(EMNLP, 2019)
Learning to Learn and Predict: A Meta-Learning Approach for Multi-Label Classification

Jiawei Wu and **Wenhan Xiong** and **William Yang Wang**
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
{jiawei_wu,xwhan,william}@cs.ucsb.edu

51194501029
曹鑫磊

Problem Definition: Multi-Label Classification

Test: Binary classification for each class (threshold=0.5)

Train: N-class cross-entropy loss with multiple labels

$$L = -\frac{1}{N}\sum_i [y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)]$$

Label: [1.0, 0.0, 0.0, 1.0, 0.0]

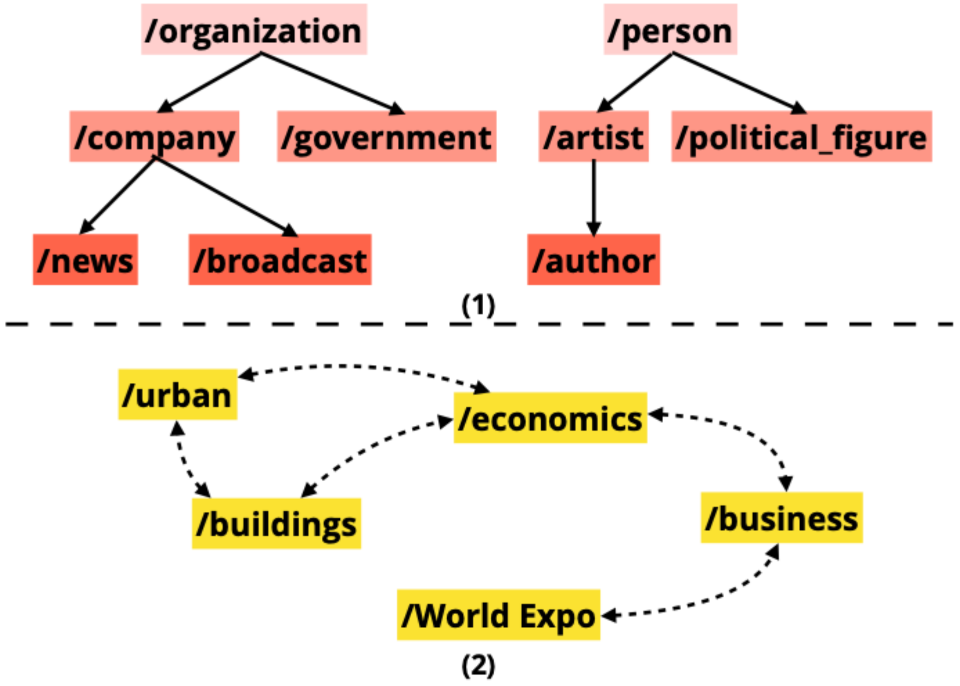
$\updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow \quad \updownarrow$

Prediction: [0.8, 0.6, 0.2, 0.3, 0.7]

Evaluation:

- Strict accuracy
- Loose micro-F1, calculate precision, recall, f1 globally
- Loose macro-F1, calculate F1 per class, and average over them

Motivation: explicit & implicit dependencies between labels



Related Work

Multi-Label Classification

- Optimal <u>Prediction</u> Policy: Select thresholds for each category separately
- Hierarchy-Aware <u>Training</u> Policy: Different level, Different weight

Meta-Learning

- Learning a meta-policy to update model parameters ☑
- Learning a good parameter initialization for fast adaptation

Contribution of this paper

- Learning training & prediction policy for each category dynamically
- model-agnostic method

