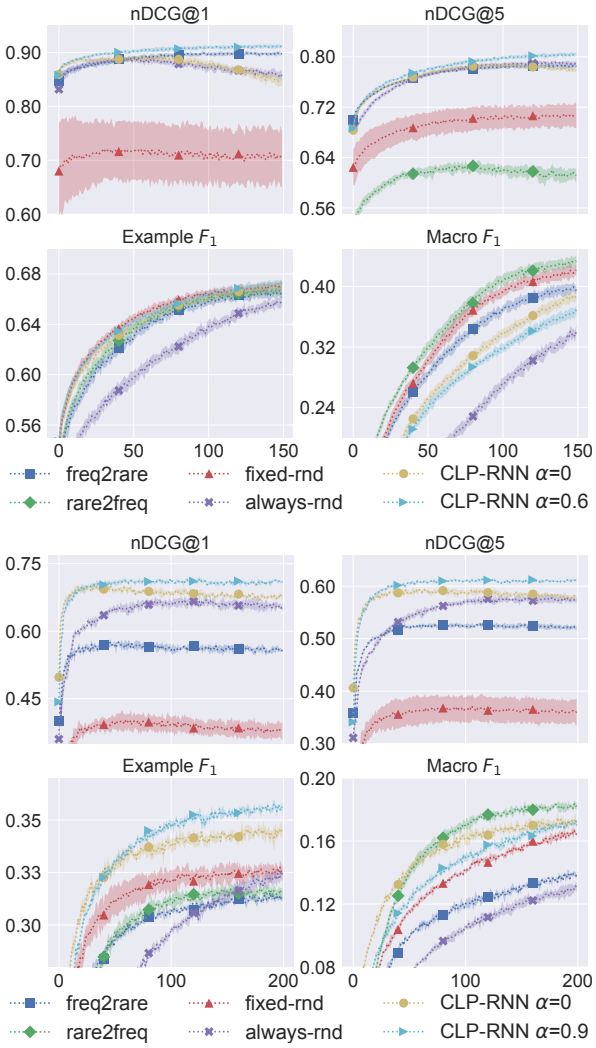

Figure 2. Comparison of label ordering strategies on the test set of Mediamill (top) and Delicious (bottom) in terms of bipartition and ranking measures. The x-axis corresponds to training epochs.

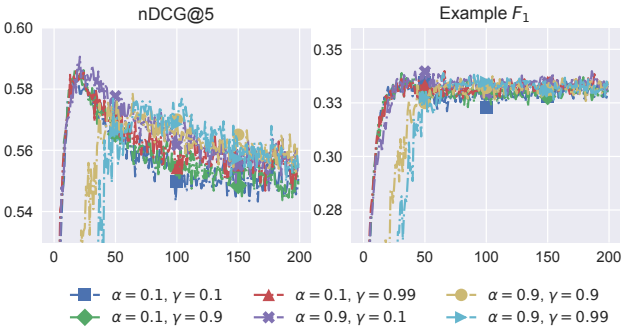

Figure 3. Sensitivity of hyperparameters α and γ for AC on the test set of the Delicious' first split. For this experiment, we did not use dropout to check the effects of α and γ.

5.1. Effect of Label Permutations

Figure 2 shows the performance of RNNs using different label ordering strategies with respect to several evaluation measures on the test set of Mediamill and Delicious after each training epoch. The averages over 10-fold cross validation results are represented as dotted lines whereas lighter areas around the lines denote the standard deviations. As the label cardinality of the Delicious dataset is higher than the other datasets, we can observe clear differences among the label permutations. Note that we used the same RNN architectures and hyperparameter settings on the same data split. The only difference is the way to convert label subsets into target label sequences.

As can be seen from the figure, the performance varies dramatically just by using different label permutations. We observed that the CLP-RNN approach performs consistently better than other label ordering strategies across all evaluation measures except macro-averaged F_1 (also on datasets not shown here). We also found that **always-rnd** works unexpectedly well especially in terms of ranking measures on Mediamill. As RNNs with **always-rnd** are trained on different label sequence patterns as targets for the same training samples every time, training convergence can be very slow compared to other strategies. As expected, their performance increases very slowly in terms of bipartition measures due to the label sequence randomization. However, we observe the opposite in terms of ranking measures. An explanation is that frequent labels appear more often at the beginning of the target label sequences simply due to their higher overall frequency. In contrast to **always-rnd**, **fixed-rnd** performs poorly in terms of generalization performance of rankings and yield higher variance over multiple runs because we create an arbitrary order of labels, which is intact during training, for each run. **freq2rare** shows stable performance across all measures, but **rare2freq** performs very poorly in terms of ranking measures. RNN with the **freq2rare** strategy always receives the most frequent labels at the beginning of the sequences during training, so that RNNs are trained to make lower prediction errors on those labels. Since nDCG@k measures the precision of the top-k labels only, it prefers approaches with a higher performance on frequent labels. The opposite result can be found on the bottom-right sub-figure that compares the performance of different label orderings in terms of label-based macro F_1 measure. In addition, **rare2freq** shows slightly better performance on other bipartition measures.

