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

Abstract

Many tasks in natural language processing can be viewed as multi-label classification problems. However, most of the existing models are trained with the standard cross-entropy loss function and use a fixed prediction policy (e.g., a threshold of 0.5) for all the labels, which completely ignores the complexity and dependencies among different labels. In this paper, we propose a meta-learning method to capture these complex label dependencies. More specifically, our method utilizes a meta-learner to jointly learn the training policies and prediction policies for different labels. The training policies are then used to train the classifier with the cross-entropy loss function, and the prediction policies are further implemented for prediction. Experimental results on fine-grained entity typing and text classification demonstrate that our proposed method can obtain more accurate multi-label classification results.

1 Introduction

Multi-label classification aims at learning to make predictions on instances that are associated with multiple labels simultaneously, whereas in a classic multi-class classification setting, typically one instance has only one label. Multi-label classification is a common learning paradigm in a large amount of real-world natural language processing (NLP) applications, such as fine-grained entity typing (Ling and Weld, 2012; Shimaoka et al., 2017; Abhishek et al., 2017; Xin et al., 2018) and text classification (Nam et al., 2014; Liu et al., 2017; Chen et al., 2017; Wu et al., 2018).

Significant amounts of research studies have been dedicated to tackle the multi-label classification problem (Zhang and Zhou, 2014), from traditional statistical models (Zhang and Zhou, 2007; Zhou et al., 2012; Surdeanu et al., 2012) to neural network-based models (Nam et al., 2014; Shimaoka et al., 2017). These models have variable structures, but usually, share the standard cross-entropy loss function for training. After training, these models tend to use a single prediction policy for all the labels to generate the final predictions. Actually, the above process is based on the assumption that there is no dependency among the labels. However, as shown in Figure 1, this assumption is hard to be satisfied in real-world datasets, and the complex label dependencies receive little attention in multi-label classification (Dembszynski et al., 2010; Dembczyński et al., 2012). Owing to the impact of the label dependencies, although one object may have multiple labels simultaneously in the multi-label classification setting, the level of difficulty of prediction for different labels can vary a lot. Firstly, for those labels that are aligned with knowledge graphs, they usually indicate a hierarchical dependency struc-

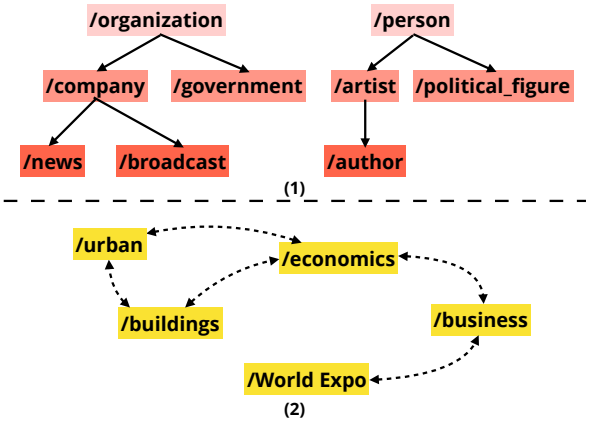

Figure 1: Illustration of complex label dependencies in multi-label classification. (1) Labels are aligned with some knowledge graphs. In this case, labels are organized in a hierarchical dependency structure. (2) Even if labels do not demonstrate explicit correlation, they can still have implicit semantic dependencies.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4354–4364,
Hong Kong, China, November 3–7, 2019.

ture (Ling and Weld, 2012; Ren et al., 2016; Ruder et al., 2016). It is intuitive that one trained classifier is easier to distinguish high-level parent labels, such as `/organization` and `/person`, but harder to distinguish low-level child labels, such as `/news` and `/broadcast`. Secondly, for those cases where labels do not demonstrate explicit correlation, the labels still contain implicit semantic dependencies, which is extremely common in the NLP field. For instance, the label `/urban` and `/economics` have obvious semantic correlation even if they are not organized in a hierarchical structure. Meanwhile, the labels with more implicit dependencies are easier to predict because they expose more semantic information during the training. These intuitions inspire our work on learning different training policies and prediction policies for different labels.

