Maximizing Subset Accuracy with Recurrent Neural Networks in Multi-label Classification @ NIPS17

https://qdata.github.io/deep2Read

Presenter: Chao Jiang

Spring 2018

Outline

Task

Multi-label classification (MLC)

- Text classification task:
  - Article 1: news + science
  - Article 2: politics + novel
  - Article 3: news + sports
  - Article 4: ?
- Formally, a set of N samples $D = (x_n, y_n)_{n=1}^{N}$, each of which consists of an input $x \in X$ and its target $y \in Y$ which are assumed to be $i.i.d$ over a sample space $X \times Y$, learning a function f that maps inputs to subsets of a label set $L = \{1, 2, \cdots, \mathrm{L}\}$

Notation

Multi-label classification (MLC)

- A set of N samples $D = (x_n, y_n)_{n=1}^{N}$, each of which consists of an input $x \in X$ and its target $y \in Y$
- $L = \{1, 2, \cdots, \mathrm{L}\}$: label set, all possible labels
- $T_n = |y_n|$: the size of the label set associated to x_n, how many labels x_n have
- $C = \frac{1}{T} \sum_{n=1}^{N} T_n$: the cardinality of D, average number of labels one sample has, usually much smaller than L
- y could be viewed as a binary vector of size L, i.e., $\hat{y} \in \{0, 1\}^L$

Existing solutions

- label powerset (LP)
  - define each possible label combinations as a new class label, $S_L = \{\{1\}, \{1,2\}, \cdots \{1,2,3,\cdots,L\}\}$, then, addresses MLC as a multi-class classification problem with $min(N, 2^L)$ possible labels such that
  $$P(y_1, y_2, \cdots, y_L|x) \xrightarrow{LP} P(y_{LP} = k|x)$$
  - Disadvantage 1: training LP models becomes intractable for large-scale problems with an increasing number of labels in S_L
  - Disadvantage 2: even if the number of labels L is small enough, the problem is still prone to suffer from data scarcity because each label subset in LP will in general only have a few training instances
- binary relevance (BR)
  - Converting into a binary classification problem for each label

Probabilistic Classifier Chain (PCC)

- Using PCC to learn the joint probability of labels, which decomposes the joint probability into L conditionals:

$$P(y_1, y_2, \cdots, y_L|x) = \prod_{i=1}^{L} P(y_l|y_{<i}, x)$$

  where $y_{<i} = \{y_1, \cdots, y_{i1}\}$ denotes a set of labels that precede a label y_i.
- For training PCCs, L functions need to be learned independently to construct a probability tree with 2^L leaf nodes.

Probabilistic Classifier Chain (PCC)

- Disadvantage 1: obtaining the exact solution of such a probabilistic tree requires to find an optimal path from the root to a leaf node. A naive approach for doing so requires 2^L path evaluations in the inference step, and is therefore also intractable.
- Disadvantage 2: a cascadation of errors as the length of a chain gets longer
- Disadvantage 3: the classifiers f_i are trained independently according to a fixed label order, but in practice the label order in a chain has an impact on estimating the conditional probabilities

Outline

Formulating Predicting Subsets as Sequence Prediction

- LP and PCC could be viewed as means of subset accuracy maximization
- Similar to PCC, the joint probability can be computed as product of conditional probabilities, but unlike PCC, only T $\ll$ L terms are needed.
- Maximizing the joint probability of positive labels can be viewed as subset accuracy maximization, the joint probability of positive labels can be written as

$$P(\mathbf{y}_{p_1}, \mathbf{y}_{p_2}, \cdots, \mathbf{y}_{p_T} | \boldsymbol{x}) = \prod_{i=1}^{T} P(\mathbf{y}_{p_i} | \boldsymbol{y}_{<p_i}, \boldsymbol{x}).$$

- y can be represented as a set of 1-of-L vectors such that $y = \{y_{p_i}\}_{i=1}^{T}$ and $y_{p_i} \in R^L$ where T is the number of positive labels associated with an instance x

Formulating predicting subsets as sequence prediction

- this equation Eq.(3) has same form as prior equation EQ.(2) except for the number of output variables.
- While Eq.(2) is meant to maximize the joint probability over the entire 2^L configurations, Eq.(3) represents the probability of sets of positive labels and ignores negative labels.
- Advantage 1: the number of conditional probabilities to be estimated is dramatically reduced from L to T, search space from 2^L to L^T
- Advantage 2: reducing the length of the chain might be helpful in reducing the cascading errors

Why RNN?

