Task-adaptive Label Dependency Transfer for Few-shot Named Entity Recognition

Shan Zhang, Bin Cao,[*] Tianming Zhang, Yuqi Liu and Jing Fan
Zhejiang University of Technology, Hangzhou, China.
{zhangshan,bincao,tmzhang,liuyuqi,fanjing}@zjut.edu.cn

Abstract

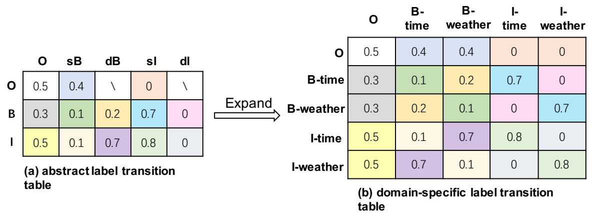
Named Entity Recognition (NER), as a crucial subtask in natural language processing (NLP), suffers from limited labeled samples (a.k.a. few-shot). Meta-learning methods are widely used for few-shot NER, but these existing methods overlook the importance of label dependency for NER, resulting in suboptimal performance. However, applying meta-learning methods to label dependency learning faces a special challenge, that is, due to the discrepancy of label sets in different domains, the label dependencies can not be transferred across domains. In this paper, we propose the **T**ask-adaptive **L**abel **D**ependency **T**ransfer (TLDT) method to make label dependency transferable and effectively adapt to new tasks by a few samples. TLDT improves the existing optimization-based meta-learning methods by learning general initialization and individual parameter update rule for label dependency. Extensive experiments show that TLDT achieves significant improvement over the state-of-the-art methods.

1 Introduction

Named Entity Recognition (NER) aims to locate spans of text and classify them into the pre-defined entity categories such as person, location and organization(Mikheev et al., 1999). In practice, NER is troubled with annotated data scarcity, known as few-shot NER. In recent years, meta-learning is usually used for few-shot NER(Li et al., 2020; de Lichy et al., 2021). Meta-learning first learns prior knowledge from rich-source domains, and then the prior knowledge is used to guide NER models to adapt to an unseen target domain by a few samples. It is worth noting that, as a sequence labeling task(McCallum and Li, 2003), few-shot NER also benefits from considering the dependencies between labels(Ma and Hovy, 2016; Akbik et al., 2018; Liu et al., 2019).

However, meta-learning for label dependency faces a special challenge. There is a label discrepancy between NER tasks in different domains, that is, both the number of labels and the label name semantics are different. The label discrepancy blocks the transfer of label dependencies from source domains to a new domain. For example, the label dependency of “B-company”-“B-project” (financial domain) can not be used for “B-drugs”-“B-etiology” (medical domain).

To remedy this, some studies(Hou et al., 2020; Yang and Katiyar, 2020; Hou et al., 2019; Zhu et al., 2020) use Linear Conditional Random Field (CRF)(Qi and Chen, 2010) framework to model the label dependency as transition probabilities from abstract labels “B”, “I” and “O” to “sB”, “dB”, “sI”, “dI” and “O”, where “B”-“sB” means their domain-specific label semantics are the same (like “B-weather”-“B-weather”) and “B”-“dB” means they are different (like “B-weather”-“B-time”). The abstract label transition table is shown in Figure 1 (a). For example, value 0.1 in green denotes the transition probability from “B” to “sB”. To adapt to a new domain, the abstract label transition is expanded into the domain-specific transition. Figure 1 shows the expansion process, where positions in the same color are filled by the same values. For example, the value of “B-time”-“B-weather” is filled with that of “B” -“dB”, i.e., 0.2 in yellow.



	O	sB	dB	sI	dI
O	0.5	0.4	\	0	\
B	0.3	0.1	0.2	0.7	0
I	0.5	0.1	0.7	0.8	0

(a) abstract label transition table

	O	B-time	B-weather	I-time	I-weather
O	0.5	0.4	0.4	0	0
B-time	0.3	0.1	0.2	0.7	0
B-weather	0.3	0.2	0.1	0	0.7
I-time	0.5	0.1	0.7	0.8	0
I-weather	0.5	0.7	0.1	0	0.8

(b) domain-specific label transition table

Figure 1: An illustration for abstract label transition expansion in Hou et al. (2020).

The domain-specific transition is directly expanded by the abstract label transition without further updating in the specific task, *such domain-*

[*]Corresponding author

specific transition lacks task adaptability and thus leads to inconsistency with reality. For example, according to the result of Figure 1 (b), the transition probability of "B-weather"-"B-time" is the same as that of "B-time"-"B-weather". It doesn't match reality. As shown in Figure 2, people are used to saying "It's raining today" instead of "It's today raining.", which means case 1 is more common than case 2. Therefore, the transition probability of "B-weather"-"B-time" should be higher than that of "B-time"-"B-weather".

label	O	O	B-weather	B-time	
case1	It	is	raining	today	✓
case2	It	is	today	raining	✗

Figure 2: An example of different expanded cases of label "B" to label "dB"

In this paper, to make the domain-specific transition adapt to the corresponding NER task, we introduce the idea of Model-agnostic meta-learning (MAML)(Finn et al., 2017). MAML aims to learn the general initialization which can quickly adapt to specific tasks with few update steps. Based on the idea, we regard the abstract label transition as the general initialization parameters across domains and use the learning mechanism of MAML to learn that. Then the domain-specific transition expanded by the general initial label transition can quickly adapt to the corresponding task with few update steps.

In addition, learning the task-adaptive label transition faces the other challenge. *The hyperparameters (learning rate and weight decay coefficient) for the domain-specific transition update in the specific task are fixed and uniform, which still limits adaption to different tasks*(Baik et al., 2020). The methods to generate the variable hyperparameters for different parameters are widely proposed. Unfortunately, the existing methods require consistent parameters for all tasks, that is, the meaning and the number of transition probabilities for each specific task is uniform. Therefore, these methods cannot be directly used for the task-adaptive label transition learning because of the label discrepancy.

