Adapting RNN Sequence Prediction Model to Multi-label Set Prediction

Kechen Qin   **Cheng Li**   **Virgil Pavlu**   **Javed A. Aslam**

Khoury College of Computer Sciences

Northeastern University

`qin.ke@husky.neu.edu`   `{chengli,vip,jaa}@ccs.neu.edu`

Abstract

We present an adaptation of RNN sequence models to the problem of multi-label classification for text, where the target is a *set of labels*, not a sequence. Previous such RNN models define probabilities for sequences but not for sets; attempts to obtain a set probability are after-thoughts of the network design, including pre-specifying the label order, or relating the sequence probability to the set probability in *ad hoc* ways.

Our formulation is derived from a principled notion of set probability, as the sum of probabilities of corresponding permutation sequences for the set. We provide a new training objective that maximizes this set probability, and a new prediction objective that finds the most probable set on a test document. These new objectives are theoretically appealing because they give the RNN model freedom to discover the best label order, which often is the natural one (but different among documents).

We develop efficient procedures to tackle the computation difficulties involved in training and prediction. Experiments on benchmark datasets demonstrate that we outperform state-of-the-art methods for this task.

1 Introduction

Multi-label text classification is an important machine learning task wherein one must predict a set of labels to associate with a given document; for example, a news article might be tagged with labels `sport`, `football`, `2018 world cup`, and `Russia`. Formally, we are given a set of label candidates $\mathcal{L} = \{1, 2, ..., L\}$, and we aim to build a classifier which maps a document x to a set of labels $\mathbf{y} \subset \mathcal{L}$. The label set $\mathbf{y}$ is typically written as a binary vector $\mathbf{y} \in \{0, 1\}^L$, with each bit y_ℓ indicating the presence or absence of a label.

Naively, one could predict each label independently without considering label dependencies.

This approach is called Binary Relevance (Boutell et al., 2004; Tsoumakas and Katakis, 2007), and is widely used due to its simplicity, but it often does not deliver good performance. Intuitively, knowing some labels—such as `sport` and `football`—should make it easier to predict `2018 world cup` and then `Russia`. There are several methods that try to capture label dependencies by building a joint probability estimation over all labels $p(\mathbf{y} = (y_1, y_2, ..., y_L)|x)$ (Ghamrawi and McCallum, 2005; Read et al., 2009; Dembczynski et al., 2010; Li et al., 2016). The most popular approach, Probabilistic Classifier Chain (PCC) (Dembczynski et al., 2010) learns labels one-by-one in a predefined fixed order: for each label, it uses one classifier to estimate the probability of that label given all previous labels predictions, $p(y_l|y_1, ..., y_{l-1}, x)$. PCC's well known drawback is that errors in early probability estimations tend to affect subsequent predictions, and can become massive when the total number of label candidates L is large.

Recurrent neural network (RNN) is originally designed to output a sequential structure, such as a sentence (Cho et al., 2014). Recently, RNNs have also been applied to multi-label classification by mapping the label set to a sequence (Wang et al., 2016; Zhang et al., 2016; Jin and Nakayama, 2016; Wang et al., 2017b,a; Chen et al., 2018; Yang et al., 2018). In contrast to PCC where a binary decision is made for each label sequentially, RNN only predicts the positive labels explicitly and therefore its decision chain length is equal to the number of positive labels, not the number of all labels. This makes RNN suffer less from early estimation errors than PCC.

Both PCC and RNN rely heavily on label orders in training and prediction. In multi-label data, the labels are given as sets, not necessarily with natural orders. RNN defines a sequence probability,

while PCC defines set probability. Various ways of arranging sets as sequences have been explored: ordering alphabetically, by frequency, based on a label hierarchy, or according to some label ranking algorithm (Liu and Tsang, 2015). Previous experimental results show that which order to choose can have a significant impact on learning and prediction (Vinyals et al., 2016; Nam et al., 2017; Chen et al., 2018). In the above example, starting label predictions sequence with Russia, while correct, would make the other predictions very difficult.

Previous work has shown that it is possible to train an RNN on multi-label data without specifying the label order in advance. With special training objectives, RNN can explore different label orders and converge to some order automatically (Vinyals et al., 2016). In this paper we follow the same line of study: We consider how to adapt RNN sequence model to multi-label set prediction without specifying the label order. Specifically, we make the following contributions:

1. We analyze existing RNN models proposed for multi-label prediction, and show that existing training and prediction objectives are not well justified mathematically and have undesired consequences in practice.

2. We develop efficient approximate training and prediction methods. We propose new training and prediction objectives based on a principled notion of set probability. Our new formulation avoids the drawbacks of existing ones and gives the RNN model freedom to discover the best label order.

3. We crawl two new datasets for multi-label prediction task, and apply our method to them. We also test our method on two existing multi-label datasets. The experimental results show that our method outperforms state-of-the-art methods on all datasets. We release the datasets at http://www.ccis.neu.edu/home/kechenqin.