5.2. Effect of Policy Search in Learning Label Permutations

In the previous subsection, we have compared different label ordering strategies including both static and dynamic ones. In this section, we will present the effect of hyperparameters

Table 2. Comparison showing averages over several runs ± standard deviation and published ranking measure results in the XML repository for SLEEC, FastXML and Parabel (same datasets and splits used).

	Methods	Example F_1	Macro F_1	Prec@1	Prec@3	Prec@5	nDCG@3	nDCG@5
Mediamill	SLEEC	-	-	87.82	73.45	59.17	81.50	79.22
	FastXML	-	-	84.22	67.33	53.04	75.41	72.37
	Parabel	-	-	83.91	67.12	52.99	75.22	72.21
	freq2rare	66.63±0.33	39.68±0.69	90.05±0.31	74.20±0.18	58.39±0.29	82.43±0.25	78.74±0.29
	rare2freq	66.95±0.26	**43.33**±0.62	53.67±1.31	59.57±0.78	52.49±0.37	61.42±0.87	63.46±0.52
	fixed-rnd	67.21±0.25	41.85±0.90	73.95±5.20	65.58±2.31	55.55±0.83	71.14±2.50	71.16±1.66
	always-rnd	66.25±0.25	34.03±0.58	89.08±0.18	73.90±0.24	59.45±0.31	81.77±0.18	79.29±0.21
	CLP-RNN (α=0)	67.22±0.15	38.75±0.88	89.40±0.42	73.84±0.30	59.29±0.17	81.99±0.27	78.92±0.27
	CLP-RNN (α=0.6)	**67.27**±0.30	36.49±0.74	**91.27**±0.28	**75.25**±0.32	**59.75**±0.30	**83.73**±0.32	**80.52**±0.28
Delicious	SLEEC	-	-	67.59	61.38	56.56	62.87	59.28
	FastXML	-	-	69.61	64.12	59.27	65.47	61.90
	Parabel	-	-	67.44	61.83	56.75	63.15	59.41
	freq2rare	31.36±0.17	13.94±0.29	57.21±0.38	54.28±0.31	51.16±0.36	55.04±0.35	52.83±0.45
	rare2freq	31.60±0.15	**18.00**±0.31	17.46±0.38	18.49±0.51	20.31±0.72	18.10±0.52	19.47±0.67
	fixed-rnd	32.74±0.27	16.48±0.31	40.59±1.31	37.21±3.06	35.74±2.60	38.10±2.69	36.96±2.64
	always-rnd	32.45±0.05	13.00±0.25	66.58±0.90	60.46±0.54	54.95±0.55	61.62±0.88	57.63±0.71
	CLP-RNN (α=0)	34.43±0.54	17.33±0.17	69.57±0.43	61.57±0.69	55.73±0.56	63.34±0.77	58.80±0.65
	CLP-RNN (α=0.9)	**35.80**±0.35	**18.00**±0.51	**70.54**±0.77	**63.39**±0.65	**57.72**±0.58	**65.04**±0.60	**60.69**±0.53
EURlex	SLEEC	-	-	79.26	64.30	52.33	68.13	61.60
	FastXML	-	-	71.36	59.90	50.39	62.87	58.06
	Parabel	-	-	81.73	68.78	57.44	72.15	66.40
	freq2rare	49.83±0.05	23.92±0.12	63.79±0.34	58.23±0.11	49.25±0.11	59.84±0.28	55.91±0.08
	rare2freq	50.38±0.08	27.23±0.08	51.17±0.24	51.91±0.01	48.54±0.05	52.03±0.07	52.11±0.20
	fixed-rnd	50.39±0.03	24.95±0.56	58.98±0.66	54.29±0.06	48.78±0.16	55.56±0.09	53.86±0.11
	always-rnd	49.34±0.07	22.17±0.23	72.90±0.64	61.36±0.18	47.62±0.14	64.38±0.01	56.52±0.16
	CLP-RNN (α=0)	51.04±0.23	26.25±0.31	**78.63**±0.41	63.49±0.17	51.87±0.24	67.28±0.18	60.93±0.26
	CLP-RNN (α=0.2)	**53.61**±0.15	**29.96**±0.25	78.25±0.55	**65.32**±0.23	**54.25**±0.13	**68.57**±0.25	**62.85**±0.21