Method

- As a RL Framework

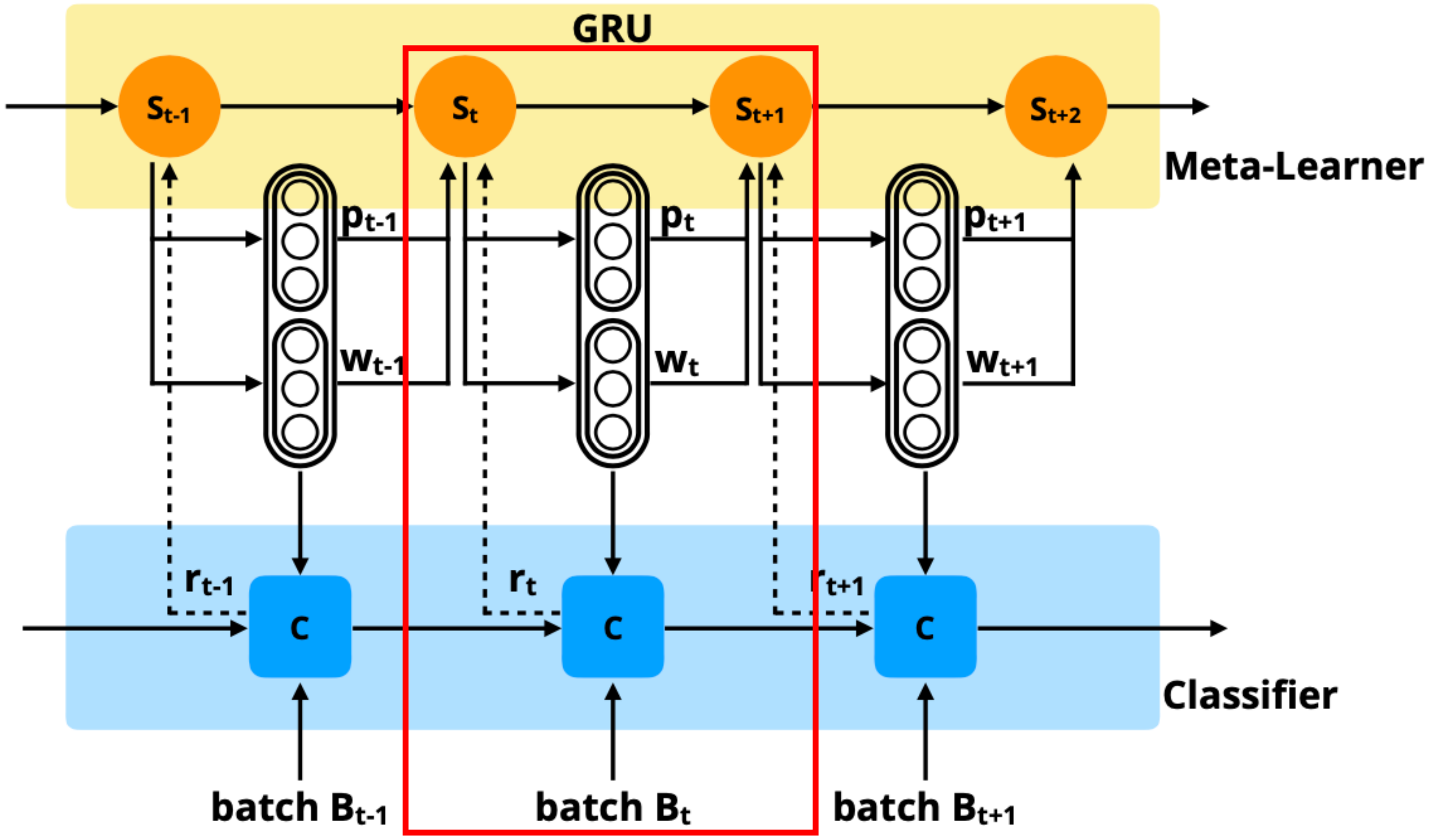


Figure 2: The meta-learning framework for multi-label classification.

Classifier

- Classifier C
- N-class multi-label classification
- training policy $\boldsymbol{w} = \left(w^{(1)}, w^{(2)}, \cdots, w^{(N)}\right)$, standard: $\left(\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}\right)$
- prediction policy $\boldsymbol{p} = \left(p^{(1)}, p^{(2)}, \cdots, p^{(N)}\right)$, standard: $(0.5, 0.5, \cdots, 0.5)$

$$L(\theta_t^C) = -\sum_i^{B_t} \sum_j^N \underline{w_t^{(j)}} N \left\{ y_i^{*(j)} \log y_i^{(j)} + \left(1 - y_i^{*(j)}\right) \log\left(1 - y_i^{(j)}\right) \right\}$$

Meta-Learner

State: $s_t = \text{GRU}\left(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix}\right)$

Action:

- $w_t = \text{softmax}(W_w s_t + b_w)$
- $p_t = \text{sigmoid}(W_p s_t + b_p)$

Reward: $\mathrm{r_t} = \sum_{\mathrm{i}}^{\mathrm{B_t}} \sum_{j=1}^{N} (-1)^{y_i^{*(j)}} \frac{p_t^{(j)} - y_i^{(j)}}{p_t^{(j)}}$

- How much is prediction closer to ground truth than threshold(%)?
- ground truth is 1, threshold should be close to 0.
- ground truth is 0, threshold should be close to 1.

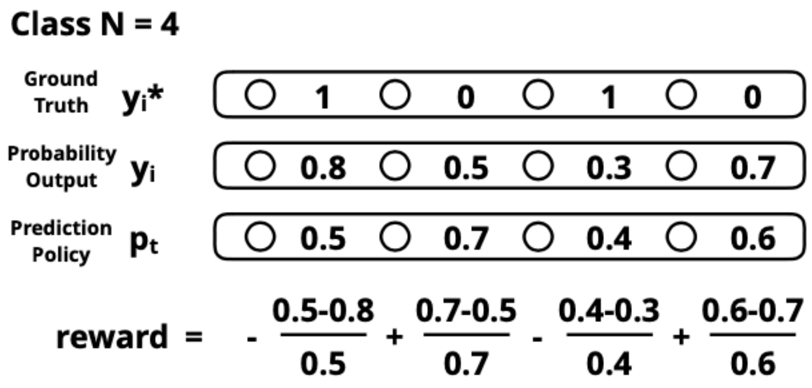


Figure 3: A example about the computation process of reward (one sample with class $N = 4$).