To overcome this problem, we model hyperparameters for individual transition probability instead of all probabilities. Specifically, we first construct sequence features of one individual transition probability in the iterative update process. Then we use the features as the input and propose a Long Shot Term Memory (LSTM) based model to capture the sequence information to learn the task-adaptive update rule. The rule helps the model to generate two variable hyperparameters for the individual transition probability. This approach further enhances task adaptability by assigning variable update hyperparameters to different transition probabilities learning in the specific task.

In summary, we propose a **T**ask-adaptive **L**abel **D**ependency **T**ransfer (TLDT) method which can be widely used under the CRF framework for label dependency in the few-shot NER. Our contributions are summarized as follows:

1. We introduce the idea of MAML to make the transferable label dependency quickly adapt to new tasks by a few gradient updates.
2. We propose the update rule learning that generates task-adaptive hyperparameters for the domain-specific label dependency to effectively adapt to the corresponding NER task.
3. Extensive experimental evaluation demonstrates the outstanding performance of TLDT for few-shot NER, where TLDT outperforms the state-of-the-art methods by 6.47 to 14.62 F1 scores in the 5-shot setting.

2 Background

2.1 Problem Definition

In few-shot learning, NER models aim to output the label sequence $y = (y_1, y_2, ..., y_L)$ of a given sentence $x = (x_1, x_2, ..., x_L)$ by just a few training samples. A domain as a task T_i is a set of (x,y) pairs. NER models are first trained on source domains $T = \{T_1, T_2, ..., T_m\}$ to learn prior knowledge. Then NER models are tested on unseen target domains $T' = \{T'_1, T'_2, ..., T'_n\}$ guided by the prior knowledge, where the T'_i only contains a few labeled examples S called support set.

Formally, the task of K-shot (K samples for each label) NER is defined as follows: given a query sentence x and a K-shot support set S, find the best label sequence y^* of x:

$$y^* = \arg\max_y P(y|x, S) \quad (1)$$

2.2 CRF Framework for NER

Linear Conditional Random Field (CRF) is widely applied in NER, and it outputs the most likely label sequence y for a given sentence x, by considering the connections of entities-labels and labels-labels respectively. The model for the linear CRF defines the probability $p(y|x)$ over all possible label sequences y, given x, by:

$$P(y|x) = \frac{exp(F(y,x))}{\sum_{y' \in Y} exp(F(y',x))} \quad (2)$$

where $F(y, x)$ calculates the score of assigning y to x and is defined as

$$F(y,x) = \sum_{l=1}^{L} (f_T(y_{l-1}, y_l) + f_E(y_l, x)) \quad (3)$$

which counts scores about assigning y_l to x_l at all positions in the sequence. The score at position l concludes labels-labels score $f_T(y_{l-1}, y_l)$ and entities-labels score $f_E(y_l, x)$, known as transition score and emission score in CRF. In our approach, we focus on improving the transition score for few-shot NER. Therefore, we do not impose restrictions on the calculation methods of emission score.

Transition score captures the dependency between labels, which is the probability $P(y_l|y_{l-1})$ of label y_{l-1} to label y_l. CRF constructs a $N \times N$ transition matrix to record label-labels transition scores, where N is the number of labels. As shown in Figure 1 (b), the row of the matrix represents the label of the previous word, and the column represents the label of the current word. The values in the matrix make up the transition score.

In the CRF model training phase, given a training set $\mathcal{D}$, the loss function L is minimized on the set, where L is given by

$$L = \sum_{(x,y) \in \mathcal{D}} -logP(y|x) \quad (4)$$

Based on the trained model, the Viterbi algorithm (Forney, 1973) is usually used to find the best label sequence for each input.

2.3 Meta-Learning

Meta-learning has been wildly used as one strategy to improve the generalization performance of few-shot NER. The idea of meta-learning is "learn to learn". Specifically, the goal of meta-learning is to "learn" prior knowledge such that models can achieve the best generalization performance when they "learn" new tasks.

To this end, the learning process is divided into two levels: an inner- and an outer-level. We define the inner-level parameter as $\boldsymbol{\theta}$ and the meta-parameter at the outer-level is denoted as $\boldsymbol{\Phi}$. At the inner level, support set S is sampled for T_i to make a model adapt to T_i guided by meta-parameter $\boldsymbol{\Phi}$, where S is set to include N labels and each label has K examples. Formally, the model learns optimal parameter $\boldsymbol{\theta}$ for task T_i by minimizing the loss function in Equation (4) on S which is denoted as $\mathcal{L}^S_{T_i}$:

$$\boldsymbol{\theta}^*_{T_i} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}^S_{T_i}(\boldsymbol{\theta}, \boldsymbol{\Phi}) \quad (5)$$

At outer-level, a new set of samples called query set Q from T_i is used to evaluate the generalization ability of the adapted model, which corresponds to loss function $\mathcal{L}^Q_{T_i}$ (Equation (4) on Q). The feedback from $\mathcal{L}^Q_{T_i}$ is then used to adjust the meta-parameter $\boldsymbol{\Phi}$ to make the model achieve higher generalization performance. Finally, the optimal meta-parameter $\boldsymbol{\Phi}^*$ is determined across all tasks.

In detail, the meta-learning objective is to minimize the loss function of the model predictions on the query sets of all tasks:

$$\boldsymbol{\Phi}^* = \arg\min_{\boldsymbol{\Phi}} \sum_{T} \mathcal{L}^Q_{T_i}(\boldsymbol{\theta}^*(\boldsymbol{\Phi})) \quad (6)$$

3 Task-adaptive Label Dependency Transfer

We aim to apply the meta-learning idea to learn the general knowledge of transition scores under the CRF framework, which makes the transition scores effectively adapt to new tasks by a few samples. Since the transition scores learned by the existing methods lack the task adaptability, we introduce the general initialization learning mechanism based on the idea of MAML, to learn the general initial transition scores that can quickly adapt to the specific tasks by few update steps.

In addition, the fixed and uniform update hyperparameters (learning rate and weight decay coefficient) for the transition scores update in the specific task also result in the limitation of task adaptability. However, due to the label discrepancy, the existing meta-learning methods to generate the variable hyperparameters cannot be used for the transition scores update. To remedy this, we propose the update rule learning mechanism to assign the variable hyperparameters for different transition scores.