The training policies and prediction policies for all the labels can be viewed as a series of hyper-parameters. However, to learn high-quality policies, one needs to specify both explicit and implicit label dependencies, which is not manually realistic. To resolve both issues mentioned above, we propose a meta-learning framework to model these label dependencies and learn training and prediction policies automatically. Concretely, we introduce a joint learning formulation of the meta-learning method and multi-label classification. A gated recurrent unit (GRU)-based (Chung et al., 2014) meta-learner is implemented to capture the label dependencies and learn these hyper-parameters during the training process of a classifier. Empirically, we show our method outperforms previous related methods on fine-grained entity typing and text classification problems. In summary, our contributions are three-fold:

- We are the first to propose a joint formulation of "learning to learn" and "learning to predict" in a multi-label classification setting.
- Our learning method can learn a weight and a decision policy for each label, which can then be incorporated into training and prediction.
- We show that our method is model-agnostic and can apply to different models in multi-label classification and outperform baselines.

In Section 2, we outline related work in multi-label classification and meta-learning. We then describe our proposed method in Section 3. We show experimental results in Section 4. Finally, we conclude in Section 7.

2 Related Work

2.1 Multi-Label Classification

Multi-label classification assigns instances with multiple labels simultaneously (Tsoumakas et al., 2006). (Shore and Johnson, 1980; De Boer et al., 2005) introduce prediction policies that weight the training loss function with external knowledge. However, in real-world multi-label classification, it is hard to obtain knowledge to determine prediction policies. As for the prediction policies, Yang (2001) select the thresholds that achieve the best evaluation measure on a validation set, while Lewis et al. (2004) utilize a cross-validation method to determine the thresholds. Lipton et al. (2014) propose an optimal decision rule to maximize F1 measure with thresholds. However, most of the previous methods can be viewed as post-processing because the prediction policies are computed after training. Meanwhile, the ability of prediction policies to help train the classifier is less explored.

Compared to previous related methods, we propose a principled approach that learns a training policy and a prediction policy for each label automatically with modeling label dependencies implicitly. Furthermore, the prediction policies are learned during the training process instead of the post-processing, which can also help to train a better classifier.

2.2 Meta-learning

Meta-learning is a "learning to learn" method, in which a learner learns new tasks and another meta-learner learns to train the learner (Bengio et al., 1990; Runarsson and Jonsson, 2000; Thrun and Pratt, 2012). There are two types of meta-learning:

- learning a meta-policy to update model parameters (Andrychowicz et al., 2016; Mishra et al., 2018).
- learning a good parameter initialization for fast adaptation (Duan et al., 2016; Vinyals et al., 2016; Finn et al., 2017; Snell et al., 2017; Gu et al., 2018).

In this paper, we propose to extend meta-learning algorithm for multi-label classification based on the first category. Instead of only training the