Training Procedure

$$J(\theta_{meta}) = \mathbb{E}_{\pi}\left[\sum_{t=1}^{T} r_t\right]$$

Algorithm 1: The algorithm of our meta-learning framework.

Given a set of labeled training data U
Given a untrained classifier C
for *episode* $\leftarrow$ *1* **to** M **do**
  Initialize $w_0 \leftarrow (\frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N}) \in \mathbb{R}^N$
  Initialize $p_0 \leftarrow (0.5, 0.5, \cdots, 0.5) \in \mathbb{R}^N$
  for *time step* $t \leftarrow$ *1* **to** T **do**
    $s_t \leftarrow \mathrm{GRU}(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix})$
    $w_t \leftarrow \mathrm{softmax}(W_w s_t + b_w)$
    $p_t \leftarrow \mathrm{sigmoid}(W_p s_t + b_p)$
    Sample a batch B_t from U
    Update C using B_t with w_t-based objective function in Equation 1
    Compute reward r_t with p_t in Equation 6
  Update θ_{meta} using $g \propto \nabla_\theta J(\theta_{meta})$

Before and during Test

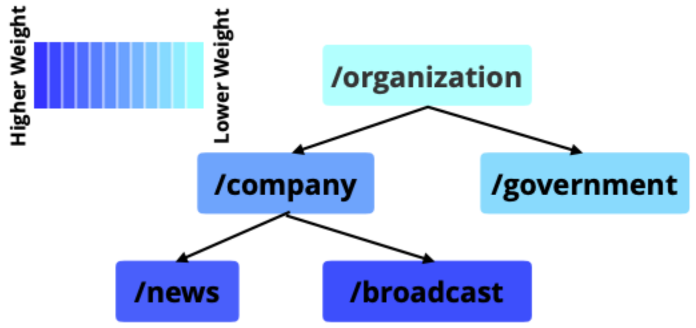
1. Rerun for one more episode, but using the whole training set as a batch at each time step.
2. The generated policies w_T and p_T is chosen as the final policies.
3. Train a classifier with the w_T-weighted cross-entropy objective function
4. Test based on the prediction policy p_T.

Experiment: Fine-grained Entity Typing

Datasets	FIGER			OntoNotes			BBN		
Metrics	**Acc.**	**Macro**	**Micro**	**Acc.**	**Macro**	**Micro**	**Acc.**	**Macro**	**Micro**
SOTA	0.659	0.807	0.770	0.521	0.686	0.626	0.655	0.729	0.751
SOTA+Hier-Training	0.661	0.812	0.773	0.532	0.690	0.640	0.657	0.735	0.754
SOTA+Meta-Training	0.670	0.817	0.779	0.539	0.702	0.648	0.662	0.736	0.761
SOTA+ScutFBR-Prediction	0.662	0.814	0.782	0.542	0.695	0.650	0.661	0.736	0.758
SOTA+ODR-Prediction	0.669	0.818	0.782	0.537	0.703	0.648	0.664	0.738	0.764
SOTA+Meta-Prediction	0.674	0.823	0.786	0.544	0.709	0.657	0.671	0.744	0.769
Predictions-as-Features	0.663	0.816	0.785	0.544	0.699	0.655	0.663	0.738	0.761
Subset Maximization	0.678	0.827	0.790	0.546	0.713	**0.661**	0.673	0.748	0.772
SOTA+Meta-Training-Prediction	**0.685**	**0.829**	**0.794**	**0.552**	**0.719**	**0.661**	**0.678**	**0.752**	**0.775**

Table 2: The experimental results on fine-grained entity typing datasets. Acc.: Accuracy.

- Predictions-as-Features: Train a classifier for each label, take predictions produced by the former classifiers as features
- Subset Maximization: View the problem as classifier chains, and use RNN



Experiment: Text Classification

Datasets	Reuters-21578			RCV1-V2		
Metrics	**Accuracy**	**Macro-F1**	**Micro-F1**	**Accuracy**	**Macro-F1**	**Micro-F1**
CNN	0.537	0.472	0.841	0.616	0.642	0.838
CNN+Meta-Training	0.542	0.476	0.843	0.631	0.655	0.852
CNN+ScutFBR-Prediction	0.549	0.477	0.849	0.634	0.651	0.856
CNN+ODR-Prediction	0.541	0.475	0.848	0.630	0.653	0.849
CNN+Meta-Prediction	0.549	0.479	0.851	0.639	0.658	0.857
Predictions-as-Features	0.539	0.476	0.845	0.621	0.644	0.847
Subset Maximization	0.543	0.478	0.849	0.632	0.660	0.859
CNN+Meta-Training-Prediction	**0.556**	**0.483**	**0.854**	**0.647**	**0.669**	**0.864**

Table 5: The experimental results on text classification datasets.

THANKS