Federated Domain Adaptation for Named Entity Recognition via Distilling with Heterogeneous Tag Sets

Rui Wang[1] **Tong Yu**[2†] **Junda Wu**[3] **Handong Zhao**[2] **Sungchul Kim**[2]
Ruiyi Zhang[2] **Subrata Mitra**[2] **Ricardo Henao**[1,4]
[1]Duke University [2]Adobe Research [3]New York University [4]KAUST
{rui.wang16, ricardo.henao}@duke.edu jw6466@nyu.edu
{tyu, hazhao, sukim, ruizhang, sumitra}@adobe.com

Abstract

Federated learning involves collaborative training with private data from multiple platforms, while not violating data privacy. We study the problem of federated domain adaptation for Named Entity Recognition (NER), where we seek to transfer knowledge across different platforms with data of multiple domains. In addition, we consider a practical and challenging scenario, where NER datasets of different platforms of federated learning are annotated with heterogeneous tag sets, *i.e.*, different sets of entity types. The goal is to train a global model with federated learning, such that it can predict with a complete tag set, *i.e.*, with all the occurring entity types for data across all platforms. To cope with the heterogeneous tag sets in a multi-domain setting, we propose a distillation approach along with a mechanism of instance weighting to facilitate knowledge transfer across platforms. Besides, we release two re-annotated clinic NER datasets, for testing the proposed method in the clinic domain. Our method shows superior empirical performance for clinic NER with federated learning.

1 Introduction

Federated learning for Named Entity Recognition (NER) is the task of collaboratively learn with NER datasets from multiple platforms, while not violating data privacy, *i.e.*, without sharing data across different platforms (Ge et al., 2020). A platform can be an institution, *e.g.*, a hospital or a drug company, where a private collection of clinic NER dataset are locally stored. In reality, data from different platforms are usually sampled from different clinic domains, due to different patient groups, *etc*. Additionally, different schemes may be used when annotating for different platforms. This happen when healthcare providers use customized tag sets to create their own datasets (Beryozkin et al., 2019). As an example, a hospital may hold a dataset of clinical reports from doctors annotated with entities of patient *Disease* and *Drugs* prescribed by the doctors, while a drug company may have text data of patient feedback, annotated with *Drugs* and their adverse drug effects (*ADE*). In this case, it would be mutually beneficial for the hospital and the drug company if they can train in a federated manner a shared (global) NER model with both datasets. The global model should in principle predict with the complete tag set, *i.e.*, {*Disease*, *Drugs*, *ADE*}, enabling the hospital to also recognize the *ADE* in their clinic reports and the drug company to identify *Disease* from their patient feedback, without sharing or re-annotating their local datasets. This can be regarded as a problem of domain adaptation, since the key challenge is to efficiently transfer knowledge of locally unlabeled entity types, *i.e.*, *Disease* and *ADE* across domains/platforms, so that the resulting global model can work for both the hospital and the drug company.

So motivated, we study federated domain adaptation for clinic NER in the multi-domain setting where datasets from multiple platforms representing different domains. Further, we address a more challenging scenario in which different platforms also annotate with different tag sets, *i.e.*, set of entity types. The goal is to benefit all platforms from federated learning, via training a global model that predicts with the complete tags set, including all the encountered entity types, for text of different domains/platforms. Note there are previous works studying federated NER in a multi-domain setting (Ge et al., 2020; Zhao et al., 2021). However, these works generally presume that the NER model for one platform only predicts with the entity types annotated in the local training data, unlike our setting that requires predicting on a larger tag set (the complete tag set). Here, we claim that such an assumption might not be practical in a multi-domain setting. To illustrate this, suppose there is a platform with annotations of training data

†Corresponding Author

Findings of the Association for Computational Linguistics: ACL 2023, pages 7449–7463

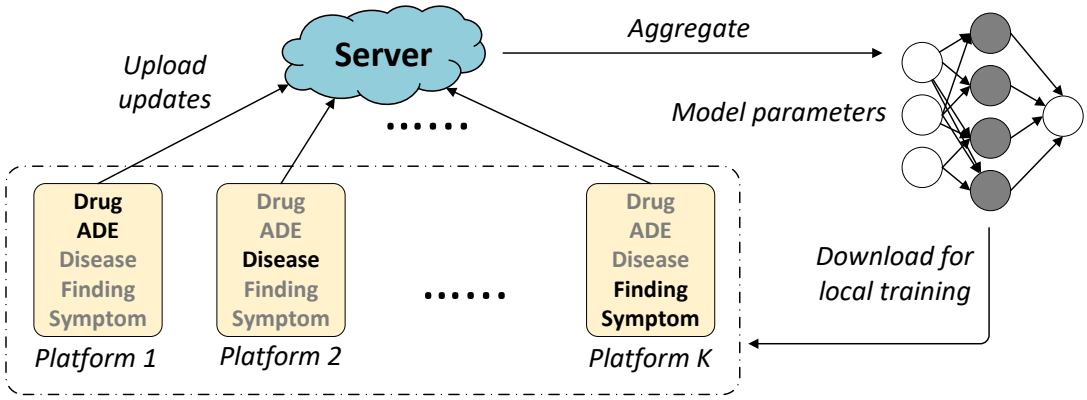


Figure 1: Framework for federated learning for clinical NER with heterogeneous tag sets. Different platforms are annotated with different tag sets, $\{Drug, ADE\}$, $\{Disease\}$, $\{Finding, Symptom\}$, *etc.*, denoted with the unshaded font. The goal is to learn an NER model that predict with all the entity types, while not sharing data across platforms. In addition, we assume the text data of different platforms are from different domains (*e.g.*, platform 1 is from a drug company and platform 2 is from clinical reports).

that sufficiently cover enough entity types for its own propose of evaluation. For this platform, with enough amount of training data locally, joint training with data from other distant domains may harm the performance of the resulting model on its local data, *i.e.*, there is no guarantee that data of other platforms is similar enough to be beneficial to its own domain. As a result, such a platform might be reluctant in joining federated learning, further considering the potential risk of data leakage in any federated learning system (Li et al., 2021). On the contrary, we require to predict with the complete tag set, while annotating with incomplete (subset of the complete tag set) and heterogeneous tag sets locally. This motivates a platform to participate in federated learning, so that it can benefit from knowledge of locally unlabeled entity types transferred from other platforms.