Based on the above two mechanisms, we propose TLDT and the framework of TLDT is shown in Figure 3. At outer-level, TLDT learns the general initial transition scores shared between tasks to make a model fast adapt to new tasks. In addition, TLDT learns an update rule to generate adjustable hyperparameters for transition scores update at inner-level. At inner-level, based on the initialization and adjustable hyperparameters, the transition scores effectively adapt to a new task by a few samples. The specific details of the approach are as follows.

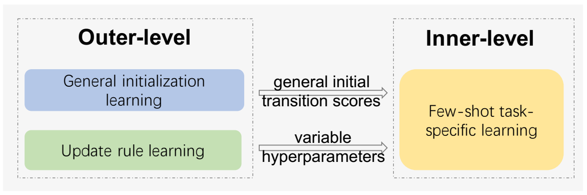

Outer-level
General initialization learning
Update rule learning
general initial transition scores
variable hyperparameters
Inner-level
Few-shot task-specific learning

Figure 3: The framework of proposed TLDT

3.1 General Initialization Learning

Learning initial values of transition scores makes training begin from a beneficial starting point that not only reduces the risk of overfitting but also allows the model to adapt to new tasks rapidly(Ravi and Larochelle, 2017). However, the general initial transition scores learned by the existing methods do not follow the entire meta-learning processing and thus lack task adaptability. To solve this problem, TLDT adopts the CDT mechanism(Hou et al., 2020) to construct the abstract transition scores as the general initialization. Then TLDT follows the idea of MAML to learn the general initialization that can further adapt to a specific task with only a few gradient updates.

Let transition score $\boldsymbol{\theta}_i \in R^{N^2}$ be a vector flattened by an $N \times N$ dimensional transition matrix. At the inner-level, the initial value of $\boldsymbol{\theta}_i$ of a specific task T_i is given by the expanded abstract transition score $\boldsymbol{\gamma}$, which is defined as $\boldsymbol{\theta}_i = \mathcal{E}(\boldsymbol{\gamma})$ and $\mathcal{E}$ is the mapping function of CDT mechanism. Then, as shown in Equation (5), $\boldsymbol{\theta}_i$ is further updated in the new task T_i according to the loss $\mathcal{L}^S_{T_i}$.

At outer-level, the shared initial parameter $\boldsymbol{\gamma}$ is meta-parameter and updated on query sets of all task T according to Equation (6). The complete training procedure is displayed in Algorithm 1 shown in Appendix A.1.

3.2 Task-adaptive Update Rule Learning

Although general initialization learning mechanism can improve the adaptability of transition scores, using a fixed learning rate for task-specific transition scores update still limits adaptation to different tasks(Baik et al., 2020). In this section, we propose an update rule learning mechanism for task-adaptive transition scores optimization. Following the works(Ravi and Larochelle, 2017; Li et al., 2017; Baik et al., 2020), the update rule focus on producing two variable hyperparameters for different transition scores in the specific task, namely, learning rate α and weight decay coefficient β. Different from conventional meta-learning methods, we build the update rule model for each individual transition score of transition scores in the specific tasks, to avoid the problem caused by label discrepancy. Specifically, we construct sequence features based on the optimization process. The features of one individual transition score are the input. Given the input, we propose an LSTM-based model to generate two appropriate hyperparameters for the individual transition score. Next, we detail our method which is divided into three parts: feature construction, model framework and learning mechanism.

Feature Construction We define the learning state as $\tau = [\theta^p, \nabla_{\theta^p}\mathcal{L}^S_{Ti}]$ to construct features for the individual parameter, namely, the p.th element of $\boldsymbol{\theta}$. More importantly, we consider multi-step learning states, where we use a learning rate δ to update the transition score $\boldsymbol{\theta}$ in J steps. Intuitively, compared to only considering the learning state at a single step, multi-step learning states can provide more information to the model. Noting that it allows the model to learn the downward trend from the current point to the convergence point, and further produce more appropriate hyperparameters.

In addition to the learning state τ, we also consider the distance d between the parameter θ^p at the current step j and the previous step $j-1$ to provide guidance for model learning, which is denoted as $d^p_j = \theta^p_j - \theta^p_{j-1}$. Intuitively, a large distance means that the current point is far from the optimal point, and the learning rate should be large.

In conclusion, the $[\theta^p_j, \nabla_{\theta^p_j}\mathcal{L}^S_{T_i}, d^p_j]$ is a 3-dimensional feature vector v_j at step j. Finally, we construct features for $\boldsymbol{\theta}_i$ denoted as a $J \times N^2 \times 3$ tensor F, where $F_{:p:} \in R^{J\times 3}$ represents J-step feature for θ^p and consists of $(v_1, ..., v_J)$, and ":" denotes all elements within the given index.

Model Framework The parameter updated iteratively means that the parameter update at step j is inflected by the previous update process (i.e. based on the result of step $j-1$). Consequently, we use LSTM to capture the sequence dependencies of multi-step updates. As shown in Figure 4, For θ^p, the feature sequence $(v_1, ..., v_J)$ is the input to an LSTM model and hidden states of J steps are concatenated as a $\textit{hidden size} \cdot J$ vector. Then the vector is fed to a linear layer and the learning

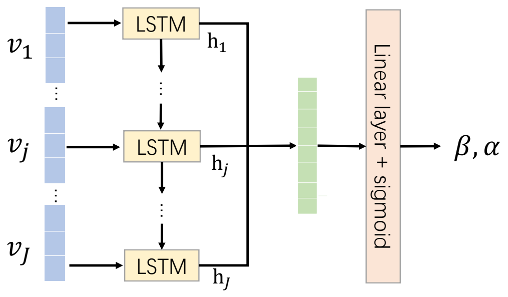


Figure 4: LSTM as meta-model

rate α^p and weight decay coefficient β^p for θ^p are obtained after a sigmoid activation function.