α and γ for CLP-RNN, which is a hybrid model of policy gradient, i.e., AC, and bootstarpping approach to construct a target label ordering based on the current model parameters. Although it is possible to pretrain CLP-RNN by setting α to 0 for a certain number of epochs, we would like to focus the effect of α only in this work. We compared 16 pairs of trade-off parameter $\alpha \in \{0.1, 0.3, 0.6, 0.9\}$ and discount factor $\gamma = \{0.1, 0.3, 0.9, 0.99\}$ on the Delicious dataset with the same setting as explained in Section 4, and results are shown in Fig. 3. For the sake of clarity, we omitted 10 out of 16 configurations of α and γ because the differences between them were negligible. Note that if α is set to 0.0 and 1.0, the methods are identical to MbLP and AC, respectively. As can be seen in the figure, CLP-RNN is robust to the choice of hyperparameters α and γ unless both hyperparameters are close to 1. To be more specific, CLP-RNN converges to relatively bad local minima when we set α to 0.9 and γ is greater than 0.9. The reason of slow convergence to poor parameter spaces of CLP-RNN can be attributed to the stochastic learning process of AC and insufficient training information on rare labels.

5.3. Overall Comparison

We emphasize that our proposed methods extend the state-of-the-art approaches predicting label sequences using RNNs (Nam et al., 2017) whose goal is to generate bi-partition predictions by maximizing subset accuracy. State-of-the-art approaches for extremely large label spaces such as SLEEC (Bhatia et al., 2015), FastXML (Prabhu & Varma, 2014) and Parabel (Prabhu et al., 2018), which focus on ranking measures, can be adapted to bipartition measures by using proper thresholding techniques, however, this is not a trivial task. For example, Nam et al. (2017) reported that label set predictions of FastXML obtained by probabilistic thresholding underperformed the baselines used in our comparison in term of bipartition measures, i.e., RNNs with static label ordering strategies **freq2rare** and **rare2freq**.

Table 2 shows an overall comparison on the test sets. Note that Prec@k and nDCG@k are equivalent for $k = 1$ and also that example-based and micro-averaged F_1 are similar to each other, hence the latter ones, respectively, are skipped. Furthermore, we show the results of the pure MbLP approach ($\alpha = 0$) as well as the performance of the combined version for which we tuned α on the validation set.

We found that CLP-RNN consistently outperforms other label permutation approaches regarding all measures except macro F_1 on Mediamill and Delicious. Especially our version trading off MbLP and AC revealed a clear advantage over all other compared orderings. CLP-RNN performed particularly well on Delicious, where we have a relatively large number of labels L and a high label cardinality C.

Table 3. Wall-clock time in seconds per epoch during training

	Mediamill	Delicious	EURLex
freq2rare	10.42	8.56	7.56
MbLP	14.74	13.93	8.04
CLP-RNN	23.78	22.69	12.57

The observation, that **rare2freq** performs extraordinary well for rare labels, is repeated in this comparison. However, when tuning α, CLP-RNN is able to catch up on Delicious and even overtake on EURLex. On the Mediamill dataset with considerably less labels using AC seems to even harm the macro F_1 performance. Fortunately, the ranking measures substantially improve in exchange. RNNs with **rare2freq** are optimized to make good predictions at rare labels, which leads us to obtain higher macro F_1 scores since the performance for each label contributes equally to the macro F_1 measure. Please note that in many cases the number of rare labels is greater than the number of frequent labels in MLC. In contrast, Prec@1 focuses on the performance of the top ranked labels for which it is crucial to place frequent labels first.

Finally, when comparing to the ranking baselines, SLEEC, FastXML and Parabel, we observe mixed results since the performance of the individual algorithms varies according to the dataset: CLP-RNN outperforms the other methods on Mediamill, FastXML on Delicious, and Parabel on EURLex. Nonetheless, we can also observe that CLP-RNN performs best or second except for a single case (Prec@1 on EURLex). The performance of CLP-RNN is remarkable since it does perform competitively to the state-of-the-art in terms of ranking measures as well as generating bipartition predictions without manual tuning of threshold that is used to convert label ranking to biparition predictions.

5.4. Computational Complexity

We have demonstrated that CLP-RNN outperforms other label ordering strategies when training RNNs. As our proposed method needs a label sequence sampling stage to learn context-dependent label orderings, a single gradient update step of CLP-RNN takes more time compared to RNNs with static label orderings such as freq2rare and rare2freq. All RNN-based MLC models were trained on NVIDIA Tesla V100 and we measured wall-clock time to check the computational cost of our proposed method. As can be seen in Table 3, the gap of training time between the static label ordering approaches such as **freq2rare** and the dynamic ones grows in terms of the average number of labels, i.e., label cardinality C in Table 1. Note that MbLP is a special case of CLP-RNN where α is set to 0, so that it does not require reward computation and the backpropagation step as AC in Equation (10). Based on our timing results, CLP-RNN is indeed $2 \sim 3$ times more expensive than **freq2rare** for iterating each epoch. However, note that training RNNs with static label ordering strategies $2 \sim 3$ times longer does not improve their performance over CLP-RNN.