- As each instance has a different value for T, they need MLC methods which is capable of dealing with a different number of output targets across instances.
- Outputs a stop label indicating the end of the sequence
- What's more, such probability chain rule (use previous output as next input) fits for RNN structure

Outline

Determining Label Permutations

- Because they use RNN to compute the joint probabilities, thus the order of the labels need to be determined as a prior.
- They tried four different orders of labels:
    - f2r: from frequent to rare labels
    - r2f: from rare to frequent labels
    - topological sorting: if the label is given as a tree, do DFS to determine the order of the labels
    - reverse topological sorting

Outline

Different RNN architectures

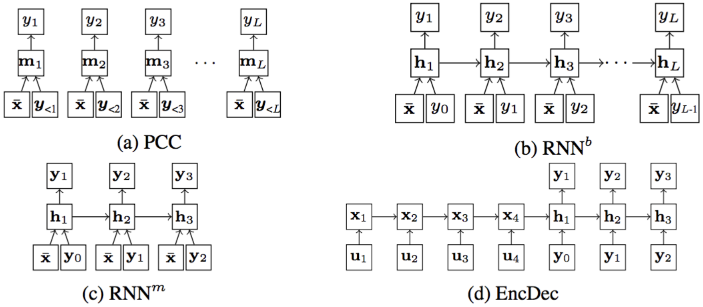
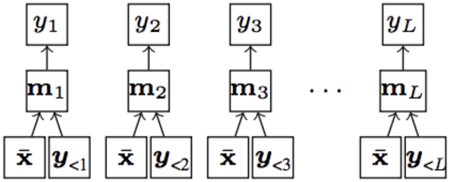


Figure 1: Illustration of PCC and RNN architectures for MLC. For the purpose of illustration, we assume $T = 3$ and $\boldsymbol{x}$ consists of 4 elements.

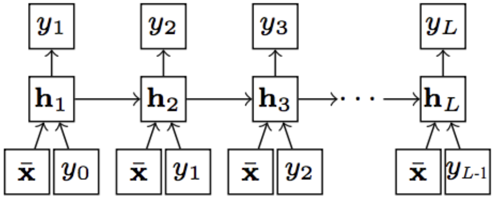
Different RNN architectures



(a) PCC

- Using neural networks to implement a probability classifier chain as baseline
- Using a fixed input representation computed from an instance x (average of all the word vectors).

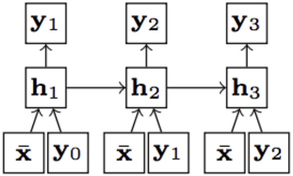
Different RNN architectures



(b) RNN^b

- A RNN architecture that learns a sequence of L binary targets can be seen as a NN counterpart of PCC because its objective is to maximize Eq.(2) just like in PCC
- output is binary, length is L
- Using gated recurrent units (GRUs)

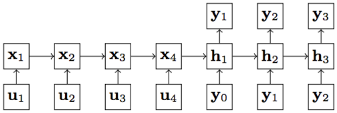
Different RNN architectures



(c) RNN^m

- A RNN architecture maximizing Eq.(3) to take advantage of the smaller label subset size T than L
- output size is L dimension, length is T
- Note that the key difference between RNN^b and RNN^m is whether target labels are binary targets y_i or 1-of-L targets y_i .
- Under the assumption that the hidden states hi preserve the information on all previous labels $y_{<i}$, learning RNN^b and RNN^m can be interpreted as learning classifiers in a chain