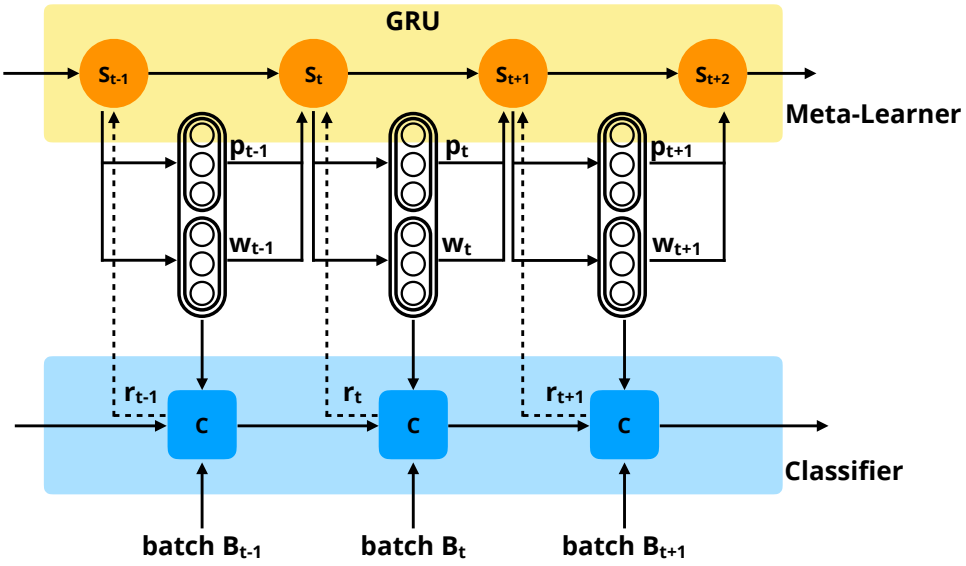

Figure 2: The meta-learning framework for multi-label classification.

model with meta-policy, we also consider prediction with meta-policy.

3 Method

In this section, we describe our meta-learning framework for learning and predicting multi-label classification in detail. In multi-label classification, the conventional methods usually learn classifiers with the standard cross-entropy loss function. After training, a single prediction policy (usually, a threshold of 0.5) is applied to all labels to generate the prediction. However, as mentioned in Figure 1, these methods ignore the explicit and implicit label dependencies among the labels. To improve the performance of multi-label classification, our main idea is to learn high-quality training policies and prediction policies by meta-learning, which can also model the label dependencies.

We view the training policies and prediction policies as a series of hyper-parameters, and then formulate the training of the classifier as a sequential process. At each iteration (time step t), the classifier updates its parameters with one sampled batch. The exploration of this classifier's hyper-parameters can be modeled by a meta-learner. We first briefly introduce the classifier model used in multi-label classification. Then we describe the meta-learner, and the interaction between the meta-learner and the classifier. Finally, we explain how to train the meta-learner in our unified framework.

3.1 Classifier Model

As mentioned in Section 1, many different model structures have already been explored in multi-label classification. Our framework is model-agnostic, which means different model structures are compatible if only they choose the cross-entropy loss function as the objective function. The classifier is represented as C. For a N- class multi-label classification, we represent the training policy as the vector $w = (w^{(1)}, w^{(2)}, \cdots, w^{(N)})$ and the prediction policy as the vector $p = (p^{(1)}, p^{(2)}, \cdots, p^{(N)})$, where $w^{(i)}$ and $p^{(i)}$ are the training weight and predicting threshold for the i-th class. w_t and p_t refer the weight vector and threshold vector at time step t. Then the goal of our framework is to learn a high-quality w and p for a certain classifier C.

To update the parameters of the classifier C, at each time step t, we sample a batch B_t from the training set U. We then set a weighted cross-entropy objective function to update the C, which is defined as:

$$L(\theta_t^C) = -\sum_{i}^{B_t}\sum_{j}^{N} w_t^{(j)} N\{y_i^{*(j)} \log y_i^{(j)} + (1 - y_i^{*(j)}) \log(1 - y_i^{(j)})\}, \quad (1)$$

where y_i^* indicates the ground truth prediction of the i-th sample from the B_t, and $y_i^{(j)}$ is the j-th entry of the corresponding output vector y_i. The

standard cross-entropy loss function is a special case when $w_t^{(j)} = \frac{1}{N}(j = 1, 2, \cdots, N)$.

3.2 Meta-Learner

Meta-learning is a widely used reinforcement learning method to learn the meta-information of a machine learning system (Bengio et al., 1990; Runarsson and Jonsson, 2000; Thrun and Pratt, 2012). The core of our proposed method is a meta-learner, which is trained to learn a training and a prediction policy for multi-label classification. At each time step t, the meta-learner observes the current state s_t, and then generate a training policy w_t and a prediction policy p_t. Based on the policies w_t and p_t, the parameters of classifier C can be update with the sampled batch B_t as described in Section 3.1. After training, the meta-learner receives a reward r_t. The goal of our meta-learner at each time step t is to choose the two policies w_t and p_t that can maximize the future reward

$$R_t = \sum_{t'=t}^{T} r_{t'}, \tag{2}$$

where a training episode terminates at time T.