To address the heterogeneous tag sets and facility knowledge transfer across platforms, with regards to the locally unlabeled entity types, we propose a distillation approach, that distills knowledge of unlabeled entity types from other platforms via pseudo-annotations with the complete tag set. Based on the proposed distillation, we further propose a instance weighting mechanism, so that knowledge learned with local data is more transferable across platforms. We adopt a prompt-based NER model (Chen et al., 2022) with superior performance for cross-domain NER, and only transmit prompt-related parameters (7% of the model size) for each round of federated learning to reduce the communication cost. We should note that a comprehensive evaluation of the global model in the setting considered requires testing data with the complete tag set for each domain/platform. However, existing public clinical datasets of different domains are usually annotated using different tag sets (with small overlap), *i.e.*, they lack evaluation data that is consistently annotated with the complete tag sets for multiple domains. Therefore, we re-annotate the ADE-Corpus (Gurulingappa et al., 2012) and SMM4H (Weissenbacher et al., 2019) datasets using the annotation scheme of CADEC (Karimi et al., 2015), resulting in datasets of multiple domains that are annotated consistently for evaluation. Our contributions are as follow:

- We study federated learning for clinic NER, where data in different platforms can be from multiple domains and annotated with heterogeneous tag sets.
- We propose a distillation approach along with a weighting mechanism to facilitate knowledge transfer across different platforms.
- We release two re-annotated clinic datasets for evaluation in clinical settings and to encourage future research. Empirical results show that our method delivers superior performance in the considered setting.

2 Preliminaries

2.1 Problem Formulation

Figure 1 illustrates our considered setting for clinic NER. Suppose there are K platforms in federated learning with datasets $\{\mathcal{D}_k\}_{k=1}^{K}$, $\mathcal{D}_k =$

Algorithm 1 Our Federated Learning Algorithm.

Assume the non-trainable parameters of the NER model are available in each platform.
Randomly initialize the trainable parameters, $\boldsymbol{\theta}_1$, for the NER model on the server of federated learning.
Initialize the instance weights $w_{i,1}^k = 1$, for $i = 1, \ldots, N_k$ and $k = 1, \ldots, K$ (see Section 3.3).
for the t^{th} round of federated learning **do**
  % Update instance weighting
  Compute $w_{i,t+1}^k$ according to (12).
  % Local training
  Download $\boldsymbol{\theta}_t$ to each local platform.
  Train locally on each platform, via distilling with the pseudo-complete annotation set resulting in $\{\boldsymbol{\theta}_t^k\}_{k=1}^K$ (see Section 3.2).
  % Aggregation
  Upload $\{\boldsymbol{\theta}_t^k\}_{k=1}^K$ to the server and aggregate with (1), generating $\boldsymbol{\theta}_{t+1}$ for the next round of federated learning.
end for

$\{\boldsymbol{X}_i, \boldsymbol{Y}_i^{\mathcal{T}_k}\}_{i=1}^{N_k}$, with N_k being the size of $\mathcal{D}_k$. $\boldsymbol{X}_i$ is a text sequence and $\boldsymbol{Y}_i^{\mathcal{T}_k}$ is its NER label sequence, annotated with tag set $\mathcal{T}_k$. In Figure 1, we have $\mathcal{T}_1 = \{Drug, ADE\}$, $\mathcal{T}_2 = \{Disease\}$, *etc.* We assume $\boldsymbol{X}_i$ of different platforms are from different text domains. The goal is to train an NER model that predicts with the complete tag set *i.e.* $\mathcal{T} = \cup_{k=1}^K \mathcal{T}_k$, for all platforms, without data being shared across different platforms.

2.2 Federated Learning

As illustrated in Figure 1, federated learning involves periodical communication between the server and platforms involving the trainable parameters of the model. Specifically, let $\boldsymbol{\theta}_t$ be the trainable parameters of the NER model before the t^{th} communication round of federated learning. We assume the non-trainable parameters, *e.g.*, the pretrained parameters of a PLM, are available locally in each platform. A typical training cycle of federated learning includes:

Local Training: $\boldsymbol{\theta}_t$ is transferred to each platform and is then trained/updated locally with the private data of each platform. Specifically, $\boldsymbol{\theta}_t$ is trained for E_{loc} epochs separately on different platforms. We denote $\{\boldsymbol{\theta}_t^k\}_{k=1}^K$ as the trainable parameters of different platforms from local training.

Aggregation: After local training, each platform will transfer their updated parameters $\{\boldsymbol{\theta}_t^k\}_{k=1}^K$ to the server. Since the goal of our federated learning setting is to training a global model for all platforms, the server will aggregate the $\{\boldsymbol{\theta}_t^k\}_{k=1}^K$, generating $\boldsymbol{\theta}_{t+1}$ for the next round of communication. The aggregation is usually performed via weighted averaging, *i.e.*,

$$\boldsymbol{\theta}_{t+1} = \sum_{k=1}^{K} m_k \boldsymbol{\theta}_t^k, \tag{1}$$

where $\sum_k m_k = 1$. Since aggregation is not the focus of this work, we will discuss the values of m_k in the Appendix. Algorithm 1 shows the complete procedure of federated learning. The proposed distillation and instance weighting mechanism are described in Sections 3.2 and 3.3, respectively.

3 Methodology

3.1 Model Architecture

In order to efficiently train a global model for all the participants, we need to *i)* Facilitate knowledge transfer across different platforms/domains, so that each client can benefit from knowledge regarding locally unlabeled entity types, transferred from other platforms. *ii)* Reduce the communication cost of federated learning. With these considerations, we adopt LightNER (Chen et al., 2022) as our NER model for federated learning. Below, we briefly describe the LightNER model, along with the rationale that we adopt it for our setting.

Sequence-to-Sequence NER: NER is conventionally identified a sequence labeling problem, which predicts with a label-specific output layer (Luo et al., 2020; Lee et al., 2020) on top of a Pretrained Language Model (PLM), *e.g.*, BERT. However, such models may have inferior performance for cross-domain problems, since the label-specific output layer that is trained from scratch cannot benefit from the pretrained knowledge for generalization (Chen et al., 2022). To solve this, recent works (Cui et al., 2021; Chen et al., 2022) adopt a sequence-to-sequence framework for NER based on the pretrained BART model (Lewis et al., 2019), where the entity labels are predicted as natural language tokens in the output sequence, leveraging the pretrained semantics knowledge from BART token embeddings for better generalization. By formulating NER as sequence-to-sequence generation, LightNER achieve superior performance for cross-domain NER tasks, a merit that we value for our setting involving multiple domains. Given a length-L text sequence $\boldsymbol{X}_i = [\boldsymbol{x}_l]_{l=1}^L$ from platform k,

the model should generate the following label sequence $\boldsymbol{Y}_i^{\mathcal{T}^k}$, indicating the start/end positions and entity types of each entity within $\boldsymbol{X}_i$,

$$\boldsymbol{Y}^{\mathcal{T}^k} = [\boldsymbol{p}_1^{\mathcal{T}^k}; \cdots; \boldsymbol{p}_n^{\mathcal{T}^k}],\ \boldsymbol{p}_c^{\mathcal{T}^k} = [\boldsymbol{s}_c, \boldsymbol{e}_c, \boldsymbol{t}_c], \quad (2)$$

where $c = 1, \cdots, n$ and $[;]$ denotes concatenation. n is the number of entities in $\boldsymbol{X}_i$. $\boldsymbol{p}_c^{\mathcal{T}}$ denotes the c^{th} entity annotated within $\boldsymbol{X}_i$, where $\boldsymbol{s}_c$/$\boldsymbol{e}_c$ denotes its start/end position in $\boldsymbol{X}_i$, and $t_c \in \mathcal{T}^k$ is the entity type.