2 Mapping Sequences to Sets

In this section, we describe how existing approaches map sequences to sets, by writing down their objective functions using consistent notations. To review RNN designed for sequences, let $\mathbf{s} = (s_1, s_2, ..., s_T)$ be an input sequence of outcomes, in a particular order, where $s_t \in$ $\{1, 2, ..., L\}$; the order is often critical to the datapoint. An RNN model defines a probability distribution over all possible output sequences given the input in the form $p(\mathbf{s} = (s_1, s_2, ..., s_T)|x) = \prod_{t=1}^{T} p(s_t|x, s_1, s_2, ..., s_{t-1})$. To train the RNN model, one maximizes the likelihood of the ground truth sequence.

At prediction time, one seeks to find the sequence with the highest probability $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$, and this is usually implemented approximately with a beam search procedure (Lowerre, 1976) (we modified into Algorithm 1). The sequence history is encoded with an internal memory vector h_t which is updated over time. RNN is also often equipped with the attention mechanism (Bahdanau et al., 2014), which in each timestep t puts different weights on different words (features) and thus effectively attends on a list of important words. The context vector c_t is computed as the weighted average over the dense representation of important words to capture information from the document. The context c_t, the RNN memory h_t at timestep t, and the encoding of previous label s_{t-1} are all concatenated and used to model the label probability distribution at time t as $p(s_t|x, s_1, s_2, ..., s_{t-1}) \sim softmax(\phi(c_t, h_t, s_{t-1}))$, where ϕ is a non-linear function, and *softmax* is the normalized exponential function.

To apply RNN to multi-label problems, one approach is to map the given set of labels $\mathbf{y}$ to a sequence $\mathbf{s} = (s_1, s_2, ..., s_T)$, on training documents. This is usually obtained by writing the label set in a globally fixed order (e.g. by label frequency), as in PCC. Once the mapping is done, RNN is trained with the standard maximum likelihood objective (Nam et al., 2017):

$$maximize \sum_{n=1}^{N} \log p(\mathbf{s}^{(n)}|x^{(n)}) \qquad (1)$$

where $x^{(n)}$ is the n-th document and N is the total number of documents in the corpus.

Vinyals et al. (2016) proposes to dynamically choose during training the sequence order deemed as most probable by the current RNN model:

$$maximize \sum_{n=1}^{N} \max_{\mathbf{s} \in \pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)}) \qquad (2)$$

where the $\pi(\mathbf{y}^{(n)})$ stands for all permutations of the label set $\mathbf{y}^{(n)}$. This eliminates the need to manually specify the label order. However, as noticed

Methods	Training objectives	Prediction objectives
seq2seq-RNN	$maximize \sum_{n=1}^{N} \log p(\mathbf{s}^{(n)}|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*), \mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$
Vinyals-RNN-max	$maximize \sum_{n=1}^{N} \max_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*), \mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$
Vinyals-RNN-uniform	$maximize \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*), \mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$
Vinyals-RNN-sample	$maximize \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)}) \log p(\mathbf{s}|x^{(n)})$	$\hat{\mathbf{y}} = set(\mathbf{s}^*), \mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$
set-RNN (ours)	$maximize \sum_{n=1}^{N} \log \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)})$	$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|x)$

Table 1: Comparison between previous and our *set-RNN* training and prediction objectives.

by the authors, this objective cannot be used in the early training stages: the early order choice (often random) is reinforced by this objective and can be stuck upon permanently. To address this issue, Vinyals et al. (2016) also proposes two smoother alternative objectives to initialize the model training:

The authors suggest that one first consider many random orders for each label set in order to explore the space:

$$maximize \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} \log p(\mathbf{s}|x^{(n)}) \quad (3)$$

After that, one can sample sequences following the model predictive distribution instead of uniform distribution:

$$maximize \sum_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)}) \log p(\mathbf{s}|x^{(n)}) \quad (4)$$

In training, one needs to schedule the transition among these objectives, a rather tricky endeavor. At prediction time, one needs to find the most probable set. This is done by (approximately) finding the most probable sequence $\mathbf{s}^* = \arg\max_{\mathbf{s}} p(\mathbf{s}|x)$ and treating it as a set $\hat{\mathbf{y}} = set(\mathbf{s}^*)$. With a large number of sequences, it is quite possible that the argmax has actually a low probability, which can lead to neglecting important information when we ignore sequences other than the top one.