3.2.1 State Representation

The state representation, in our framework, is designed to connect the policy generation and the environment. At each time step t, the training policy, and the prediction policy are generated based on the state s_t. A reward will be computed based on the change of the environment. In our case, the performance change of the classifier C. In order to successfully explore the policy space and generate high-quality policies, the meta-learner needs to remember what similar policies have already been tried and make further generation based on these memories.

Based on the above intuition, we formulate the meta-learner as a recurrent neural network (RNN)-based structure. To simplify our method, we use a GRU in our experiments. The state representation s_t is directly defined as the hidden state h_t of the GRU at time step t. The s_t is computed according to:

$$s_t = \text{GRU}(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix}), \tag{3}$$

where the input of GRU at time step t is the concatenation of the prediction policy and training policy generated at time step $t-1$.

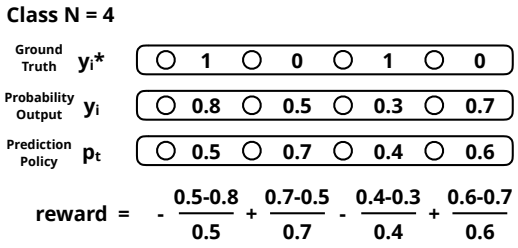

Figure 3: A example about the computation process of reward (one sample with class $N = 4$).

3.2.2 Policy Generation

At each time step t, the meta-learner can generate two policies, the training policy w_t and the prediction policy p_t. As mentioned in Section 3.1, both w_t and p_t are represented as a N-dimensional vector format.

To incorporate the training policy w_t into the cross-entropy objective function in Equation 1 and keep the training gradients of the classifier in the same magnitude during the whole training episode, the condition $\sum_i^N w_t^{(i)} = 1$ for w_t must be satisfied. Thus, at each time step t, the training policy is generated as:

$$w_t = \text{softmax}(W_w s_t + b_w). \tag{4}$$

As for the prediction policy, it is obvious that $p_t^{(i)} \in (0, 1)$ must be satisfied for $i = 1, 2, \cdots, N$. Then we define the prediction policy as:

$$p_t = \text{sigmoid}(W_p s_t + b_p). \tag{5}$$

The s_t is the state representation of the meta-learner at time step t. W_w, b_w, W_p and b_p all are learnable parameters.

3.2.3 Reward Function

The meta-leaner is trained to generate high-quality training and prediction policies jointly to improve the performance of the classifier C. To capture this performance change, we design a reward function based on the probability distributions of samples.

At each time step t, we first generate the training policy w_t and the prediction policy p_t according to Section 3.2.2. Then a batch B_t is sampled from the training set and used to update the classifier C based on Equation 1. We evaluate the output probability distribution of all the samples from B_t on the classifier C and compute the reward as:

$$r_t = \sum_{i}^{B_t} \sum_{j=1}^{N} (-1)^{y_i^{*(j)}} \frac{p_t^{(j)} - y_i^{(j)}}{p_t^{(j)}}, \tag{6}$$

where y_i is the output probability vector of the i-th sample from the batch B_t and y_i^* is the corresponding ground truth vector. The superscript (j) represents the j-th entry of a vector. A simple example about how the reward is computed is shown in Figure 3.

3.3 Training and Testing

The θ_{meta} is the set of all the parameters in the meta-learner, and the parameters can be trained by maximizing the total expect reward. The expected reward for an episode terminating at time step T is defined as:

$$J(\theta_{meta}) = \mathbb{E}_{\pi}[\sum_{t=1}^{T} r_t]. \quad (7)$$

$J(\theta_{meta})$ can be maximized by directly applying gradient-based optimization methods. We optimize it using policy gradient descent and implement a reward baseline to lower the variance during the training (Sutton, 1984). The details of the training process are shown in Algorithm 1. During the training, the reason why we use sample batches instead of the full training set at each time step is that we want to explore more policy space with diversity instead of iteratively fitting the whole training set.