Different RNN architectures

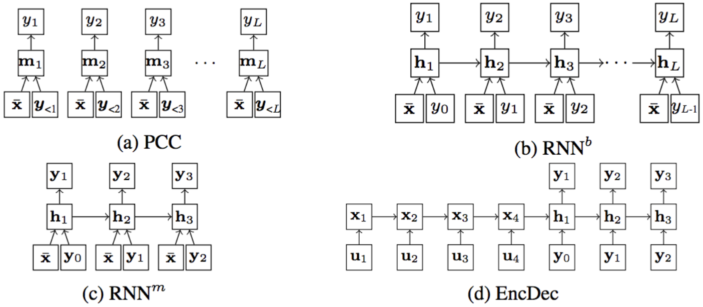


$\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$
$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$ $\mathbf{y}_0$ $\mathbf{y}_1$ $\mathbf{y}_2$

(d) EncDec

- An attention based encoder decoder network
- Indeed, EncDec is potentially more powerful than RNN^b and RNN^m because each prediction is determined based on the dynamic context of the input x unlike the fixed input representation $\overline{x}$ used in PCC, RNN^b and RNN^m

Different RNN architectures


y_1 y_2 y_3 y_L
$\mathbf{m}_1$ $\mathbf{m}_2$ $\mathbf{m}_3$ $\mathbf{m}_L$
$\bar{\mathbf{x}}$ $\boldsymbol{y}_{<1}$ $\boldsymbol{y}_{<2}$ $\boldsymbol{y}_{<3}$ $\boldsymbol{y}_{<L}$
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_L$
y_0 y_1 y_2 $y_{L\text{-}1}$
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$
$\mathbf{u}_1$ $\mathbf{u}_2$ $\mathbf{u}_3$ $\mathbf{u}_4$
$\mathbf{y}_0$ $\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$

(a) PCC

(b) RNN^b

(c) RNN^m

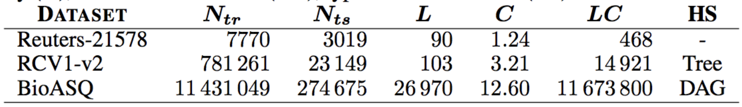
(d) EncDec

Figure 1: Illustration of PCC and RNN architectures for MLC. For the purpose of illustration, we assume $T = 3$ and $\boldsymbol{x}$ consists of 4 elements.

Experimental Setup

- Baselines:
  - BR(NN)
  - LP(NN)
  - PCC (NN) (beam search with beam size of 5 is used at inference time)
  - Another NN baseline, consider a feed-forward NN with binary cross entropy per label

Table 2: Summary of datasets. # training documents (N_{tr}), # test documents (N_{ts}), # labels (L), label cardinality (C), # label combinations (LC), type of label structure (HS).



DATASET	N_{tr}	N_{ts}	L	C	LC	HS
Reuters-21578	7770	3019	90	1.24	468	-
RCV1-v2	781 261	23 149	103	3.21	14 921	Tree
BioASQ	11 431 049	274 675	26 970	12.60	11 673 800	DAG

Evaluation Measures

- Example-based measures
  - Subset accuracy (ACC): $ACC(y, \hat{y}) = \frac{1}{L} \sum_{j=1}^{L} \mathbb{1}[y = \hat{y}]$
  - Hamming accuracy (HA) computes how many labels are correctly predicted in $\hat{y}$: $HA(y, \hat{y}) = \frac{1}{L} \sum_{j=1}^{L} \mathbb{1}[y_j = \hat{y}_j]$
  - example-based F1-measure (ebF1): $ebF_1(y, \hat{y}) = \frac{2 \sum_{j=1}^{L} y_j \hat{h}_j}{\sum_{j=1}^{L} y_j + \sum_{j=1}^{L} \hat{y}_j}$
- Label-based measures

$$\mathrm{mi}F_1 = \frac{\sum_{j=1}^{L} 2tp_j}{\sum_{j=1}^{L} 2tp_j + fp_j + fn_j} \quad (7)$$

$$\mathrm{ma}F_1 = \frac{1}{L} \sum_{j=1}^{L} \frac{2tp_j}{2tp_j + fp_j + fn_j} \quad (8)$$

Reuters-21578

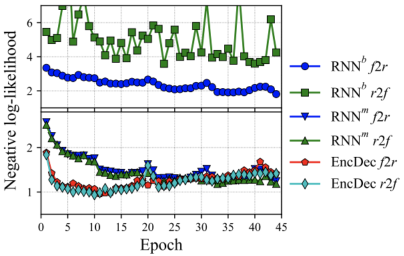


Figure 2: Negative log-likelihood of RNNs on the validation set of Reuters-21578.