Learning Mechanism The update rule learning process of TLDT is divided into two levels: inner- and outer-level. At inner-level, the LSTM model g_ψ with parameter ψ generates adjustable hyperparameters $\boldsymbol{\alpha_i}$ and $\boldsymbol{\beta_i}$, given the feature tensor $\boldsymbol{F_i}$ of $\boldsymbol{\theta_i}$:

$$(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) = g_\psi(\boldsymbol{F_i}) \tag{7}$$

Then $\boldsymbol{\alpha_i}$ and $\boldsymbol{\beta_i}$ are used for updating parameter $\boldsymbol{\theta}_i$ of the specific task T_i:

$$\boldsymbol{\theta}'_i = \boldsymbol{\beta}_i \odot \boldsymbol{\theta}_i - \boldsymbol{\alpha}_i \odot \nabla_{\boldsymbol{\theta}_i} \mathcal{L}^S_{T_i}(\boldsymbol{\theta}_i) \tag{8}$$

where $\odot$ denotes Hadamard (element-wise) product.

At outer-level, the parameter ψ is updated as meta-parameter on query sets of all task T:

$$\psi = \psi - \eta \nabla_\psi \sum_{T_i \in T} \mathcal{L}^S_{T_i}(\boldsymbol{\theta}_i) \tag{9}$$

where learning rate η is used for updating meta-parameter.

The overall procedure of update rule learning is summarized in Algorithm 2 shown in Appendix A.2. At the inner-level (lines 3 to 13), the algorithm calculates the hyperparameters $(\boldsymbol{\alpha_i}, \boldsymbol{\beta_i})$ and update the transition score $\boldsymbol{\theta_i}$ to adapt to a specific task. In detail, the model g_ψ first constructs J steps features for parameter $\boldsymbol{\theta_i}$ (lines 6 to 10). For each step, the current parameter $\boldsymbol{\theta_{i,j}}$, the corresponding gradient $\nabla_{\boldsymbol{\theta}_{i,j}} \mathcal{L}^S_{Ti}(\boldsymbol{\theta}_{i,j})$ and the distance $\boldsymbol{d}_{i,j}$ are concatenated as a 3-dimension feature vector (lines 7 to 10). Noting that $\boldsymbol{d}_{i,j}$ is set 0 at the last step. The J steps feature vectors are concatenated into feature tensor $\boldsymbol{F}_i$. Then, $\boldsymbol{F_i}$ is fed to the model to get hyperparameters $\boldsymbol{\alpha_i}$ and $\boldsymbol{\beta_i}$ (line 11). The hyperparameters are used to update $\boldsymbol{\theta_i}$ based on gradient optimization (lines 12 to 13). At the outer-level, the model parameter g_ψ is as the meta-parameter and updated on all query sets (line 14).

4 Experiment

This section experimentally evaluates the performance of TLDT on multiple few-shot NER datasets. First, we introduce the experiment setting. Second, we compare TLDT against existing methods in few-shot NER scenarios to demonstrate the effectiveness of TLDT. Then we design ablation experiments to demonstrate the necessity both of each component of TLDT and the features constructed for update rule learning. Finally, We give the effectiveness analysis of TLDT that can improve label dependency transfer on few-shot NER.

4.1 Settings

Dataset We use 4 different datasets as the benchmark dataset following previous studies: 1) CoNLL-2003(Sang and De Meulder, 2003); 2) GUM(Zeldes, 2017); 3) WNUT-2017(Derczynski et al., 2017) and 4) Ontonotes(Pradhan et al., 2013). To simulate the few-shot situation, we adapt the method proposed by Hou et al. (2020) to construct the few-shot datasets from original datasets for 1-shot/5-shots NER. It is worth noting that, due to device limitations and time costs, we randomly sample a part of each original dataset. Therefore, the number of samples is smaller than that in the work(Hou et al., 2020). The detailed statistics of these datasets are shown in Appendix A.3.

Evaluation We conduct main experiments on 1-shot and 5-shot setting as previous work (Hou et al., 2020; Wang et al., 2021). We use four-fold cross-validation to test the robustness of our method on the four benchmark datasets. For each fold, we use one dataset as the test set, one as the validation set, and the remaining two datasets as the training set. We randomly generate four-fold experimental data that the same dataset will not be repeated as a test set or a validation set. F1-score is the evaluation metric and the results are averaged over 5 runs with different random seeds $\pm$ one standard deviation.

Comparison Methods We divide comparison methods into two groups based on whether the label dependency is considered or not. Four methods without label dependency include (1) **Matching Network** (**MNet**)(Vinyals et al., 2016) classifies each entity according to its similarity with the samples of each class; (2) **WarmProtoZero** (**WPZ**)(Fritzler et al., 2019) adopts a similar strategy as MNet, except replacing the matching network with the prototypical network; (3) **TapNet**(Yoon et al., 2019) constructs different mapping

spaces for different tasks. In these mapping spaces, entities corresponding to the same label are close to each other. (4) **LSTM+CRF**(Lample et al., 2016), a traditional sequence labeling method for NER by using LSTM to consider token-label relationship under the CRF framework. (5) **SpanNER**(Ma et al., 2022), a span-level NER method that divides the NER task into two subtasks: span detection and entity typing, which bypass the token-wise label dependency. To consider label dependency, the above three methods including MNet, WPZ, and TapNet are as emission modules. The state-of-the-art method **CDT**(Hou et al., 2020) which constructs an abstract label transition score is used for label dependency transfer in the specific few-shot task.
Hyperparameters We use the uncased BERT-Base (Kenton and Toutanova) to calculate contextual embeddings for all baseline models and our model. We use Adam optimizer (Kingma and Ba, 2015) to train the models with batch size 4 and the learning rate is selected from {1e-4, 1e-5, 1e-6}. For CDT and TLDT which are used for transition score, we set the learning rate η for the meta-parameter update, which is taken from {1e-2, 1e-3, 1e-4}. The learning rate α only for **learning initialization parameters** (**LIP**) update at inner-level is selected from {1e-1, 1e-2, 1e-3}. For feature construction of **learning the update rule** (**LUR**) in TLDT, the update step J and the learning rate δ are taken from {1, 3, 5, 7, 9, 11} and {0.1, 0.2, 0.3, 0.4, 0.5} respectively. We run all experiments on NVIDIA RTX 3090 GPU.