At test time, we rerun the meta-learner and the classifier simultaneously for one episode. However, we use the whole training set as a batch at each time step. The generated policies w_T and p_T is chosen as the final policies. We then train a classifier with the w_T-weighted cross-entropy objective function and test it based on the prediction policy p_T.

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**
    $s_t \leftarrow \text{GRU}(s_{t-1}, \begin{bmatrix} p_{t-1} \\ w_{t-1} \end{bmatrix})$
    $w_t \leftarrow \text{softmax}(W_w s_t + b_w)$
    $p_t \leftarrow \text{sigmoid}(W_p s_t + b_p)$
    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 $g \propto \nabla_\theta J(\theta_{meta})$

4 Experimental Setups

We evaluate our proposed method in following two settings: (1) **Fine-grained Entity Typing**, where the labels have explicit hierarchical dependencies; (2) **Text Classification**, where one needs to model the implicit label dependencies.

4.1 Baselines

Since our method is model-agnostic, we directly employ the state-of-the-art (SOTA) models for both two tasks. The details of SOTA models will be discussed in Section 5 and 6. We compare our method with multiple baselines:

- **Hierarchy-Aware Training Policy**: To explicitly add hierarchical information during the training, the labels are first given integer weights. For instance, the first-level (parent) labels are given weight 1 while the third-level labels are given 3. All the integer weights then are normalized to add into the cross-entropy loss function in Equation 1.

- **SCutFBR Prediction Policy**: The rank-based policy method RCut and proportion-based assignments PCut are jointly considered to set prediction policies for different labels after obtaining the trained classifier (Yang, 2001).

- **ODR Prediction Policy**: After training, an optimal decision rule is to implement to choose prediction policies based on maximizing micro F1 scores (Lipton et al., 2014).

- **Predictions-as-Features**: The model trains a classifier for each label, organize the classifiers in a partially ordered structure, and take predictions produced by the former classifiers as the latter classifiers' features (Li et al., 2015).

- **Subset Maximization**: The model views the multi-label prediction problem as classifier chains, and then replace classifier chains with recurrent neural networks (Nam et al., 2017).

Datasets	FIGER	OntoNotes	BBN
#Types	128	89	47
Max Hierarchy Depth	2	3	2
#Training	2,690,286	220,398	86,078
#Testing	563	9,603	13,187

Table 1: The statistics of entity typing datasets.

5 Fine-grained Entity Typing

Entity type classification is the task for assigning semantic types to entity mentions based on their context. For instance, the system needs label the entity *"San Francisco Bay"* as `/location`, `/location/region` based on its context *"... the rest of San Francisco Bay, a spot ..."*.

In a fine-grained classification setting, entities are aligned with knowledge graphs (Ling and Weld, 2012), which typically makes labels be arranged in a hierarchical structure. We utilize our meta-learning framework to tackle this problem and evaluate our method.

Datasets We evaluate our method on three widely-used fine-grained entity typing datasets, FIGER, OntoNotes, and BBN, which are preprocessed by Ren et al. (2016). The statistics of these datasets are listed in Table 1.

FIGER: The training data is automatically generated by distant supervision, and then aligned with Freebase. The test data is collected from news reports and manually annotated by Ling and Weld (2012).

OntoNotes: The training sentences are collected from OntoNotes text corpus (Weischedel et al., 2013), and linked to Freebase. Gillick et al. (2014) releases a manually annotated test dataset.

BBN: The dataset consists of sentences from Wall Street Journal articles, which is entirely manually annotated (Weischedel and Brunstein, 2005).