LightNER follows the encoder-decoder architecture of BART, generating the label sequence $\boldsymbol{Y}_i^{\mathcal{T}}$ autoregressively, given $\boldsymbol{X}_i$. The LightNER model for platform k can be trained via minimizing the following loss of cross-entropy,

$$\mathcal{L}_i^{\mathcal{T}^k} = -\sum_{l=1}^{3n} \log P_{\boldsymbol{\theta}}(\boldsymbol{y}_l^{\mathcal{T}} | \boldsymbol{X}_i, \boldsymbol{y}_1^{\mathcal{T}^k}, \cdots, \boldsymbol{y}_{l-1}^{\mathcal{T}^k}), \quad (3)$$

where $\boldsymbol{y}_l^{\mathcal{T}^k}$ is the l^{th} element of $\boldsymbol{Y}_i^{\mathcal{T}^k}$ and $\boldsymbol{\theta}$ denotes the trainable parameters of LightNER.

Prompt Tuning: To preserve the pretrained knowledge from BART for better generalization across domains/platforms, we follow LightNER that freezes the pretrained parameters of BART and inserts tunable prompt parameters for training. Specifically, let $\boldsymbol{q} \in \mathcal{R}^{N_{\boldsymbol{q}} \times D}$ denote an array of $N_{\boldsymbol{q}}$ prompt tokens, where we have $N_{\boldsymbol{q}} = 10$ and $d = 768$. $\boldsymbol{q}$ is projected by a trainable layer into the keys and values of the self-attention in each pretrained transformer layer, with $\boldsymbol{q}$ being shared by all layers. The projection on $\boldsymbol{q}$ follows (Chen et al., 2022) and is detailed in Appendix B. As a result, the number of trainable parameters in the model is significantly reduced, *i.e.*, only 7% of the total model size, compared with fine-tuning all the model parameters. This leads to reduced communication cost for federated learning, considering that we only need to communicate trainable parameters between the server and platforms.

3.2 Distillation with Pseudo-Complete Annotation

The local datasets of each platform only contain annotations of $\{\mathcal{T}^k\}_{k=1}^K$, $T^k \subset \mathcal{T}$. For platform k, we denote entity types that are not annotated locally as $\mathcal{T}^{\backslash k}$, with $\mathcal{T} \cup \mathcal{T}^{\backslash k} = \mathcal{T}$. During local training, if the local trainable parameter $\boldsymbol{\theta}_t^k$ (Algorithm 1) is trained solely with the local annotations of $\mathcal{T}^k$, the resulting NER model will learn to ignore the entities of $\mathcal{T}^{\backslash k}$ from the input text sequences. This contradicts our goal of predicting with the complete tag set $\mathcal{T}$. To solve this problem, we notice that the parameter $\boldsymbol{\theta}_t$ in Algorithm 1 is aggregated from updates of different platforms ($\{\boldsymbol{\theta}_{t-1}^k\}_{k=1}^K$) with $\{\mathcal{T}^k\}_{k=1}^K$. Thus, the NER model with $\boldsymbol{\theta}_t$ should be able to predict with the complete tag set $\mathcal{T}$, including $\mathcal{T}^k$ from each platform k. Additionally, considering that $\boldsymbol{\theta}_t$ is downloaded to each platform before the local training of the t^{th} round of federated learning (Algorithm 1), the model with $\boldsymbol{\theta}_t$ should be locally available for each platform. Inspired by this, we propose to distill from the model with $\boldsymbol{\theta}_t$ while training locally with $\boldsymbol{\theta}_t^k$, so that $\boldsymbol{\theta}_t^k$ can be trained with $\mathcal{T}$ instead of $\mathcal{T}^k$.

Specifically, we extract predictions regarding $\mathcal{T}^{\backslash k}$ from the the model with $\boldsymbol{\theta}_t$ and combine them with the local annotations of $\mathcal{T}^k$, constituting the pseudo-complete annotation. $\boldsymbol{\theta}_t^k$ will be trained with the pseudo-complete annotation of the complete tag set $\mathcal{T}$. Let $\boldsymbol{X}_i$ be the i^{th} text sequence from platform k and $\boldsymbol{Y}_i^{\mathcal{T}^k} = [\boldsymbol{p}_1^{\mathcal{T}^k}; \cdots; \boldsymbol{p}_{n_{loc}}^{\mathcal{T}^k}]$ be the local annotation regarding $\mathcal{T}^k$. n_{loc} is the number of locally annotated entities of $\mathcal{T}^k$. Given $\boldsymbol{X}_i$, we first greedily decode the prediction $\hat{\boldsymbol{Y}}_i^{\mathcal{T}}$ from $\boldsymbol{\theta}_t$,

$$\hat{\boldsymbol{y}}_l^{\mathcal{T}} = arg\,max_{\boldsymbol{y}}\ P_{\boldsymbol{\theta}_t}(\boldsymbol{y} | \boldsymbol{X}_1, \hat{\boldsymbol{y}}_1^{\mathcal{T}}, \cdots, \hat{\boldsymbol{y}}_{l-1}^{\mathcal{T}}). \quad (4)$$