4.2 Main Results

Table 1 shows the 1-shot/5-shots NER results of TLDT and all baselines. Each row represents F1-scores of test domains using the corresponding method.
Result of 1-shot setting For 1-shot, there are three main observations as follows.

(1) Our TLDT performs significantly better than baseline methods without label dependency. TLDT improves at least 21, 5, 6, 7 and 8 F1-score points respectively on average, compared with methods including LSTM+CRF, SpanNER, MNet, WPZ and TapNet. These results illustrate the significance of our work to consider the label dependency for few-shot NER.

(2) TLDT combing with the emission modules (MNet, WPZ and TapNet) achieves better performance than CDT with these modules for considering label dependency. While using the same emission module, TLDT outperforms CDT on all domains and improves more than 10 F1-score points on the OntoNotes dataset. The reason is that CDT transfers the label dependency by using the abstract transition matrix without further update, which results in a lack of task-adaptability of the label dependency in a specific task. This indicates the effectiveness of TLDT to learn task-adaptive transfer for capturing label dependency in few-shot NER.

(3) TLDT maintains excellent prediction performance with all emission modules. Equipped with different emission modules, TLDT always achieves the best performance. Specifically, compared to CDT, TLDT improves 3.37, 2.94 and 4.00 F1-score points respectively on average. This demonstrates that TLDT is a general label dependency transfer method under the CRF framework and has superior improvement on few-shot NER.
Result of 5-shots setting The result of 5-shots NER shows that TLDT also achieves the best performance. Compared to the baselines that ignore or weaken the label dependency, TLDT improves at least 9 F1-score points on average. In the case of considering label dependency, TLDT improves at least 7 F1-score points over CDT on average under the same emission module. The results are consistent with the 1-shot setting, which demonstrates the generalization ability of TLDT in more shots situations.

4.3 Analysis

Ablation Study In this section, we first conduct ablation analysis to indicate the necessity of each component in our method (TLDT). Second, we analyze the effectiveness of three features for the update rule learning of TLDT, namely parameter, gradient, parameter distance in the optimization process.
Necessity of each component To verify that the components including LIP and LUR in TLDT are all essential, we perform the ablation experiments for TLDT with MNet, WPZ and TapNet in 1-shot/5-shots situations and remove each component respectively. Table 2 shows the averaged F1-score of all domains on 1-shot/5-shots. We can see TLDT is more advantageous than LIP and LUR under three emission modules. Specifically, when LUR is removed, the F1-scores decrease both on 1-shot/5-shots. This demonstrates that it's crucial to learn update rule for task-adaptive label dependency trans-

K-shot	Model	Wiki	SocialMedia	OntoNotes	News	Ave.
1-shot	LSTM+CRF	1.26±0.10	1.34±0.23	0.45±0.07	4.08±0.34	1.78±0.35
	SpanNER	5.45±0.14	19.57±1.05	9.33±1.59	39.18±1.36	18.38±0.82
	MNet	2.96±0.07	20.59±0.64	6.42±0.82	39.44±0.95	17.35±0.32
	MNet+CDT	3.45±0.38	24.55±2.09	9.38±2.29	42.87±1.29	20.06±1.36
	WPZ	2.91±0.24	20.50±0.82	6.23±0.43	34.06±1.05	15.93±0.26
	WPZ+CDT	4.11±0.64	24.01±1.47	8.78±1.75	42.11±1.90	19.76±1.34
	TapNet	3.30±0.47	19.00±0.79	8.32±0.54	28.66±3.46	14.82±1.06
	TapNet+CDT	3.40±0.44	22.50±2.15	11.51±1.66	41.60±1.11	19.75±1.02
	MNet+TLDT (ours)	3.95±0.39	25.56±1.93	**22.07±1.06**	42.16±2.66	23.43±1.30
	WPZ+TLDT (ours)	**5.74±0.74**	27.67±1.69	18.91±2.60	38.47±7.98	22.70±2.84
	TapNet+TLDT (ours)	**5.74±0.79**	**27.94±1.42**	18.27±1.91	**43.05±1.70**	**23.75±1.04**
5-shot	LSTM+CRF	4.35±0.09	4.79±0.22	4.81±1.07	11.06±0.31	6.25±0.25
	SpanNER	4.74±0.78	16.38±7.30	9.66±0.31	39.59±2.24	17.59±1.88
	MNet	3.80±0.82	18.45±2.78	9.58±1.23	40.70±4.86	18.13±1.61
	MNet+CDT	6.17±0.63	20.58±3.70	14.62±1.93	48.87±4.19	22.56±1.14
	WPZ	2.77±0.34	17.03±3.28	9.41±0.82	28.84±3.91	14.51±0.92
	WPZ+CDT	2.20±1.17	19.69±2.96	17.45±1.92	37.65±3.84	19.25±1.84
	TapNet	3.04±0.56	14.37±1.69	10.85±0.96	31.69±2.40	14.99±0.26
	TapNet+CDT	5.34±1.60	22.12±2.46	15.66±2.66	40.46±4.45	20.89±0.88
	MNet+TLDT (ours)	8.26±1.65	27.93±1.74	**31.03±3.07**	**49.31±4.99**	**29.13±1.90**
	WPZ+TLDT (ours)	**10.14±1.60**	24.42±3.13	30.65±1.92	44.02±1.65	27.30±1.26
	TapNet+TLDT (ours)	6.66±0.88	**29.08±2.70**	30.41±3.31	45.95±1.83	28.02±0.92

Table 1: F1-scores with standard deviations on the four benchmark datasets

Model	1-shot	5-shot
MNet+LIP	22.08±1.45	26.78±1.96
MNet+LUR	18.29±1.71	24.11±2.08
MNet+TLDT	**23.43±1.30**	**29.13±1.90**
WPZ+LIP	21.86±1.24	21.60±2.10
WPZ+LUR	20.27±3.29	23.40±1.52
WPZ+TLDT	**22.70±2.84**	**27.30±1.26**
TapNet+LIP	21.75±0.74	21.82±0.54
TapNet+LUR	20.47±0.70	23.95±2.70
TapNet+TLDT	**23.75±1.04**	**28.02±0.92**

Table 2: Ablation test over different components of TLDT on few-shot NER. Results are averaged F1-score of all domains.

fer. Similarly, if LIP is removed, the F1-scores also decrease both on 1-shot/5-shots. This shows that it's important to learn initialization parameters for label dependency transfer in few-shot NER.