3 Adapting RNN Sequence Prediction Model to Multi-label Set Prediction

We propose a new way of adapting RNN to multi-label set prediction, which we call *set-RNN*. We appreciate the RNN model structure (Rumelhart et al., 1988) (defines a probability distribution over all possible sequences directly) and introduce training and prediction objectives tailored

for sets that take advantage of it, while making a clear distinction between the sequence probability $p(\mathbf{s}|x)$ and the set probability $p(\mathbf{y}|x)$. We define the set probability as the sum of sequences probabilities for all sequence permutations of the set, namely $p(\mathbf{y}|x) = \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$. Based on this formulation, an RNN also defines a probability distribution over all possible sets indirectly since $\sum_{\mathbf{y}} p(\mathbf{y}|x) = \sum_{\mathbf{y}} \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x) = \sum_{\mathbf{s}} p(\mathbf{s}|x) = 1$. (For this equation to hold, in theory, we should also consider permutations $\mathbf{s}$ with repeated labels, such as $(1, 2, 3, 1)$. But in practice, we find it very rare for RNN to actually generate sequences with repeated labels in our setup, and whether allowing repetition or not does not make much difference.)

In standard maximum likelihood training, one wishes to maximize the likelihood of given label sets, namely, $\prod_{n=1}^{N} p(\mathbf{y}^{(n)}|x^{(n)}) = \prod_{n=1}^{N} \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)})$, or equivalently,

$$maximize \sum_{n=1}^{N} \log \sum_{\mathbf{s}\in\pi(\mathbf{y}^{(n)})} p(\mathbf{s}|x^{(n)}) \quad (5)$$

3.1 How is our new formulation different?

This training objective (5) looks similar to the objective (3) considered in previous work (Vinyals et al., 2016), but in fact they correspond to very different transformations. Under the maximum likelihood framework, our objective (5) corresponds to the transformation $p(\mathbf{y}|x) = \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$, while objective (3) corresponds to the transformation $p(\mathbf{y}|x) = \prod_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$. The latter transformation does not define a valid probability distribution over $\mathbf{y}$ (i.e., $\sum_{\mathbf{y}} p(\mathbf{y}|x) \neq 1$), and it has an undesired consequence in practical model training: because of the multiplication operation, the RNN model has to assign equally high probabilities to all sequence permutations of the given label set in order to maximize the set probability. If only some sequence permutations

receive high probabilities while others receive low probabilities, the set probability computed as the product of sequence probabilities will still be low. In other words, if for each document, RNN finds one good way of ordering relevant labels (such as hierarchically) and allocates most of the probability mass to the sequence in that order, the model still assigns low probabilities to the ground truth label sets and will be penalized heavily. As a consequence the model has little freedom in discovering and concentrating on some natural label order. In contrast, with our proposed training objective, in which the multiplication operation is replaced by the summation operation, it suffices to find only one reasonable permutation of the labels for each document. It is worth noting that different documents can have different label orders; thus our proposed training objective gives the RNN model far more freedom on label order. The other two objectives (2) and (4) proposed in (Vinyals et al., 2016) are less restrictive than (3), but they have to work in conjunction with (3) because of the self reinforcement issue. Our proposed training objective has a natural probabilistic interpretation, and does not suffer from self reinforcement issue. Thus it can serve as a stand alone training objective. Also, using Jensen's inequality, one can show that objective (3) is maximizing a lower bound on the log-likelihood, while objective (5) is maximizing it directly.

3.2 Training by Maximizing Set Probability

Training an RNN model with the proposed objective (5) requires summing up sequence (permutation) probabilities for a set $\mathbf{y}$, where $|\mathbf{y}|$ is the cardinality of the set. Thus evaluating this objective exactly can be intractable. We can approximate this sum by only considering the top K highest probability sequences produced by the RNN model. We introduce a variant of beam search for sets with width K and with the search candidates in each step restricted to only labels in the set (see Algorithm 1 with $ALL = 1$). This approximate inference procedure is carried out repeatedly before each batch training step, in order to find highest probability sequences for all training instances occurring in that batch. The overall training procedure is summarized in Algorithm 2.

3.3 Predicting the Most Probable Set

The transformation $p(\mathbf{y}|x) = \sum_{\mathbf{s}\in\pi(\mathbf{y})} p(\mathbf{s}|x)$ also naturally leads to a prediction procedure,

---

Algorithm 1: Beam_Search

Input : Instance x
  Subset of labels considered $G \subset \mathcal{L}$
  Boolean flag ALL: 1 if sequences must contain all G labels; 0 if partial sequences are allowed
Output: A list of top sequences and the associated probabilities

1 Let $\mathbf{s}_1,\mathbf{s}_2,...,\mathbf{s}_K$ be the top K sequences found so far. Initially, all K sequences are empty. $\oplus$ means concatenation.
2 **while** *true* **do**
3   // Step 1: Generate Candidate Sequences from each existing sequence $s_k \in K$ and all possible new labels $l \in G$:
4   Expand all non-stopped sequences:
5   $C = \{\mathbf{s}_k \oplus l | l \in G, STOP \notin s_k\}$
6   Include stopped sequences:
7   $C = C \cup \{\mathbf{s}_k | STOP \in s_k\}$
8   Terminate non-stopped sequences:
9   **if** $ALL == 0$ **then**
10     $C = C \cup \{\mathbf{s}_k \oplus STOP | STOP \notin s_k\}$
11   **end**
12   // Step 2: Select highest probabilities sequences from candidate set C
13   $K = $ topK-argmax$_k\{\text{prob}[s_k] | s_k \in C\}$
14   **if** *all top K sequences end with $STOP$ or contain all labels in G* **then**
15   Terminate the algorithm
16   **end**
17 **end**
18 **return** *sequence list* $\mathbf{s}_1,\mathbf{s}_2,...,\mathbf{s}_K$ *and the associated probabilities*