As mentioned above, the prediction $\hat{\boldsymbol{Y}}_i^{\mathcal{T}}$ should have the complete tag set $\mathcal{T}$. Predictions regarding $\mathcal{T}^{\backslash k}$ within $\hat{\boldsymbol{Y}}_i^{\mathcal{T}}$ represents the knowledge of un-annotated entity types in platform k, which is transferred from other platforms. We extract such predictions from $\hat{\boldsymbol{Y}}_i^{\mathcal{T}}$, denoted as, $\hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\backslash k}} = [\boldsymbol{p}_1^{\mathcal{T}^{\backslash k}}; \cdots; \boldsymbol{p}_{n_{trans}}^{\mathcal{T}^{\backslash k}}]$. n_{trans} is the number of entities in $\hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\backslash k}}$. $\{\boldsymbol{p}_c^{\mathcal{T}^{\backslash k}}\}_{c=1}^{n_{trans}}$ is defined as in (2), representing entities that are predicted as types from $\mathcal{T}^{\backslash k}$ with $\boldsymbol{\theta}_t$. We combine $\hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\backslash k}}$ with the existing annotation $\boldsymbol{Y}_i^{\mathcal{T}^k}$ from platform k, generating the pseudo-complete annotation,

$$\boldsymbol{Y}_i^{\mathcal{T}} = [\boldsymbol{Y}_i^{\mathcal{T}^k}; \hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\backslash k}}] = [\boldsymbol{p}_1^{\mathcal{T}}; \cdots; \boldsymbol{p}_{n_{loc}+n_{trans}}^{\mathcal{T}}], \quad (5)$$

where each entity is from either $\boldsymbol{Y}_i^{\mathcal{T}^k}$ or $\hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\backslash k}}$. $\boldsymbol{Y}_i^{\mathcal{T}}$ is constructed to cover the complete tag set $\mathcal{T}$, illustrated in Figure 2. For local training, $\boldsymbol{\theta}_t^k$ is trained with the pseudo-complete annotation $\boldsymbol{Y}_i^{\mathcal{T}}$ instead of $\boldsymbol{Y}_i^{\mathcal{T}^k}$, with the loss of (3).

3.3 Instance Weighting

With (5), $\boldsymbol{\theta}_t^k$ is expected to be trained with $\{\boldsymbol{X}_i, \boldsymbol{Y}_i^{\mathcal{T}}\}_{i=1}^{N_k}$ during local training. Let $\boldsymbol{y}_l$ be the l^{th} element of $\boldsymbol{Y}_i^{\mathcal{T}}$, which can be categorized as

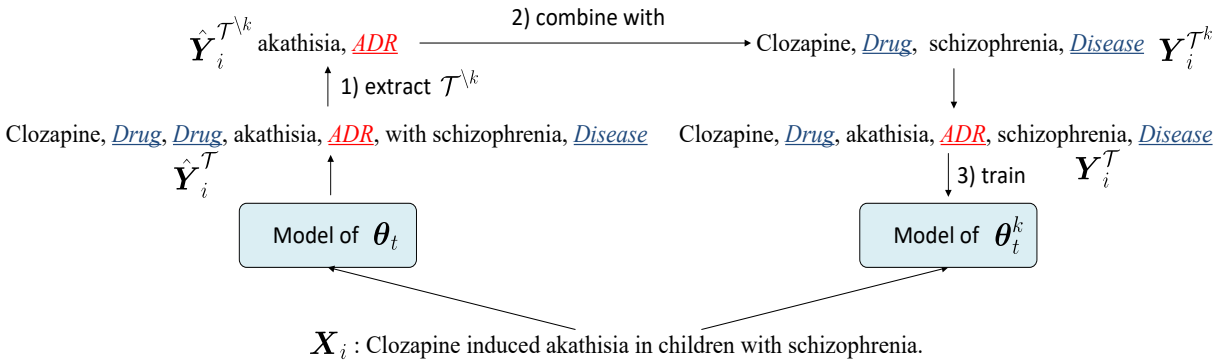


Figure 2: Constructing the pseudo-complete annotation $\boldsymbol{Y}_i^{\mathcal{T}}$ for training of $\boldsymbol{\theta}_i^k$. For platform k, we show entities of $\mathcal{T}^k$ with blue and $\mathcal{T}^{\setminus k}$ as red. For simplicity, we denote the entity $p_c = [s_c, e_c, t_c]$ in the model output directly as the entity span followed by the entity type. For example, $p_c = [0, 1, \textit{Drug}]$ is denoted as "Clozapine, *Drug*". Note that the output with $\boldsymbol{\theta}_t$ may have irregular subsequences, *e.g.*, "*Drug*, *Drug*". We discard every output entity type without appended by a text span. For this case, the second *Drug* is excluded from $\hat{\boldsymbol{Y}}_i^{\mathcal{T}}$ in implementation.

either $\boldsymbol{y}_l \in \boldsymbol{Y}^{\mathcal{T}^k}$ or $\boldsymbol{y}_l \in \hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\setminus k}}$. The training loss can be decomposed as,

$$\mathcal{L}^{\mathcal{T}} = \mathcal{L}^{\mathcal{T}^k} + \mathcal{L}^{\mathcal{T}^{\setminus k}} \tag{6}$$

$$= \frac{1}{N_k}\sum_{i=1}^{N_k} \mathcal{L}_i^{\mathcal{T}^k} + \frac{1}{N_k}\sum_{i=1}^{N_k} \mathcal{L}_i^{\mathcal{T}^{\setminus k}}, \tag{7}$$

where,

$$\mathcal{L}_i^{\mathcal{T}^k} = -\sum_{\boldsymbol{y}_l \in \boldsymbol{Y}_i^{\mathcal{T}^k}} \log P_{\boldsymbol{\theta}_t^k}(\boldsymbol{y}_l | \boldsymbol{X}, \boldsymbol{y}_1^{\mathcal{T}}, \cdots, \boldsymbol{y}_{l-1}^{\mathcal{T}}), \tag{8}$$

$$\mathcal{L}_i^{\mathcal{T}^{\setminus k}} = -\sum_{\boldsymbol{y}_l \in \hat{\boldsymbol{Y}}_i^{\mathcal{T}^{\setminus k}}} \log P_{\boldsymbol{\theta}_t^k}(\boldsymbol{y}_l | \boldsymbol{X}, \boldsymbol{y}_1^{\mathcal{T}}, \cdots, \boldsymbol{y}_{l-1}^{\mathcal{T}}). \tag{9}$$

$\{\mathcal{L}^{\mathcal{T}^k}\}_{k=1}^K$ represents training with the local annotations of $\mathcal{T}^k$. The knowledge learnt from $\mathcal{L}^{\mathcal{T}^k}$ will be transferred to other platforms where annotations of $\mathcal{T}^k$ is not available. Correspondingly, $\{\mathcal{L}^{\mathcal{T}^{\setminus k}}\}_{k=1}^K$ represents how platform k can benefit from knowledge of $\mathcal{T}^{\setminus k}$ that is transferred from the other platforms where annotations for $\mathcal{T}^{\setminus k}$ are available. For platform k, the model is expected to benefit from the knowledge learnt with $\{\mathcal{L}^{\mathcal{T}^{k'}}\}_{k' \neq k}$, regarding entity types that are not locally annotated ($\mathcal{T}^{\setminus k}$), so that it can identify entities of $\mathcal{T}^{\setminus k}$ via training with $\mathcal{L}^{\mathcal{T}^{\setminus k}}$. With this perspective, we denote $\{\mathcal{L}^{\mathcal{T}^k}\}_{k=1}^K$ and $\{\mathcal{L}^{\mathcal{T}^{\setminus k}}\}_{k=1}^K$ as the *source* and *target* loss, respectively, in terms of the direction of knowledge transfer.