Effectiveness of three features We aim to prove the effectiveness of three features used for LUR, we combine these features in pairs to illustrate the significance of the remaining feature and the four domains in 5-shots situation are used for evaluation. Table 3 shows the results of the combination in pairs and considering all features. In the table, we can see that the combination including all features achieves the best performance on average. In other words, the absence of any one of the features makes the F1-score decrease, which demonstrates the three features are all necessary for TLDT.

Features	Ave.
parameter+gradient	18.76
parameter+distance	20.47
gradient+distance	24.59
parameter+gradient+distance	**29.32**

Table 3: F1-scores of different features on 5-shots name entity recognition

Inside study of TLDT In this section, we first indicate why LIP and LUR work for label dependency transfer in few-shot NER. Then we further validate the generalization ability of TLDT, that is, the predictive performance in test domains by the model trained on a few samples.

We conduct the following experiments. To verify the effectiveness of LIP, which learns initialization by updating in the specific task, we design a comparison method, called CDT-adapt, that CDT further adapt (CDT-adapt) to the specific task by updating the expanded matrix. To demonstrate the advantages of LUR, we compare to a method that randomly initializes parameters and uses fixed learning rate and weight decay coefficient (FixedL&W) to update the parameters in the specific task. Since LIP and LUR use a one-step update, CDT-adapt and FixedL&W also adopt one-step for consistency. To illustrate the better generalizability of TLDT compared to task-adaptive CDT and FixedL&W, we select the optimal update step from {1, 3, 5,

Model	Ave.
CDT-adapt (1 step)	19.79
LIP (ours)	21.82
FixedL&W (1 step)	15.62
LUR (ours)	23.95
CDT-adapt (optimal step)	24.43
FixedL&W (optimal step)	21.75
TLDT (ours)	**28.57**

Table 4: Comparison of generalization between CDT, FixedL&W and TLDT on 5-shots NER

7, 9} for CDT-adapt and FixedL&W and compare TLDT to them. All experiments are conducted over the four domains in 5-shot setting and the learning rate is set as 0.1.

The results of the averaged F1-scores are shown in Table 4. First, with the same learning rate and 1 step update, LIP improves CDT-adapt (1 step) 2.03 F1-score points, which indicates the advantage of task-adaptive initialization learned by LIP is obvious. Second, LUR outperforms FixedL&W obviously at least 8 F1-score points. This demonstrates the necessary of learning variable learning rate and weight decay items for task-adaptive label dependency transfer. Third, F1-scores of CDT-adapt (optimal step) and FixedL&W (optimal step) are both lower than TLDT by at least 4 and 6 points respectively on average F1-score. It shows that whether starting from the abstract matrix learned by CDT or random initialization, or applying multi-step updates with fixed learning rules, the generalization performance of these learning mechanisms is always inferior to that of TLDT with only one-step update. This suggests that their learning methods may be more prone to trapping the transition score in a local optimum or suffering the overfitting when learning on a few samples. On the contrary, TLDT has excellent generalization performance in few-shot situations by task-adaptive initialization learning and variable update rules.

Outside study of TLDT In this section, we design an analytical experiment to further illustrate how our method works. In the experiment, we compare the transition matrices learned by TLDT and CDT with the ground truth, to illustrate whether the probabilities learned by TLDT more closely match the actual data. Specifically, we count the occurrence frequency of label-label pairs in the test set as the ground truth and extract the two transition matrices of TLDT and CDT. Then we map the frequency and the values of the two matrices to interval [0, 1] respectively, and use heatmaps to represent them. As shown in Figure 5, the color band corresponding to [0, 1] is divided into five intervals, where the darker color indicates the higher label-label transition probability. For ease of illustration, we use "ABCDE" to denote these intervals.

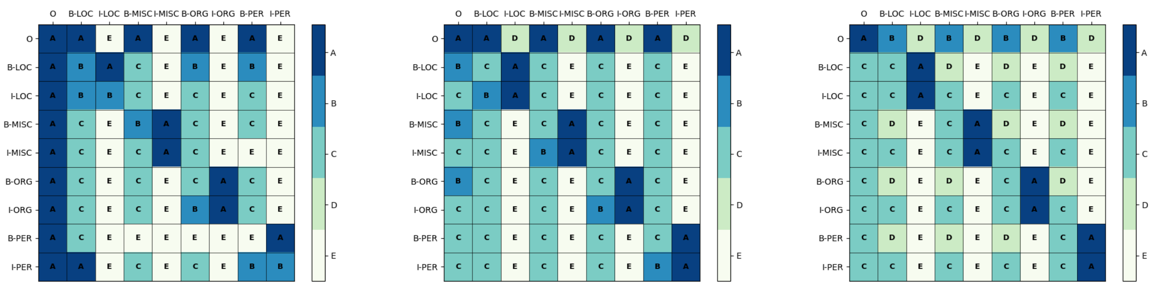
From Figure 5, we can draw three observations. (1) The matrix learned by TLDT matches more ground truth than that learned by CDT, where the number of matches is 57 and 43 respectively. It shows that the transition matrix in TLDT is more realistic than that in CDT. (2) In unmatched transition scores, the scores learned by TLDT is closer to the ground truth than that learned by CDT. For example, label pairs "B-LOC"-"O", "B-MISC"-"O" and "B-ORG"-"O" are "A" in real data. The transition scores of these label pairs learned by TLDT and CDT are "B" and "C" respectively. This demonstrates that TLDT indeed learn transition scores that more fit real data. (3) For label pairs belong to "B"-"dB", the transition scores learned by TLDT are "C" which are consistent with actual data, but that learned by CDT are all "D". The reason is that the transition matrix learned by CDT is directly expanded by the abstract transition, and the abstract transition is learned from the train set where "B"-"dB" may appear with a very low rating. Therefore, the probabilities of label pairs belong to "B"-"dB" in expanded matrix are also low. On the contrary, TLDT has learned a more realistic transition matrix by further updating that in test task by the few samples.