Implementation Details We choose the SOTA model of fine-grained entity typing as the classifier C (Shimaoka et al., 2017; Abhishek et al., 2017). The overall model structure is shown in Figure 4. Given en entity and its context sentence, the words are initialized with word embeddings (Mikolov et al., 2013). The mention representation is simply computed by averaging the word embeddings of entity mention words. As for the context representation, a bidirectional LSTM and attention mechanism are used to encode left and right context representation separately. The attention operation is computed using a Multi-Layer Perceptron (MLP) as follows:

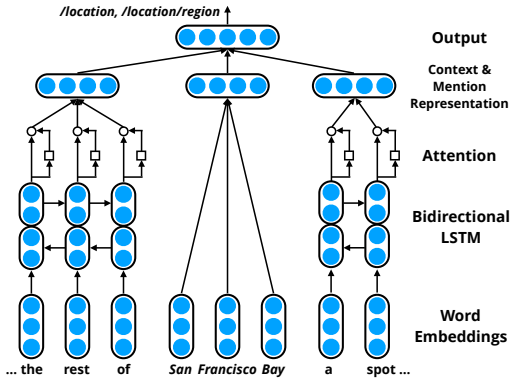

Figure 4: The SOTA model structure for fine-grained entity type classification (Shimaoka et al., 2017; Abhishek et al., 2017).

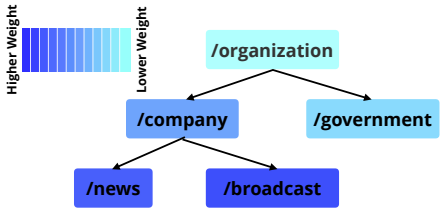

Figure 5: A case study about weights. See Section 5 for the detailed explanation.

$$a_i = \sigma\{W_1 \tanh(W_2 \begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix})\}, \quad (8)$$

where W_1 and W_2 are parameter matrices of the MLP, and $\begin{bmatrix} \overrightarrow{h}_i \\ \overleftarrow{h}_i \end{bmatrix}$ is the hidden state of Bi-LSTM at the i-th position. Note that the parameters of MLP are shared by all the entities. The left, right context representations and mention representation are concatenated as the feature vector. A softmax layer is then implemented to perform final prediction. The standard cross-entropy loss function is used and the thresholds of all the labels are set as 0.5. To avoid overfitting, we employ dropout operation on entity mention representation.

Following the previous researches (Ling and Weld, 2012; Ren et al., 2016), we use strict accuracy, loose macro, and loose micro scores to evaluate the model performance.

Results The results of fine-grained entity typing are shown in Table 2. From the results, we observe that: (1) Our meta-learning method can outperform all the baselines in all three metrics, which

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.

indicates the capability of our methods in improving the multi-label classification. (2) The SOTA model trained with learned training policy outperforms the one with hierarchy-aware training policy. It is because that our learned training policy can capture not only explicit hierarchical dependencies of labels, but also model other implicit label dependencies. (3) The three different prediction policies can improve the performance of the SOTA classifier. The results are consistent with previous researches that choosing a good prediction policy is an effective way to improve the performance (Fan and Lin, 2007; Lipton et al., 2014). (4) Compared with OntoNotes and BBN datasets, the FIGER show a relatively less improvement when applying these policies. The reason is that the test set of FIGER is not fine-grained enough (e.g., over 38% of entities are only annotated with `/person` and no more fine-grained labels) (Xin et al., 2018).

Algorithm Robustness Previous researches (Morimoto and Doya, 2005; Henderson et al., 2018) show that reinforcement learning-based methods usually lack robustness and are sensitive to the initialization, seeding datasets and pre-trained steps. Thus, we design an experiment to detect whether the trained meta-leaner is sensitive to the initialization. During the test time, instead of using the same initialization in Algorithm 1, we randomly initialize the w_0 and p_0 and learn 10 groups of policies w_T and p_T. For each group of policies, we train a classifier with w_T and evaluate it with p_T using the same metric. The results are shown in Table 3. The results demonstrate that our trained meta-learner is robust to different initialization, which indicates that the meta-learner in our method can generate high-quality and robust training and prediction policies to improve the multi-label classification.