- *RNN^b* has trouble with the r2f label ordering, because the predictions for later labels depend on sequences that are mostly zero when rare labels occur at the beginning. Hence, the model sees only few examples of non-zero targets in a single epoch.
- both *RNN^m* and EncDec converge relatively faster than *RNN^b* and do obviously not suffer from the r2f ordering.

Reuters-21578

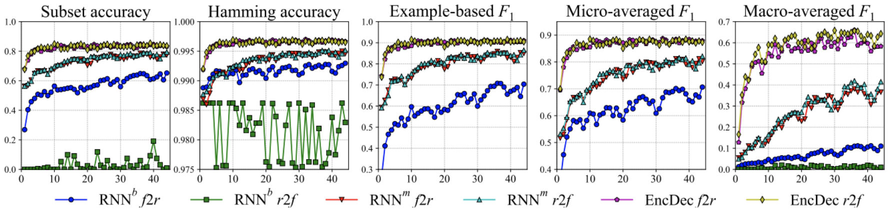


Figure 3: Performance of RNN models on the validation set of Reuters-21578 during training. Note that the x-axis denotes # epochs and we use different scales on the y-axis for each measure.

- no clear difference between the same type of models trained on different label permutations, except for RNN^b

Reuters-21578

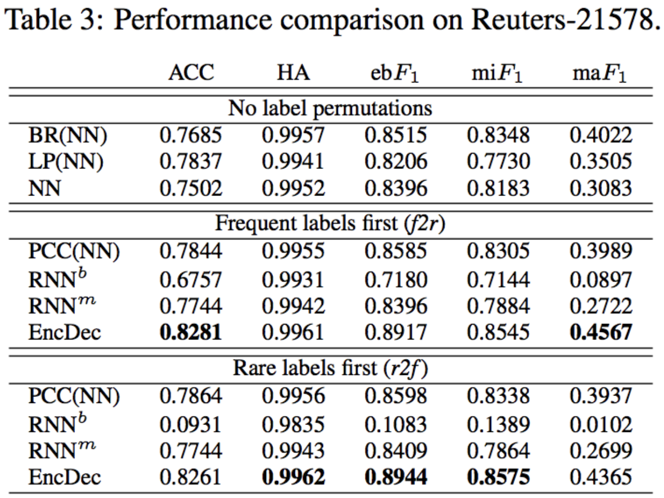
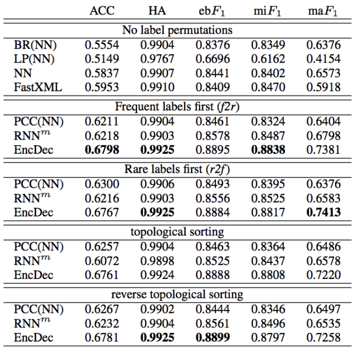


Table 3: Performance comparison on Reuters-21578.

	ACC	HA	ebF_1	miF_1	maF_1
			No label permutations		
BR(NN)	0.7685	0.9957	0.8515	0.8348	0.4022
LP(NN)	0.7837	0.9941	0.8206	0.7730	0.3505
NN	0.7502	0.9952	0.8396	0.8183	0.3083
			Frequent labels first (*f2r*)		
PCC(NN)	0.7844	0.9955	0.8585	0.8305	0.3989
RNN^b	0.6757	0.9931	0.7180	0.7144	0.0897
RNN^m	0.7744	0.9942	0.8396	0.7884	0.2722
EncDec	**0.8281**	0.9961	0.8917	0.8545	**0.4567**
			Rare labels first (*r2f*)		
PCC(NN)	0.7864	0.9956	0.8598	0.8338	0.3937
RNN^b	0.0931	0.9835	0.1083	0.1389	0.0102
RNN^m	0.7744	0.9943	0.8409	0.7864	0.2699
EncDec	0.8261	**0.9962**	**0.8944**	**0.8575**	0.4365

RCV1-v2

Table 4: Performance comparison on RCV1-v2.