To facilitate the knowledge transfer across platforms discussed above, we propose a weighting mechanism for the training instances of the source loss $\{\mathcal{L}^{\mathcal{T}^k}\}_{k=1}^K$, so that the knowledge learnt from the source loss can be more transferable for the target loss $\{\mathcal{L}_t^{\mathcal{T}^{\setminus k}}\}_{k=1}^K$. Specifically, we want to upweight instances that are more beneficial for the training in other platforms and *vise versa*. Formally, we rewrite the source loss as,

$$\mathcal{L}^{\mathcal{T}^k} = \frac{1}{N_k}\sum_{i=1}^{N_k} w_{i,t}^k \times \mathcal{L}_i^{\mathcal{T}^k}, \tag{10}$$

where $w_{i,t}^k = 1$ reduces to (7). $w_{i,t}^k$ is the weight for the i^{th} sample for platform k at the t^{th} federated learning round, measuring how the knowledge from training with $\mathcal{L}_i^{\mathcal{T}^k}$ (source) is transferable for the target loss in other platforms, *i.e.*, $\{\mathcal{L}^{\mathcal{T}^{\setminus k}}\}_{k' \neq k}$ (target). For conciseness, we omit the subscript t that denotes the number of federated learning round in presenting the loss functions, but only showing it for the weight $w_{i,t}^k$.

The question remaining is how to measure the transferablility of knowledge learnt from $\mathcal{L}_i^{\mathcal{T}^k}$ in the federated learning setting. Since the federate learning is a privay-preserving framework that only allows communicating model updates between the server and platforms, we define $\boldsymbol{w}_{i,t}^k$ according to the gradient similarity between the source and target loss. Specifically, for the i^{th} sample of platform k, we first compute the gradients of its source loss and mean of the target loss from other platforms, which we denote as $\boldsymbol{g}_i^{src}$ and $\boldsymbol{g}^{tgt}$, respectively,

$$\boldsymbol{g}_i^{src} = \frac{\partial \mathcal{L}_i^{\mathcal{T}^k}}{\partial \boldsymbol{q}}, \boldsymbol{g}^{tgt} = \sum_{k' \neq k} \frac{\partial (\mathcal{L}^{\mathcal{T}^{\setminus k}})}{\partial \boldsymbol{q}}, \tag{11}$$

$\boldsymbol{q}$ is the prompt embeddings as introduced in Section 3. $\boldsymbol{w}_{i,t}^k$ is updated with the cosine similarity

between the two gradients,

$$w^k_{i,t+1} = \alpha \cdot w^k_{i,t} + (1-\alpha) \cdot \frac{< \boldsymbol{g}^{src}_i, \boldsymbol{g}^{tgt} >}{||\boldsymbol{g}^{src}_i||_2 ||\boldsymbol{g}^{tgt}||_2}, \quad (12)$$

where α is a momentum value. $< \cdot, \cdot >$ denotes the dot product and $|| \cdot ||_2$ is the L_2 norm. $\boldsymbol{w}^k_{i,t}$ is computed before local training (Algorithm 1). For platform k, we save $\boldsymbol{g}^{src}_i$ locally and upload the gradient of the target loss $\mathcal{L}^{\mathcal{T}^{\backslash k}}$ to the server for computing $\boldsymbol{g}^{tgt}$. $\boldsymbol{g}^{tgt}$ is computed on the server side, then downloaded to each platform for updating $w^k_{i,t}$ with (12). We further elaborate the procedures of updating $w^k_{i,t}$ in Algorithm 2. Note that updating $w^k_{i,t}$ does not involving training of the NER model and $w^k_{i,t}$ is not shared to the server or other platforms. We should also notice that the above uploading and downloading of gradients introduce additional communication cost. With such concern, we only compute gradients with respect to $\boldsymbol{q} \in R^{N_{\boldsymbol{q}} \times d}$ (as in (11)), which has only several thousand parameters (Section 3), inducing only minor communication cost. We use $\boldsymbol{q}$ to calculate the gradient similarity, because $\boldsymbol{q}$ is shared by each pretrained layer in BART (Section 3), thus it should correspond to the general information regarding prompt tuning.

4 Related Works

NER with Heterogeneous Tag Sets. Greenberg et al. (2018); Beryozkin et al. (2019) investigate on training over NER datasets with heterogeneous tag sets. However, they assume these datasets are available in a *centralized* location. Such an assumption is not practical in training with clinical data, for which privacy preservation is of primary concern (Hassan Mahlool and Hamzah Abed, 2022). Additionally, they do ot explicitly consider the differences in data distribution for the text from different datasets. Our work is orthogonal to these works, since we assume *decentralized* training, *i.e.*, federated learning, where we account for the issues of privacy and communication costs that do not exist in training with *centralized* datasets.

Federated Domain Adaptation Peng et al. (2019) is the first work studying domain adaptation for federated learning. Recently, Hong et al. (2021) further studies the fairness and debasing problem in federated domain adaptation. These works adopt a discriminator module for adversarial domain adaptation, which increases the communication cost of federated learning. Yao et al. (2022) studies federated domain adaptation via sharing statistics of data distributions of the local platforms. However, such an approach may be vulnerable to membership inference attacks (Shokri et al., 2017), resulting in data leakage, thus may not be applicable to clinical data for which data privacy is the primary concern. Additionally, these work only consider the task of image classification. Our work studies federated domain adaptation for clinical NER. Note that federated domain adaptation is different from federated learning with non-IID (Independent and Identically Distributed) data (*e.g.*,, (Li et al., 2020)). The latter focus on the problem with slow convergence or diverged results in aggregating with updates from non-IID data. Instead, we targets at effectively transferring knowledge across platforms/domains, so that each platform can benefit from knowledge of locally unannotated entity types transferred from other platforms.