---

which is different from the previous standard of directly using most probable sequence as a set. We instead aim to find the most likely set $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} p(\mathbf{y}|x)$, which involves summing up probabilities for all of its permutations. To make it tractable, we propose a two-level beam search procedure. First we run standard RNN beam search (Algorithm 1 with $ALL = 0$) to generate a list of highest probability sequences. We then consider the label set associated with each label sequence. For each set, we evaluate its probability using the same approximate summation procedure as the one used during model training (Algorithm 1 with $ALL = 1$): we run our modified beam search to find the top few highest probability sequences associated with the set and sum up their

Algorithm 2: Training method for set-RNN

Input : Multi-label dataset
$(x^{(n)}, \mathbf{y}^{(n)}), n = 1, 2, ..., N$

Output: Trained RNN model parameters

1 **foreach** *batch* **do**
2   **foreach** $(x^n, \mathbf{y}^n)$ *in the batch* **do**
3   Get top K sequences :
4        $\{\mathbf{s}_1^n, ..., \mathbf{s}_K^n, p(\mathbf{s}_1^n|x^n), ..., p(\mathbf{s}_K^n|x^n)\}=$
         $= \text{Beam\_Search}(x^n, \mathbf{y}^n, ALL = 1)$
5   **end**
6   Update model parameters by maximizing
$$\sum_{(x^n, \mathbf{y}^n) \in \text{batch}} \log \sum_{\mathbf{s} \in \{\mathbf{s}_1^n, ..., \mathbf{s}_K^n\}} p(\mathbf{s}|x^n)$$
7 **end**

Algorithm 3: Prediction Method for set-RNN

Input : Instance x

Output: Predicted label set $\hat{\mathbf{y}}$

1 Obtain K highest probability sequences :
2 $\{\mathbf{s}_1, ..., \mathbf{s}_K\} = \text{Beam\_Search}(x, \mathcal{L}, ALL = 0)$
3 Map each sequence $\mathbf{s}_k$ to the corresponding set $\mathbf{y}_k$ and remove duplicate sets (if any)
4 **foreach** $\mathbf{y}_k$ **do**
5   Get K most probable sequences associated with $\mathbf{y}_k$ and their probabilities :
6     $\{\mathbf{s}'_1, ..., \mathbf{s}'_K, p(\mathbf{s}'_1|x), ..., p(\mathbf{s}'_K|x)\}=$
7        $= \text{Beam\_Search}(x, \mathbf{y}_k, ALL = 1)$
8   Set probability is approx by summing up :
$$p(\mathbf{y}_k|x) \approx \sum_{\mathbf{s} \in \{\mathbf{s}'_1, ..., \mathbf{s}'_K\}} p(\mathbf{s}|x)$$
9 **end**
10 $\hat{\mathbf{y}} = argmax_{\mathbf{y}_k}(p(\mathbf{y}_k|x))$

---

probabilities. Among these sets that we have evaluated, we choose the one with the highest probability as the prediction. The overall prediction procedure is summarized in Algorithm 3. As we shall show in case study, the most probable set may not correspond to the most probable sequence; these are certainly cases where our method has an advantage.

Both our method and the competitor state-of-the-art (Vinyals-RNNs) are at most K times slower than a vanilla-RNN, due to the time spent on dealing with K permutations per datapoint. Our proposed method is about as fast as the Vinyals-RNN methods, except for the Vinyals-RNN-uniform which is a bit faster (by a factor of 1.5) because its epochs do not run the additional forward pass.

4 Results and Analysis

4.1 Experimental Setup

We test our proposed set-RNN method on 4 real-world datasets, RCV1-v2, Slashdot, TheGuardian, and Arxiv Academic Paper Dataset (AAPD) (Yang et al., 2018). We take the public RCV1-v2 release[1] and randomly sample 50,000 documents. We crawl Slashdot and TheGuardian documents from their websites[2] and treat the official editor tags as ground truth. We also gather a list of user tags[3] for each document and treat them as additional features. For AAPD dataset, we follow

---

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

[2] Slashdot: https://slashdot.org/ Note that there is another public Slashdot multi-label dataset (Read et al., 2009) but we do not use that one because it is quite small. TheGuardian: https://www.theguardian.com

[3] www.zubiaga.org/datasets/socialbm0311/

---

the same train/test split as in (Yang et al., 2018). Table 2 contains statistics of these four datasets. Links to document, official editor tags, and user tags are avaliable at http://www.ccis.neu.edu/home/kechenqin.