Metrics	Best	Worst	Average	STDEV
Accuracy	0.689	0.679	0.681	0.0043
Macro-F1	0.835	0.821	0.827	0.0039
Micro-F1	0.796	0.787	0.789	0.0036

Table 3: The robustness analysis on the FIGER dataset.

Datasets	Reuters-21578	RCV1-V2
#Labels	90	103
#Average Labels/instance	1.13	3.24
#Training	7,769	781,265
#Testing	3,019	23,149

Table 4: The statistics of text classification datasets.

Weight Analysis To analyze whether our meta-learner can really model the label dependencies, we perform a simple case study. In fine-grained entity classification, we choose 5 labels that are originally organized in a hierarchical structure from the OntoNotes dataset. The corresponding entries of the training policy vector w_T are extracted and expressed with colors. The case study is shown in Figure 5. From the results, we can observe that high-level (parent) labels tend to have less training weights than low-level (child) labels. The results are consistent with our intuition that the labels with more dependencies expose more semantic information during the training.

6 Text Classification

Text classification is a classic problem for NLP, where one needs to categorized documents into pre-defined classes (Nam et al., 2014; Liu et al., 2017; Chen et al., 2017). We choose the datasets

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.

Metrics	Best	Worst	Average	STDEV
Accuracy	0.652	0.641	0.646	0.0028
Macro-F1	0.678	0.654	0.663	0.0041
Micro-F1	0.874	0.855	0.863	0.0033

Table 6: The robustness analysis on the RCV1 dataset.

in which samples have multiple labels and evaluate our model on text classification problem.

Datasets Following the settings in (Nam et al., 2014), we choose two multi-label datasets, Reuters-21578 and RCV1-V2, to test our method. The statistics of two datasets are listed in Table 4.

Reuters-21578: The instances are collected Reuters news articles during the period 1987 to 1991. We use the same training/test split as previous work (Yang, 2001; Nam et al., 2014).

RCV1-V2: RCV1-V2 collects newswire stories from Reuters (Lewis et al., 2004). The training and test dataset originally consist of $23,149$ train and $781,265$ test instances, but we switch them to better training and evaluation (Nam et al., 2014).

Setup Many researches have proved convolutional neural networks (CNN) are effective in extracting information for text classification (LeCun et al., 1998; Kim, 2014; Zhang et al., 2015). Following the (Kim, 2014), we set a CNN model as the classifier C to evaluate our method. Concretely, we use CNN-non-static mentioned in Kim (2014), which means we initialize the word embeddings with pre-trained Word2Vec and update the word embeddings during the training. The standard cross-entropy loss function is implemented, and the thresholds of all the classes are set as 0.5.

We still use strict accuracy, loose macro, and loose micro scores to evaluate the model performance following the settings in (Lewis et al., 2004; Yang and Gopal, 2012; Nam et al., 2014).

Results The results of text classification are shown in Table 5. From the results, we can observe that: (1) Our meta-learning method can outperform all the baselines on two text classification tasks, which indicates that our approach is consistent with different tasks. (2) Compared with RCV1-V2, the classification results on Reuters-21578 show less improvement. The reason is that the number of average labels per instance in Reuters-21578 is 1.13, while the number is 3.24 for RCV1-V2 according to Table 4. That means the multi-label classification on Reuters is close to the multi-class classification. There are little potential label dependencies in Reuters-21578.

Algorithm Robustness Similar to Section 5, we evaluate whether our trained meta-learner is sensitive to the initialization. We follow the same steps mentioned in Section 5, and show the results in Table 6. The results indicate that the robustness of our meta-learner is consistent within different tasks and model structures, which again shows that the trained meta-learner can generate high-quality and robust policies.