5 Related work

Few-shot Named Entity Recognition

Conventional few-shot NER methods focus on modeling the correlation of entities-labels. These methods(Mettes et al., 2019; Tong et al., 2021; Li et al., 2021) aim to learn a prototype for each label and classify an entity by finding the nearest prototype in a mapping space. But as a sequence labeling task, label dependency also need to be taken into account in few-shot NER. Some works add the linear CRF layer as the last layer of their models to learn label dependency(Gong et al., 2021; Ma and Hovy, 2016; Li et al., 2020). However, the learned label dependency cannot be transferred to new tasks due to the label discrepancy.

To solve this problem, the CDT mechanism is proposed to learn abstract label dependencies (Hou et al., 2020) and is widely adopted in many works(Yang and Katiyar, 2020; Hou et al., 2019;



(a) Occurrence frequency of label pairs (b) Transition matrix learned by TLDT (c) Transition matrix learned by CDT

Figure 5: Comparison of the ground truth with transition matrices learned by TLDT and CDT

Zhu et al., 2020). However, the task-specific label dependencies obtained by the CDT mechanism lack task adaptability. In this paper, we propose TLDT to make label dependencies transferable and effectively adapt to new tasks in few-shot NER.

Optimization-based Meta-Learning

The studies on optimization-based meta-learning can be divided into the following two categories: *Initial parameter optimization.* A lot of work focus on learning weight initialization such that models can fast adapt to new tasks. MAML(Finn et al., 2017) learned the initialization that makes a model adapt to new tasks by a few gradient updates. Some work has been proposed to improve MAML(Rajeswaran et al., 2019; Nichol et al., 2018; Grant et al., 2018). However, the existing methods can not be directly applied to our task due to the label discrepancies. In this paper, we improve MAML by using the abstract initialization. *Hyperparameters learning.* The existing studies learn adjustable hyperparameters for inner-loop parameter optimization. Ravi and Larochelle (2017) used an LSTM meta-learner trained to be an optimization algorithm. Li et al. (2017) proposed Meta-SGD to learn a learning rate vector for all tasks. ALFA is proposed by Baik et al. (2020) to specify the form of parameter update rule to include the learning rate and weight decay terms. These methods require consistent parameters for all tasks, but in our task, the discrepancy of label sets causes inconsistency. TLDT models the individual parameter and captures the sequence information to generate the task-adaptive hyperparameters.

6 Conclusion

In this paper, we propose an optimization-based meta-learning approach to transfer label dependency for few-shot NER. To make label dependency transferable and effectively adapt to new tasks, we propose a general initialization learning mechanism that learns shared initial label dependency and can quickly adapt to new tasks with a few gradient updates. In addition, we also propose an update rule to generate task-adaptive hyperparameters for label dependency optimization in a specific task. Experiment results validate that both the general initialization learning and update rule learning can improve the few-shot NER accuracy.

7 Limitations

Regarding our work, we summarize the following three limitations.

(1) TLDT is proposed under the assumption that entities in a sentence are independent of each other, that is, they do not overlap. In the case of overlapping entities, TLDT cannot capture the label dependency of these entities.

(2) We tested TLDT on four public datasets and these datasets contain a part of the same labels. This may help TLDT to learn the general knowledge of the label dependency. We did not give experimental proof that if the label sets of the datasets all totally different, whether the TLDT can maintain good robustness.

(3) To overcome the label discrepancy problem and generate adjustable hyperparameters, we model for the individual transition score in the specific task. However, the mechanism neglects the correlation between parameters, which limits the ability to update rule learning.

8 Acknowledgement

This research was partially supported by National Natural Science Foundation of China (No. 62276233), Key Research Project of Zhejiang Province (2023C01048) and Natural Science Foundation of Zhejiang Province of China (LQ22F020018).

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. 2020. Meta-learning with adaptive hyperparameters. *Advances in Neural Information Processing Systems*, 33:20755–20765.

Cyprien de Lichy, Hadrien Glaude, and William Campbell. 2021. Meta-learning for few-shot named entity recognition. In *Proceedings of the 1st Workshop on Meta Learning and Its Applications to Natural Language Processing*, pages 44–58.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the wnut2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Alexander Fritzler, Varvara Logacheva, and Maksim Kretov. 2019. Few-shot classification in named entity recognition task. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 993–1000.

Yuan Gong, Lu Mao, and Changliang Li. 2021. Few-shot learning for named entity recognition based on bert and two-level model fusion. *Data Intelligence*, 3(4):568–577.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. 2018. Recasting gradient-based meta-learning as hierarchical bayes. In *International Conference on Learning Representations*.

Yutai Hou, Wanxiang Che, Yongkui Lai, Zhihan Zhou, Yijia Liu, Han Liu, and Ting Liu. 2020. Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1381–1393.

Yutai Hou, Zhihan Zhou, Yijia Liu, Ning Wang, Wanxiang Che, Han Liu, and Ting Liu. 2019. Few-shot sequence labeling with label dependency transfer and pair-wise embedding. *arXiv preprint arXiv:1906.08711*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Universal Language Model Fine-tuning for Text Classification*, page 278.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Gen Li, Varun Jampani, Laura Sevilla-Lara, Deqing Sun, Jonghyun Kim, and Joongkyu Kim. 2021. Adaptive prototype learning and allocation for few-shot segmentation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8330–8339. IEEE.

Jing Li, Billy Chiu, Shanshan Feng, and Hao Wang. 2020. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. 2017. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*.