Data	#Train	#Test	Cardinality	#Labels	Doc length
Slashdot	19,258	4,814	4.15	291	64
RCV1-v2	40,000	10,000	3.17	101	121
TheGuardian	37,638	9,409	7.41	1,527	505
AAPD	53,840	1,000	2.41	54	163

Table 2: Statistics of the datasets.

To process documents, we filter out stopwords and punctuations. Each document is truncated to have maximum 500 words for TheGuardian and AAPD, and 120 for Slashdot and RCV1-v2. Zero padding is used if the document contains less words than the maximum number. Numbers and out-of-vocabulary words are replaced with special tokens. Words, user tags and labels are all encoded as 300-dimensional vectors using WORD2VEC (Mikolov et al., 2013).

We implement RNNs with attention using TENSORFLOW-1.4.0 (Abadi et al., 2016). The dynamic function for RNNs is chosen to be Gated recurrent units (GRU) with 2 layers and at most 50 units in decoder. The size of the GRU unit is 300. We set dropout rate to 0.3, and train the model with Adam optimizer (Kingma and Ba, 2014) with learning rate 0.0005. Beam size is set to be 12 at both training and inference stages. We adopt *label-F1* (average F1 over labels) and *instance-F1*

Methods	Slashdot		RCV1-v2		TheGuardian		AAPD			
	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	hamming-loss	micro-F1
BR	.271	.484	.486	.802	.292	.572	.529	.654	.0230	.685
BR-support	.247	.516	.486	.805	.296	.594	.545	.689	**.0228**	.696
PCC	.279	.480	.595	.818	-	-	.541	.688	.0255	.682
seq2seq-RNN	.270	.528	.561	.824	.331	.603	.510	.708	.0254	.701
Vinyals-RNN-uniform	.279	.527	.578	.826	.313	.567	.532	.721	.0241	.711
Vinyals-RNN-sample	.300	.531	.590	.828	.339	.597	.527	.706	.0259	.697
Vinyals-RNN-max	.293	.530	.588	.829	.343	.599	.535	.709	.0256	.700
Vinyals-RNN-max-direct	.226	.518	.539	.808	.313	.583	.490	.702	.0257	.694
SGM	-	-	-	-	-	-	-	-	.0245	.710
set-RNN	**.310**	**.538**	**.607**	**.838**	**.361**	**.607**	**.548**	**.731**	.0241	**.720**

Table 3: Comparison of different approaches. "-" means result not available. For *hamming loss*, the lower the value is, the better the model performs. For all other measures, the higher the better.

Methods	Slashdot		RCV1-v2		TheGuardian		AAPD	
	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1	label-F1	instance-F1
seq2seq-RNN	.270→.269	.528→.528	.561→.561	.824→.824	.331→.336	.603→.603	.510→.511	.708→.709
Vinyals-RNN-uniform	.279→.288	**.527→.537**	.578→.587	.826→.833	**.313→.336**	**.567→.585**	**.532→.542**	.721→.724
Vinyals-RNN-sample	.300→.303	.531→.537	.590→.597	.828→.833	**.339→.351**	.597→.602	.527→.530	.706→.708
Vinyals-RNN-max	.293→.301	.530→.535	.588→.585	.829→.830	.343→.352	.599→.604	.535→.537	.709→.712
Vinyals-RNN-max-direct	.226→.228	.518→.519	.539→.538	.808→.808	.313→.316	.583→.584	.490→.490	.702→.701
set-RNN	**.297→.310**	**.528→.538**	**.593→.607**	.831→.838	**.349→.361**	**.595→.607**	.548→.548	.728→.731

Table 4: Predicting the most probable sequence vs. predicting the most probable set. Numbers before the arrow: predicting the most probable sequence. Numbers after the arrow: predicting the most probable set. We highlight scores which get significantly improved in **bold** (improvement is larger than 0.01).

(average F1 over instances) as our main evaluation metrics, as defined below:

$$\text{label-F1} = \frac{1}{L} \sum_{\ell=1}^{L} \frac{2 \sum_{n=1}^{N} y_\ell^{(n)} \hat{y}_\ell^{(n)}}{\sum_{n=1}^{N} y_\ell^{(n)} + \sum_{n=1}^{N} \hat{y}_\ell^{(n)}}$$

$$\text{instance-F1} = \frac{1}{N} \sum_{n=1}^{N} \frac{2 \sum_{\ell=1}^{L} y_\ell^{(n)} \hat{y}_\ell^{(n)}}{\sum_{\ell=1}^{L} y_\ell^{(n)} + \sum_{\ell=1}^{L} \hat{y}_\ell^{(n)}}$$

where for each instance n, $y_\ell^{(n)} = 1$ if label ℓ is a given label in ground truth; $\hat{y}_\ell^{(n)} = 1$ if label ℓ is a predicted label.