7 Conclusion

In this paper, we propose a novel meta-learner to improve the multi-label classification. By modeling the explicit and implicit label dependencies automatically, the meta-learner in our model can learn to generate high-quality training and prediction policies to help both the training and testing process of multi-label classifiers in a principled way. We evaluate our models on two tasks, fine-grained entity typing and text classification. Experimental results show that our method outper-

forms other baselines.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thoughtful comments. This research was supported in part by DARPA Grant D18AP00044 funded under the DARPA YFA program. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

References

Abhishek Abhishek, Ashish Anand, and Amit Awekar. 2017. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 797–807.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3981–3989.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. 1990. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle.

Sheng Chen, Akshay Soni, Aasish Pappu, and Yashar Mehdad. 2017. Doctag2vec: An embedding based multi-label learning approach for document tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 111–120.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67.

Krzysztof Dembczyński, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2012. On label dependence and loss minimization in multi-label classification. *Machine Learning*, 88(1-2):5–45.

Krzysztof Dembszynski, Willem Waegeman, Weiwei Cheng, and Eyke Hüllermeier. 2010. On label dependence in multilabel classification. In *ICML Workshop on Learning from Multi-Label Data*.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Rong-En Fan and Chih-Jen Lin. 2007. A study on threshold selection for multi-label classification. *Department of Computer Science, National Taiwan University*, pages 1–23.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135.

Dan Gillick, Nevena Lazic, Kuzman Ganchev, Jesse Kirchner, and David Huynh. 2014. Context-dependent fine-grained entity type tagging. *arXiv preprint arXiv:1412.1820*.

Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 86(11):2278–2324.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Li Li, Houfeng Wang, Xu Sun, Baobao Chang, Shi Zhao, and Lei Sha. 2015. Multi-label text categorization with joint learning predictions-as-features method. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 835–839.

Xiao Ling and Daniel S Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*.

Zachary C Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. 2014. Optimal thresholding of classifiers to maximize f1 measure. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 225–239.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the*

40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), pages 115–124.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3111–3119.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2018. A simple neural attentive meta-learner. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Jun Morimoto and Kenji Doya. 2005. Robust reinforcement learning. *Neural computation*, 17(2):335–359.

Jinseok Nam, Jungi Kim, Eneldo Loza Mencía, Iryna Gurevych, and Johannes Fürnkranz. 2014. Large-scale multi-label text classificationrevisiting neural networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, pages 437–452.

Jinseok Nam, Eneldo Loza Mencía, Hyunwoo J Kim, and Johannes Fürnkranz. 2017. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 5413–5423.

Xiang Ren, Wenqi He, Meng Qu, Lifu Huang, Heng Ji, and Jiawei Han. 2016. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1369–1378.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. A hierarchical model of reviews for aspect-based sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 999–1005.

Thomas Philip Runarsson and Magnus Thor Jonsson. 2000. Evolution and design of distributed learning rules. In *Proceedings of the 2000 IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 59–63.

Sonse Shimaoka, Pontus Stenetorp, Kentaro Inui, and Sebastian Riedel. 2017. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 1271–1280.

John Shore and Rodney Johnson. 1980. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26(1):26–37.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS)*, pages 4077–4087.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 455–465.

Richard Stuart Sutton. 1984. Temporal credit assignment in reinforcement learning.

Sebastian Thrun and Lorien Pratt. 2012. *Learning to learn*. Springer Science & Business Media.

Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. 2006. A review of multi-label classification methods. In *Proceedings of the 2nd ADBIS Workshop on Data Mining and Knowledge Discovery (ADMKD)*.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS)*, pages 3630–3638.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium (LDC)*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium (LDC)*.

Jiawei Wu, Lei Li, and William Yang Wang. 2018. Reinforced co-training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1252–1262.

Ji Xin, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2018. Improving neural fine-grained entity typing with knowledge attention. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Yiming Yang. 2001. A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR)*, pages 137–145.

Yiming Yang and Siddharth Gopal. 2012. Multilabel classification with meta-level features in a learning-to-rank framework. *Machine Learning*, 88(1-2):47–68.

Min-Ling Zhang and Zhi-Hua Zhou. 2007. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048.

Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NeurIPS)*, pages 649–657.

Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. 2012. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320.