Federated Learning for NER. Ge et al. (2020) presents a pilot study of federated learning for clinical NER. Zhao et al. (2021) introduces adversarial training to solve the adversarial attack problem for federated NER. One of the major problems is that these approaches require sharing or communicating the whole NER model (or its encoder) between the server and platforms of federated learning. This will induce huge communication cost in training with the recent Pretrained Langauge Models (PLMs) (Kenton and Toutanova, 2019; Lewis et al., 2019), *i.e.*, containing hundreds of millions of parameters. In this work, we study using a prompt-based pretrained NER model (Chen et al., 2022) for our federated learning, thus only communicating prompt-related parameters. This significantly reduces the communication cost compared to fine tuning all the pretrained parameters. Further, different from Ge et al. (2020); Zhao et al. (2021), we focus on federated domain adaptation that efficiently transfer knowledge among platforms of different domains. (Wu et al., 2021) investigates knowledge distillation in federated learning with NER, but is not targeting the federated domain adaptation problem as in our setting.

5 Experiments

5.1 Baselines and Ablations

We first compare with the classic adversarial domain adaptation with (Ganin et al., 2016), and two more recent works of federated domain adapta-

tion (Peng et al., 2019; Hong et al., 2021). Note that these methods are originally designed for image classification. We re-implement them with our NER model, *i.e.*, LightNER, for comparison. Please refer to Appendix A for details.

Note that these approaches generally require an additional domain discriminator for adversarial domain matching. Such discriminator is trained and communicated along with the NER model. This introduces additional communication cost, as with uploading and downloading the gradients of $\boldsymbol{q}$ in Section 3.3. In the Appendix A, we compared the communication cost of our instance weighting with $\boldsymbol{q}$ with that of the discriminator. Our communication cost is lower, while achieving better performance as in Table 1 and 2.

We denote training with Algorithm 1 as *Ours*. For the ablation study, we consider: (i) *Ours w/o distill&weight*. This is to train the LightNER model without distillation in Section 3.2 and instance weighting in Section 3.3. Specifically, the model is trained with only the local annotation $Y^{\mathcal{T}^k}$ instead of (5) , and $w^t_{i,k}$ is always set to 1. (ii) *Ours w/o weight*. It trains the NER model with (5) (as in *Ours*), while setting $w^t_{i,k} = 1$, *i.e.*, no instance weighting. Please refer to the appendix C for implementation details.

5.2 Experiments with OntoNote 5.0

Before evaluating with clinic data, we first demonstrate our method with OntoNote 5.0, a classic NER dataset of 18 entity types ($|\mathcal{T}| = 18$), with data from six domains: *nw*, *tc*, *wb*, *bn*, *bc* and *mz*. We have the number of platforms $K = 6$, with each platform representing data of a different domain. To simulate the heterogeneous tag sets, we assume the training data of each domain/platform is annotated with 3 entity types ($|\mathcal{T}^k| = 3$), which are randomly sampled from the 18 entity types without replacement. For OntoNote 5.0, we study the challenging scenario of federated domain adaptation that each entity type is only annotated in one of the six platforms, *i.e.*, $\mathcal{T}^{k_1} \cap \mathcal{T}^{k_2} = \emptyset$, for $k_1, k_2 \in \{1, \cdots, K\}$ and $k_1 \neq k_2$. We randomly sample five time and report the F1 score for each domain, averaged over different samplings. For each domain, the F1 score is computed via evaluating the global model on the testing dataset of this domain with all the 18 entity types.

Table 1 shows the resulting F1 score with OntoNote 5.0. Our method outperforms the baselines with a large margin. Instead of communicating domain discriminators as baselines, we communicate the gradients of prompt embeddings, which has a smaller size (Appendix A). Additionally, the performance gain from *Ours w/o distill&weight* to *Ours w/o weight* shows the effectiveness of our distillation with pseudo-complete annotation (Section 3.2), which allows the NER model being trained with the complete tag set during local training. Similarily, the the performance gain from *Ours w/o weight* to *Ours* validates the usefullness of our proposed instance weight mechanism. Both of these techniques contribute to the superior performance of our trained NER model.

5.3 Experiments with the Clinic Datasets

As mentioned in Section 1 and 5.2, the evaluation of our NER model requires testing data of different domains with complete tag sets. However, existing public clinic datasets are generally created with different annotation schemes. For example, datasets may be annotated with different tags sets (Beryozkin et al., 2019; Karimi et al., 2015), and even the same entity type can have various definitions in different datasets (Karimi et al., 2015). Such a lack of consistent annotations for clinic data of different domains poses challenges to the evaluation of our considered setting. Broadly speaking, this also add to the difficulty in studying general transfer learning problems with clinic NER. For instance, the classic domain adaptation (Long et al., 2015) generally involves transferring knowledge from a labeled source domain to an unlabeled target domain. The resulting model is evaluated with testing data of the target domain, annotated with the same classes/entity types as in the source domain, *i.e.*, requiring consistent annotation for data of the source and target domain, which is hardly fullfilled with public datasets when dealing with clinic NER.

To solve this problem, we take three clinic datasets: CADEC (Karimi et al., 2015), ADE (Gurulingappa et al., 2012) and SMM4H (Weissenbacher et al., 2019), which contains text from three distinct text domains, *i.e.*, formal costumer report, medical case report and casual tweets, respectively. We provide some samples of the three dataset in the Supplymentary data. These datasets are originally annotated with different tag sets. To have consistent annotation across domains. We re-annotate ADE and SMM4H with the tag sets defined in CADEC (with the largest tag set). As

Method	nw	tc	wb	bn	bc	mz	Avg
(Ganin et al., 2016)	58.75	56.95	57.31	57.57	58.38	58.15	57.85
(Peng et al., 2019)	61.32	58.32	59.64	60.10	59.75	60.09	59.87
(Hong et al., 2021)	59.27	55.27	55.86	58.16	56.79	57.33	57.11
Ours w/o distill&weight	45.10	52.15	49.42	46.52	47.55	46.71	47.91
Ours w/o weight	59.76	55.09	52.66	58.47	57.00	58.42	56.90
Ours	61.67	59.92	59.52	62.41	60.57	63.22	**61.22**

Table 1: F1 Scores with OntoNote 5.0.

Method	ADE	CADEC	SMM4H	Avg
(Ganin et al., 2016)	66.89	57.26	60.70	61.62
(Peng et al., 2019)	63.77	55.38	58.85	59.33
(Hong et al., 2021)	67.49	57.56	59.02	61.36
Ours w/o distill&weight	41.92	41.68	45.81	43.14
Ours w/o weight	66.80	54.91	60.29	60.67
Ours	69.25	55.68	62.22	**62.38**

Table 2: F1 Scores with Clinic Datasets.

a result, the three datasets are consistently annotated with the same tag set of 5 entity types, $\mathcal{T} = \{Drug, ADE, Disease, Finding, Symptom\}$, as defined in (Karimi et al., 2015). In Appendix D, we also elaborate our annotation procedure and dataset statistics. In simulating our setting of federated domain adaptation with the above datasets, we set the number of local platform $K = 3$. Each platform holds text data of a different domain/dataset. Unlike OntoNote 5.0, we consider a more flexible and practical scenario that allows overlapping among tags set of different platforms. Please refer to Appendix D on the tag sets of annotation for each local platform in experiments.