We compare our method with the following methods:

- **Binary Relevance (BR)** (Tsoumakas and Katakis, 2007) with both independent training and prediction;

- **Binary Relevance with support inference (BR-support)** (Wang et al., 2018) which trains binary classifiers independently but imposes label constraints at prediction time by only considering label sets observed during training, namely $\hat{\mathbf{y}} = \arg\max_{\text{observed } \mathbf{y}} \prod_{\ell=1}^{L} p(y_\ell|x)$;

- **Probabilistic Classifier Chain (PCC)** (Dembczynski et al., 2010) which transforms

the multi-label classification task into a chain of binary classification problems. Predictions are made with Beam Search.

- **Sequence to Sequence RNN (seq2seq-RNN)** (Nam et al., 2017) which maps each set to a sequence by decreasing label frequency and solves the multi-label task with an RNN designed for sequence prediction (see Table 1).

- **Vinyals-RNN-uniform, Vinyals-RNN-sample, and Vinyals-RNN-max** are three variants of RNNs proposed by (Vinyals et al., 2016). They are trained with different objectives that correspond to different transformations between sets and sequences. See Table 1 for a summary of their training objectives. Following the approach taken by (Vinyals et al., 2016), Vinyals-RNN-sample and Vinyals-RNN-max are initialized by Vinyals-RNN-uniform. We have also tested training Vinyals-RNN-max directly without having Vinyals-RNN-uniform as an initialization, and we name it as **Vinyals-RNN-max-direct**.

- **Sequence Generation Model (SGM)** (Yang et al., 2018) which trains the RNN model

similar to seq2seq-RNN but uses a new decoder structure that computes a weighted global embedding based on all labels as opposed to just the top one at each timestep.

In BR and PCC, logistic regressions with L1 and L2 regularizations are used as the underlying binary classifiers. seq2seq-RNN, PCC, and SGM rely on a particular label order. We adopt the decreasing label frequency order, which is the most popular choice.

4.2 Experimental Results

Table 3 shows the performance of different methods in terms of *label-F1* and *instance-F1*. The SGM results are taken directly from (Yang et al., 2018), and are originally reported only on AAPD dataset in terms of *hamming-loss* and *micro-F1*. Definitions of these two metrics can be found in (Koyejo et al., 2015).

Our method performs the best in all metrics on all datasets (except hamming loss on AAPD, see table 3). In general, RNN based methods perform better than traditional methods BR, BR-support and PCC. Among the Vinyals-RNN variants, Vinyals-RNN-max and Vinyals-sample work the best and have similar performance. However, they have to be initialized by Vinyals-RNN-uniform. Otherwise, the training gets stuck in early stage and the performance degrades significantly. One can see the clear degradation by comparing the Vinyals-RNN-max row (with initialization) with the Vinyals-RNN-max-direct row (without initialization). By contrast, our training objective in set-RNN does not suffer from this issue and can serve as a stable stand alone training objective.

On TheGuardian dataset, set-RNN performs slightly better than seq2seq-RNN in terms of instance-F1, but much better in terms of label-F1. It is known that instance-F1 is basically determined by the popular labels' performance while label-F1 is also sensitive to the performance on rare labels. Figure 1 shows that set-RNN predicts rare labels better than seq2seq-RNN.

Next we analyze how much benefit our new set prediction strategy brings in. For each RNN-based method, we test two prediction strategies: 1) finding the sequence with the highest probability and outputting the corresponding set (this is the default prediction strategy for all models except set-RNN); 2) outputting the set with the highest probability (this is the default prediction
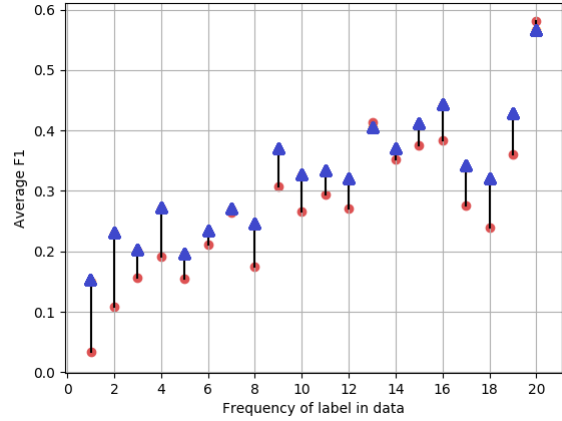

Figure 1: Average F1 over rare labels with the same frequency on TheGuardian dataset. Blue($\triangle$)=set-RNN, Red($\cdot$)=seq2seq-RNN.

strategy for set-RNN). Table 4 shows how each method performs with these two prediction strategies. One can see that Vinyals-RNN-uniform and set-RNN benefit most from predicting the top set, Vinyals-RNN-sample, Vinyals-RNN-max and Vinyals-RNN-max-direct benefit less, and seq2seq RNN does not benefit at all. Intuitively, for the top-set prediction to be different from the top-sequence prediction, the model has to spread probability mass across different sequence permutations of the same set.