Yijin Liu, Fandong Meng, Jinchao Zhang, Jinan Xu, Yufeng Chen, and Jie Zhou. 2019. Gcdt: A global context enhanced deep transition architecture for sequence labeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441.

Tingting Ma, Huiqiang Jiang, Qianhui Wu, Tiejun Zhao, and Chin-Yew Lin. 2022. Decomposed meta-learning for few-shot named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1584–1596.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *CoNLL*.

Pascal Mettes, Elise van der Pol, and Cees Snoek. 2019. Hyperspherical prototype networks. *Advances in neural information processing systems*, 32.

Andrei Mikheev, Marc Moens, and Claire Grover. 1999. Named entity recognition without gazetteers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 1–8, Bergen, Norway. Association for Computational Linguistics.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using ontonotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152.

Luole Qi and Li Chen. 2010. A linear-chain crf-based learning approach for web opinion mining. In *International Conference on Web Information Systems Engineering*, pages 128–141. Springer.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32.

Sachin Ravi and H. Larochelle. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Meihan Tong, Shuai Wang, Bin Xu, Yixin Cao, Minghui Liu, Lei Hou, and Juanzi Li. 2021. Learning from miscellaneous other-class words for few-shot named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6236–6247.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29.

Yaqing Wang, Haoda Chu, Chao Zhang, and Jing Gao. 2021. Learning from language description: Low-shot named entity recognition via decomposed framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1618–1630.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning*, pages 7115–7123. PMLR.

Amir Zeldes. 2017. The gum corpus: creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

Su Zhu, Ruisheng Cao, Lu Chen, and Kai Yu. 2020. Vector projection network for few-shot slot tagging in natural language understanding. *arXiv preprint arXiv:2009.09568*.

A Appendix

A.1 Algorithm 1

Algorithm 1 shows the complete training procedure of the general initial transition scores.

Algorithm 1: Initial parameters learning process

```
Input  : Source domains $T$;
         Learning rate $\alpha$ for inner-level;
         Learning rate $\eta$ for outer-level;
Output : Initial parameter $\boldsymbol{\gamma}$
1 $\boldsymbol{\gamma} \longleftarrow$ random initialization;
2 while not converged do
3     for each task $T_i \in T$ do
4         $\boldsymbol{\theta}_i \longleftarrow$ expand $\boldsymbol{\gamma}$ by CDT mechanism;
5         $S, Q \longleftarrow$ random dataset from $T_i$;
6         compute $\nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{T_i}^{S}(\boldsymbol{\theta}_i)$;
7         compute $\boldsymbol{\theta}_i' = \boldsymbol{\theta}_i - \alpha \nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{T_i}^{S}(\boldsymbol{\theta}_i)$
8     update $\boldsymbol{\gamma} = \boldsymbol{\gamma} - \eta \nabla_{\boldsymbol{\gamma}} \sum_{T_i \in T} \mathcal{L}_{T_i}^{Q}(\boldsymbol{\theta}_i')$
9 return $\boldsymbol{\gamma}$
```

A.2 Algorithm 2

Algorithm 2 shows the overall procedure of update rule learning mechanism to generate the variable hyperparameters for different transition scores in the specific task.

Algorithm 2: Update rule learning process

```
Input  : Source domains $T$;
         Learning rate $\eta$ for outer-level;
         Learning rate $\delta$ and update step $k$ for
extracting features;
Output : parameter $\boldsymbol{\psi}$
1  $\boldsymbol{\psi} \longleftarrow$ random initialization;
2  while not converged do
3      for each task $T_i \in T$ do
4          $S, Q \longleftarrow$ random dataset from $T_i$
5          $\boldsymbol{\theta}_i \longleftarrow$ random initialization
6          for update step j: 1 to J do
7              compute $\nabla_{\boldsymbol{\theta}_{i,j}} \mathcal{L}_{Ti}^{S}(\boldsymbol{\theta}_{i,j})$
8              $\boldsymbol{\theta}_{i,j+1} = \boldsymbol{\theta}_{i,j} - \delta \nabla_{\boldsymbol{\theta}_{i,j}} \mathcal{L}_{T_i}^{S}(\boldsymbol{\theta}_{i,j})$
9              $\boldsymbol{d}_{i,j} = \boldsymbol{\theta}_{i,j+1} - \boldsymbol{\theta}_{i,j}$
10             $\boldsymbol{F}_{j::} \longleftarrow$
                 $[\nabla_{\boldsymbol{\theta}_{i,j}} \mathcal{L}_{Ti}^{S}(\boldsymbol{\theta}_{i,j}), \boldsymbol{\theta}_{i,j+1}, \boldsymbol{d}_{i,j}]$
11         Compute $(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) = g_{\boldsymbol{\psi}}(\boldsymbol{F}_i)$
12         Compute $\nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{Ti}^{S}(\boldsymbol{\theta}_i)$
13         Compute
             $\boldsymbol{\theta}_i' = \boldsymbol{\beta_i} \odot \boldsymbol{\theta}_i - \boldsymbol{\alpha_i} \odot \nabla_{\boldsymbol{\theta}_i} \mathcal{L}_{T_i}^{S}(\boldsymbol{\theta}_i)$
14     update $\boldsymbol{\psi} = \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \sum_{T_i \in T} \mathcal{L}_{T_i}^{Q}(\boldsymbol{\theta}_i')$
15 return $\boldsymbol{\psi}$
```

A.3 Statistic of datasets

Tabel 5 shows the detailed statistics of the original dataset used to construct few-shot experiment data.

Table 5: Statistic of Dataset

Dataset	Domain	# Labels	# Sent
CoNLL	News	5	4703
GUM	WiKi	12	5300
WNUT	Social	7	10433
OntoNotes	Mixed	19	3438

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☑ A2. Did you discuss any potential risks of your work?
Section 7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☑ A4. Have you used AI writing assistants when working on this paper?
Grammarly; Perform the grammar correction task for all sections

B ☑ Did you use or create scientific artifacts?

Sections 3 and 4

☑ B1. Did you cite the creators of artifacts you used?
Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 4

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.