Table 2 shows the results of federated domain adaptation with our clinic datasets. Our method has the highest F1 score averaged over the three considered datasets/domains. Among the three client datasets, CADEC is larger and more diverse than ADE and SMM4H. Thus, CADEC may contain samples that are quite different from those in ADE and SMM4H, and knowledge learnt with such samples may not be transferable for the training of ADE and SMM4H. From our weighting mechanism (10), such samples can be downweighted during training to facilitate knowledge transfer across platforms. Since such downweighted samples may be important for the local training with CADEC, the improvement for CADEC with our weighting mechanism is slightly smaller than that on the other two clinic datasets. However, we should note that our proposed method can consistently provide improvement over the ablations for different datasets. Table 2 also shows that our annotations for ADE and SMM4H are meaningful, and can be leveraged for the training of existing advanced NER model (Chen et al., 2022). To faciliate future research, we have released our annotated clinic datasets[†].

5.4 Hyperparameter Analysis

Let η be the percentage of trainable parameters in the NER model, which is proportional to the communication cost during federated learning. In order to investigate the relation between the communication cost of federated learning and the model performance, we vary the value of η and plot η with the averaged F1 score on OntoNote 5.0 in Figure 3 (a). η is varied by changing the hidden dimension h of the projection on $\boldsymbol{q}$, explained in Appendix B. Results in Figure 3 (a) shows that, when η is not large ($\eta \leq 10$), the model performance can be improved with larger communication cost (larger η). However, when the value of η gets large enough (*e.g.*, $\eta \geq 10$), the model may overfit to the domain specific information of each client during local training, hindering the further improvement of model performance.

Figure 3 (b) shows the F1 score on OntoNote 5.0, with varing values of $\boldsymbol{E}_{loc}$, *i.e.* epoches of local training per round. All the points share the same communication cost, with the same η and communication rounds for federated learning. The model performance generally improves with longer local training (larger $\boldsymbol{E}_{loc}$). We should note that increasing $\boldsymbol{E}_{loc}$ corresponds to larger computation

[†]https://github.com/RayWangWR/ClinicDataset

cost in local training. The performance gets saturated when $\boldsymbol{E}_{loc}$ get too large, *i.e.*, $\boldsymbol{E}_{loc} \geq 2$, which indicated that the local training may have reached convergence after 2 epoches.

6 Conclusion

In this work, we study the problem of federated domain adaptation for clinic NER. We consider the practical setting with heterogeneous tag sets for different platforms of federated learning. To cope with the heterogeneous tag sets and facilitate knowledge transfer among different platforms, we propose distillation with pseudo-complete annotation and an instance weighting mechanism. In addition, we will release two re-annotated clinic datasets for our considered setting. In experiments, our trained NER model show superior performance in the considered setting.

7 Limitations

Our work is base on the existing sequence-to-sequence NER model, since its way of decoding has been shown effective for knowledge transfer between different classes (Chen et al., 2022). However, it might also be valuable to consider other token-classification-based or CRF-based (Sutton et al., 2012) NER models. Especially, it would be interesting to employ the existing CRF-based distillation method (Wang et al., 2020b) to cope with the problem of heterogeneous tag sets for NER.

8 Acknowledgements

This research was supported by ONR N00014-18-1-2871-P00002-3. The student involved was also supported by Adobe gift research funding. We would like to thank the anonymous reviewers for their insightful comments. Moreover, we want to thank Billy I. Kim for his dedicated efforts in screening the annotations.

References

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR.

Genady Beryozkin, Yoel Drori, Oren Gilon, Tzvika Hartman, and Idan Szpektor. 2019. A joint named-entity recognizer for heterogeneous tag-sets using a tag hierarchy. *arXiv preprint arXiv:1905.09135*.

Xiang Chen, Lei Li, Shumin Deng, Chuanqi Tan, Changliang Xu, Fei Huang, Luo Si, Huajun Chen, and Ningyu Zhang. 2022. Lightner: a lightweight tuning paradigm for low-resource ner via pluggable prompting. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2374–2387.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using bart. *arXiv preprint arXiv:2106.01760*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.

Suyu Ge, Fangzhao Wu, Chuhan Wu, Tao Qi, Yongfeng Huang, and Xing Xie. 2020. Fedner: Privacy-preserving medical named entity recognition with federated learning. *arXiv preprint arXiv:2003.09288*.

Nathan Greenberg, Trapit Bansal, Patrick Verga, and Andrew McCallum. 2018. Marginal likelihood training of bilstm-crf for biomedical named entity recognition from disjoint label sets. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2824–2829.

Harsha Gurulingappa, Roman Klinger, Martin Hofmann-Apitius, and Juliane Fluck. 2010. An empirical evaluation of resources for the identification of diseases and adverse effects in biomedical literature. In *2nd Workshop on Building and evaluating resources for biomedical text mining (7th edition of the Language Resources and Evaluation Conference)*, pages 15–22.

Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. *Journal of biomedical informatics*, 45(5):885–892.

Dhurgham Hassan Mahlool and Mohammed Hamzah Abed. 2022. A comprehensive survey on federated learning: Concept and applications. *arXiv e-prints*, pages arXiv–2201.

Junyuan Hong, Zhuangdi Zhu, Shuyang Yu, Zhangyang Wang, Hiroko H Dodge, and Jiayu Zhou. 2021. Federated adversarial debiasing for fair and transferable representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 617–627.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering*.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Ying Luo, Fengshun Xiao, and Hai Zhao. 2020. Hierarchical contextualized representation for named entity recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8441–8448.

Xingchao Peng, Zijun Huang, Yizhe Zhu, and Kate Saenko. 2019. Federated adversarial domain adaptation. *arXiv preprint arXiv:1911.02054*.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Charles Sutton, Andrew McCallum, et al. 2012. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. 2020a. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Fei Huang, and Kewei Tu. 2020b. Structure-level knowledge distillation for multilingual sequence labeling. *arXiv preprint arXiv:2004.03846*.