4.3 Analysis: Sequence Probability Dsitribution

Results in Table 4 motivates us to check how sharply (or uniformly) distributed the probabilities are over different sequence permutations of the predicted set. We first normalize these sequence probabilities related to the predicted set and then compute the entropy. To make predictions with different set sizes (and hence different number of sequence permutations) comparable, we further divide the entropy by the logarithm of number of sequences. Smaller entropy values indicate a sharper distributions. The results are shown in Figure 2.

seq2seq-RNN trained with fixed label order and standard RNN objective (1) generates very sharp sequence distributions. It basically only assigns probability to one sequence in the given order. The entropy is close to 0. In this case, predicting the set is no different than predicting the top sequence (see Table 4). On the other extreme is Vinyals-RNN-uniform, trained with objective (3), which spreads probabilities across many se-
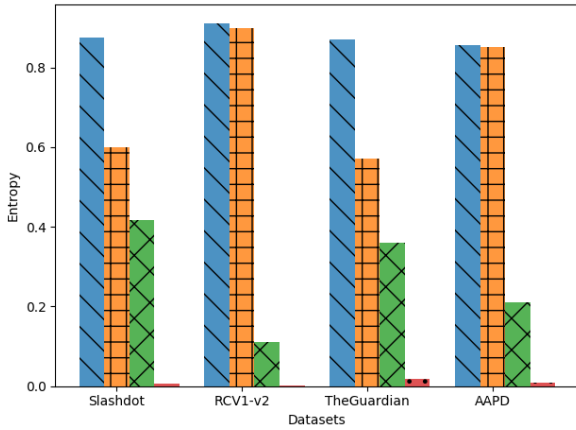
Figure 2: Entropy of sequence probability distribution for each model. Blue(\)=Vinyals-RNN-uniform, Orange(+)=set-RNN, Green(×)=Vinyals-RNN-max, Red(·)=seq2seq-RNN.

quences, and leads to the highest entropy among all models tested (the uniform distribution has the max entropy of 1). From Table 4, we see that by summing up sequence probabilities and predicting the most probable set, Vinyals-RNN-uniform's performance improves. But as discussed earlier, training with the objective (3) makes it impossible for the model to discover and concentrate on a particular natural label order (represented by a sequence). Overall Vinyals-RNN-uniform is not competitive even with the set-prediction enhancement. Between the above two extremes are Vinyals-RNN-max and set-RNN (we have omitted Vinyals-RNN-sample and Vinyals-RNN-max-direct here as they are similar to Vinyals-RNN-max). Both models are allowed to assign probability mass to a subset of sequences. Vinyals-RNN-max produces sharper sequence distributions than set-RNN, because Vinyals-RNN-max has the incentive to allocate most of the probability mass to the most probable sequence due to the max operator in its training objective (2). From Table 4, one can see that set-RNN clearly benefits from summing up sequence probabilities and predicting the most probable set while Vinyals-RNN-max does not benefit much. Therefore, the sequence probability summation is best used in both training and prediction, as in our proposed method.

Comparing 4 datasets in Table 4, we also see that Slashdot and TheGuardian, which have larger label cardinalities (therefore more permutations for one set potentially), benefit more from predicting the most probable set than RCV1 and AAPD, which have smaller label cardinalities.

5 Case Analysis

We further demonstrate how set-RNN works with two examples. In the first example from the RCV1-v2 dataset, the most probable set predicted by set-RNN (which is also the correct set in this example) does not come from the most probable sequence. Top sequences in decreasing probability order are listed in Table 5. The correct label set {forex, markets, equity, money markets, metals trading, commodity} has the maximum total probability of 0.161, but does not match the top sequence.

PROB	SEQUENCE
0.0236	equity, markets, money markets, forex
0.0196	**forex, markets, equity, money markets, metals trading, commodity**
0.0194	**equity, markets, forex, money markets, metals trading, commodity**
0.0159	**markets, equity, forex, money markets, metals trading, commodity**
0.0157	**forex, money markets, equity, metals trading, markets, commodity**
0.0153	**forex, money markets, markets, equity, metals trading, commodity**
0.0148	markets, equity, money markets, forex
0.0143	**money markets, equity, metals trading, commodity, forex, markets**
0.0123	**markets, money markets, equity, metals trading, commodity, forex**
0.0110	**markets, equity, forex, money markets, commodity, metals trading**
0.0107	**forex, markets, equity, money markets, commodity, metals trading**
0.0094	**forex, money markets, equity, markets, metals trading, commodity**

Table 5: The set-RNN predicted set (also the correct set) {forex, markets, equity, money markets, metals trading, commodity} has the max total probability of 0.161, but does not match the top sequence. Sequences for the correct set are in bold.