6. Conclusion

Our work has advanced the state-of-the-art in multi-label classification in several ways: Most importantly, we have shown that a dynamic, context-sensitive selection of label orderings consistently outperforms commonly used static techniques. Among two competing algorithms for dynamic label ordering, we found that a simple approach based on an EM-like bootstrapping of the label ordering via the learned model performs surprisingly well. Also, our empirical analyses showed that the hybrid approach based on policy gradient reinforcement learning improves even further the effectiveness of RNN based MLC approaches. As a result, we have obtained a state-of-the-art multi-label classification system with the following two advantages over previous chain-based approaches: Firstly, the length of the predicted sequence does not depend on the number of possible labels, but only on the implicitly predicted number of relevant labels, which is particularly relevant when addressing problems with large label sets. Secondly, the complex and expensive task of selecting the label permutation can be saved since it is naturally tackled directly in the training process by our proposed system. Further improvements to the training process, e.g., by conveniently combining some of the proposed sampling strategies, could also be beneficial when other objectives are desired, as our experimental results on label ranking suggest. Finally, our empirical results showed that the context-dependent label permutation learning approach is an effective solution when we are interested in particularly ranking measures which are commonly used to evaluate many real-world applications such as intelligent personal digital assistants (Kim et al., 2018; Sarikaya, 2017).

Acknowledgements

The authors would like to thank anonymous reviewers for the constructive feedback.

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. An actor-critic algorithm for sequence prediction. In *Proceedings of the International Conference on Learning Representations*,

2017.

Bhatia, K., Jain, H., Kar, P., Varma, M., and Jain, P. Sparse local embeddings for extreme multi-label classification. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 730–738. 2015.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN Encoder–Decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014.

da Silva, P. N., Gonçalves, E. C., Plastino, A., and Freitas, A. A. Distinct chains for different instances: An effective strategy for multi-label classifier chains. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pp. 453–468, 2014.

Dembczyński, K., Cheng, W., and Hüllermeier, E. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 279–286, 2010.

Doppa, J. R., Fern, A., and Tadepalli, P. HC-Search: A learning framework for search-based structured prediction. *Journal of Artificial Intelligence Research*, 50:369–407, 2014.

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1019–1027, 2016.

Kim, Y.-B., Kim, D., Kim, J.-K., and Sarikaya, R. A scalable neural shortlisting-reranking approach for large-scale domain classification in natural language understanding. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pp. 16–24, 2018.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

Kulessa, M. and Loza Mencía, E. Dynamic classifier chain with random decision trees. In *Proceedings of the 21st International Conference of Discovery Science*, pp. 33–50, 2018.

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

Li, N. and Zhou, Z.-H. Selective ensemble of classifier chains. In Zhou, Z.-H., Roli, F., and Kittler, J. (eds.), *Multiple Classifier Systems*, pp. 146–156. Springer Berlin Heidelberg, 2013.

Malerba, D., Semeraro, G., and Esposito, F. A multistrategy approach to learning multiple dependent concepts. In Nakhaeizadeh, G. and Taylor, C. C. (eds.), *Machine Learning and Statistics: The Interface*, chapter 4, pp. 87–106. Wiley, London, England, 1997.

Nam, J., Loza Mencía, E., Kim, H. J., and Fürnkranz, J. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in Neural Information Processing Systems*, pp. 5419–5429, 2017.

Prabhu, Y. and Varma, M. FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 263–272, 2014.

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., and Varma, M. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 993–1002. International World Wide Web Conferences Steering Committee, 2018.

Read, J., Pfahringer, B., Holmes, G., and Frank, E. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, 2011.

Read, J., Martino, L., and Luengo, D. Efficient Monte Carlo methods for multi-dimensional learning with classifier chains. *Pattern Recognition*, 47(3):1535 – 1546, 2014.

Sarikaya, R. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Processing Magazine*, 34(1): 67–81, 2017.

Senge, R., Del Coz, J. J., and Hüllermeier, E. On the problem of error propagation in classifier chains for multi-label classification. In Spiliopoulou, M., Schmidt-Thieme, L., and Janning, R. (eds.), *Data Analysis, Machine Learning and Knowledge Discovery*, pp. 163–170. Springer International Publishing, 2014.

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

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., and Xu, W. CNN-RNN: A unified framework for multi-label image classification. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2285–2294, 2016.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.

Zhang, M. and Zhou, Z. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.