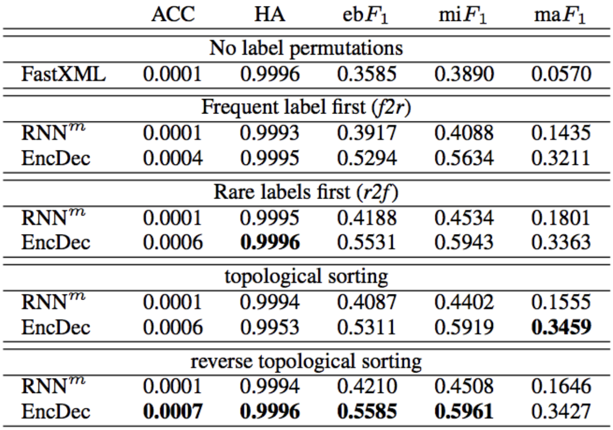


	ACC	HA	ebF_1	miF_1	maF_1
			No label permutations		
BR(NN)	0.5554	0.9904	0.8376	0.8349	0.6376
LP(NN)	0.5149	0.9767	0.6696	0.6162	0.4154
NN	0.5837	0.9907	0.8441	0.8402	0.6573
FastXML	0.5953	0.9910	0.8409	0.8470	0.5918
			Frequent labels first (*f2r*)		
PCC(NN)	0.6211	0.9904	0.8461	0.8324	0.6404
RNN^m	0.6218	0.9903	0.8578	0.8487	0.6798
EncDec	**0.6798**	**0.9925**	0.8895	**0.8838**	0.7381
			Rare labels first (*r2f*)		
PCC(NN)	0.6300	0.9906	0.8493	0.8395	0.6376
RNN^m	0.6216	0.9903	0.8556	0.8525	0.6583
EncDec	0.6767	**0.9925**	0.8884	0.8817	**0.7413**
			topological sorting		
PCC(NN)	0.6257	0.9904	0.8463	0.8364	0.6486
RNN^m	0.6072	0.9898	0.8525	0.8437	0.6578
EncDec	0.6761	0.9924	0.8888	0.8808	0.7220
			reverse topological sorting		
PCC(NN)	0.6267	0.9902	0.8444	0.8346	0.6497
RNN^m	0.6232	0.9904	0.8561	0.8496	0.6535
EncDec	0.6781	**0.9925**	**0.8899**	0.8797	0.7258

- Also comparing with FastXML, one of state-of-the-arts in extreme MLC.
- FastXML is designed to maximize top-k ranking measures such as preck for which the performance on frequent labels is important.

BioASQ

Table 5: Performance comparison on BioASQ.



	ACC	HA	ebF_1	miF_1	maF_1
			No label permutations		
FastXML	0.0001	0.9996	0.3585	0.3890	0.0570
			Frequent label first (*f2r*)		
RNN^m	0.0001	0.9993	0.3917	0.4088	0.1435
EncDec	0.0004	0.9995	0.5294	0.5634	0.3211
			Rare labels first (*r2f*)		
RNN^m	0.0001	0.9995	0.4188	0.4534	0.1801
EncDec	0.0006	**0.9996**	0.5531	0.5943	0.3363
			topological sorting		
RNN^m	0.0001	0.9994	0.4087	0.4402	0.1555
EncDec	0.0006	0.9953	0.5311	0.5919	**0.3459**
			reverse topological sorting		
RNN^m	0.0001	0.9994	0.4210	0.4508	0.1646
EncDec	**0.0007**	**0.9996**	**0.5585**	**0.5961**	0.3427