Next we demonstrate the issue with prescribing the sequence order in seq2seq-RNN with a TheGuardian example[4]. Figure 3 shows the predictions made by seq2seq-RNN and our method. In this particular example the top sequence agrees with the top set in our method's prediction so we can just analyze the top sequence. seq2seq-RNN predicts `Tate Modern` (incorrect but more popular label) while we predict `Tate Britain` (correct but less popular label). The seq2seq predicted sequence is in the decreasing label frequency order while our predicted sequence is not. In the training data, `Exhibition` is more frequent than `Tate Britain` and `Tate Modern`. If we arrange labels by decreasing frequency, `Exhibition` is immediately followed by `Tate Modern` 19 times, and by `Tate Britain` only 3 times. So it is far more likely to have `Tate Modern` than `Tate Britain` after `Exhibition`. However, at the set level, `Exhibition` and `Tate Modern` co-occurs 22 times while `Exhibition` and `Tate Britain`

---

[4]This document can be viewed at http://www.guardian.co.uk/artanddesign/jonathanjonesblog/2009/apr/08/altermodernism-nicolas-bourriaud
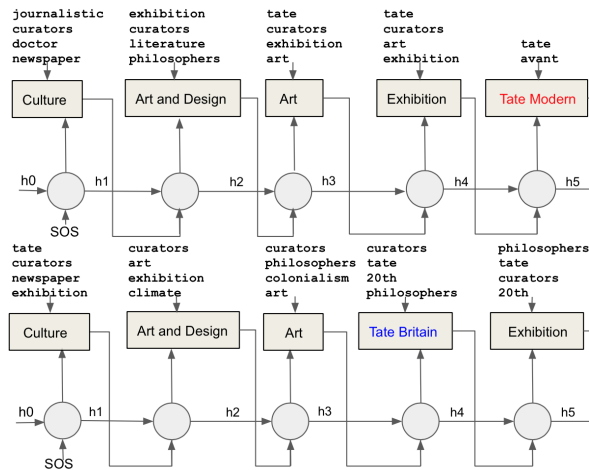
Figure 3: Top: best sequence by seq2seq-RNN; bottom: best sequence by set-RNN. Above models, at each time, we list the top unigrams selected by attention.

co-occurs 12 times, so the difference is not so dramatic. In this case, imposing the sequence order biases the probability estimation and leads to incorrect predictions.

6 Conclusion

In this work, we present an adaptation of RNN sequence models to the problem of multi-label classification for text. RNN only directly defines probabilities for sequences, but not for sets. Different from previous approaches, which either transform a set to a sequence in some pre-specified order, or relate the sequence probability to the set probability in some ad hoc way, our formulation is derived from a principled notion of set probability. We define the set probability as the sum of all corresponding sequence permutation probabilities. We derive a new training objective that maximizes the set probability and a new prediction objective that finds the most probable set. These new objectives are theoretically more appealing than existing ones, because they give the RNN model more freedom to automatically discover and utilize the best label orders.

Acknowledgements

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016.*, pages 265–283.

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

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

Shang-Fu Chen, Yi-Chen Chen, Chih-Kuan Yeh, and Yu-Chiang Frank Wang. 2018. Order-free RNN with visual attention for multi-label classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018.*

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. 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 1724–1734.

Krzysztof Dembczynski, Weiwei Cheng, and Eyke Hüllermeier. 2010. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 279–286.

Nadia Ghamrawi and Andrew McCallum. 2005. Collective multi-label classification. In *Proceedings of the 14th ACM international con-

ference on Information and knowledge management, pages 195–200. ACM.

Jiren Jin and Hideki Nakayama. 2016. Annotation order matters: Recurrent image annotator for arbitrary length image tagging. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pages 2452–2457.

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

Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Consistent multilabel classification. In *Advances in Neural Information Processing Systems*, pages 3321–3329.

Cheng Li, Bingyu Wang, Virgil Pavlu, and Javed Aslam. 2016. Conditional bernoulli mixtures for multi-label classification. In *International Conference on Machine Learning*, pages 2482–2491.

Weiwei Liu and Ivor Tsang. 2015. On the optimality of classifier chain for multi-label classification. In *Advances in Neural Information Processing Systems*, pages 712–720.

Bruce T Lowerre. 1976. The harpy speech recognition system. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J. Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5419–5429.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

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

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. *CoRR*, abs/1511.06391.

Bingyu Wang, Cheng Li, Virgil Pavlu, and Jay Aslam. 2018. A pipeline for optimizing f1-measure in multi-label text classification. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 913–918. IEEE.

Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. CNN-RNN: A unified framework for multi-label image classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2285–2294.

Jingya Wang, Xiatian Zhu, Shaogang Gong, and Wei Li. 2017a. Attribute recognition by joint recurrent learning of context and correlation. *CoRR*, abs/1709.08553.

Zhouxia Wang, Tianshui Chen, Guanbin Li, Ruijia Xu, and Liang Lin. 2017b. Multi-label image recognition by recurrently discovering attentional regions. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 464–472.

Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. 2018. SGM: sequence generation model for multi-label classification. *CoRR*, abs/1806.04822.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, and Jianfeng Lu. 2016. Multi-label image classification with regional latent semantic dependencies. *CoRR*, abs/1612.01082.