Davy Weissenbacher, Abeed Sarker, Arjun Magge, Ashlynn Daughton, Karen O'Connor, Michael Paul, and Graciela Gonzalez. 2019. Overview of the fourth social media mining for health (smm4h) shared tasks at acl 2019. In *Proceedings of the fourth social media mining for health applications (# SMM4H) workshop & shared task*, pages 21–30.

Chuhan Wu, Fangzhao Wu, Ruixuan Liu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2021. Fedkd: Communication efficient federated learning via knowledge distillation. *arXiv preprint arXiv:2108.13323*.

Chun-Han Yao, Boqing Gong, Hang Qi, Yin Cui, Yukun Zhu, and Ming-Hsuan Yang. 2022. Federated multi-target domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1424–1433.

Hanyu Zhao, Sha Yuan, Niantao Xie, Jiahong Leng, and Guoqiang Wang. 2021. A federated adversarial learning method for biomedical named entity recognition. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2962–2969. IEEE.

A Implementation of Baselines

Below we talk about our considered baselines.

(Ganin et al., 2016): It aligns the features of different domains via adversarial matching, using a domain discriminator. We add a K way domain discriminator, on the hidden states of every layer in the the encoder of our model NER model. The discriminator will try to classify the domain from which the data of the hidden states is generated.

(Peng et al., 2019): In addition to adversarial matching with a discriminator, (Peng et al., 2019) also consider enhancing cross-domain generalization via disentangling the task-specific information from the domain-specific information. Therefore, apart from using the discriminator, we also add the disentanglement loss on the last layer of the decoder of our NER model.

(Hong et al., 2021): Similar to (Ganin et al., 2016) and (Peng et al., 2019), (Hong et al., 2021) also uses a K way discriminator for adversarial domain matching. The difference is that it adopt a squared adversarial loss during training, for fairness among local platforms. Additionally, it minimize the prediction entropy of image classification for unlabeled samples. In order to adapt it to our case where the prediction is a label sequence (instead of a single label), we minimize the prediction entropy on the tokens of $\hat{Y}_i^{\mathcal{T}^{\setminus k}}$.

As mentioned in Section 5.1, these approaches generally requires a domain discriminator that is

Algorithm 2 Algorithm for Instance Weighting

Input: $w_{i,t}^k, i = 1, \cdots, N_k, k = 1, \cdots, K$.
Output: $w_{i,t+1}^k, i = 1, \cdots, N_k, k = 1, \cdots, K$.

% Compute and save the gradients of the source and target loss.
for $k = 1, \cdots, K$ **do**
  for $i = 1, \cdots, N_k$ **do**
    Compute the source and target loss, $L_i^{\mathcal{T}^k}$ and $L_i^{\mathcal{T}^{\setminus k}}$, respectively.
    Backpropagate for the gradients $\frac{\partial L_i^{\mathcal{T}^k}}{\partial \boldsymbol{q}}$ and $\frac{\partial L_i^{\mathcal{T}^{\setminus k}}}{\partial \boldsymbol{q}}$, while not updating the model.
  end for
  Save $g_i^{src} = \frac{\partial L_i^{\mathcal{T}^k}}{\partial \boldsymbol{q}}$ locally.
  Compute $\frac{\partial L^{\mathcal{T}^{\setminus k}}}{\partial \boldsymbol{q}} = \sum_{i=1}^{N_k} \frac{\partial L_i^{\mathcal{T}^{\setminus k}}}{\partial \boldsymbol{q}}$, and upload to the server.
end for
% Update the weights with via cosine similarity between gradients.
for $k = 1 \cdots, K$ **do**
  Compute $g^{tgt} = \sum_{k' \neq k} \frac{\partial (L^{\mathcal{T}^{\setminus k}})}{\partial \boldsymbol{q}}$ on the server and download to platform k.
  for $i = 1, \cdots, N_k$ **do**
    Compute the cosine similarity between g_i^{src} and g^{tgt} on platform k.
    Update $w_{i,t}^k$ to $w_{i,t+1}^k$ according to (12).
  end for
end for

trained and communicated along with the model, increasing the communication cost. We use the same K way discriminators for all the baselines. For each layer of BART encoder in LightNER, we add a K way discriminator with a single linear layer. For these discriminators of each layer, the only parameter is a matrix of size $K \times d$, with d being the hidden dimension of the BART encoder.

Communication cost: We quantify it as the number of trainable parameters involved in the model. Since the BART encoder has 12 layers, the communication cost of the discriminators is $12 \times K \times d$, which is $72 \times d$ for OntoNote 5.0 and $36 \times d$ for the clinic datasets.

Comparitively, the communication cost for updating our instance weighting is $N_{\boldsymbol{q}} \times d$, *i.e.*, $10 \times d$, since we have $N_{\boldsymbol{q}} = 10$. Therefore, our instance weighting has less communication cost than the discriminators.

B Details of Prompt Implementation

We following (Chen et al., 2022) in implementing the prompt in Section 3. Generally speaking, (Chen et al., 2022) insert an array of key embeddings and value embeddings into the self-attention module of each transformer layer in BART (Lewis et al., 2019). The inserted key embeddings and value embeddings are denoted as $\boldsymbol{\Phi}_K \in \mathbb{R}^{N_{\boldsymbol{q}} \times d}$ and $\boldsymbol{\Phi}_V \in \mathbb{R}^{N_{\boldsymbol{q}} \times d}$, respectively. Let $\boldsymbol{X}^l$ be the input of a transformer layer in BART. The self-attention module first projects $\boldsymbol{X}^l$ into embeddings of the key ($\boldsymbol{K}^l$), query ($\boldsymbol{Q}^l$) and value ($\boldsymbol{V}^l$),

$$\boldsymbol{K}^l = \boldsymbol{X}^l \boldsymbol{W}^{\boldsymbol{K}}, \boldsymbol{Q}^l = \boldsymbol{X}^l \boldsymbol{W}^{\boldsymbol{Q}}, \boldsymbol{V}^l = \boldsymbol{X}^l \boldsymbol{W}^{\boldsymbol{V}} \tag{13}$$

where $\boldsymbol{W}^{\boldsymbol{K}}, \boldsymbol{W}^{\boldsymbol{Q}}, \boldsymbol{W}^{\boldsymbol{V}} \in \mathbb{R}^{d \times d}$ are the project matrices. The self-attention output with inserted $\boldsymbol{\Phi}_K$ and $\boldsymbol{\Phi}_V$ can be computed as,

$$output^l = softmax(\frac{\boldsymbol{Q}^l [\boldsymbol{K}^l; \boldsymbol{\Phi}_K]_r^\intercal}{d})[\boldsymbol{V}^l; \boldsymbol{\Phi}_V]_r \tag{14}$$

where $output^l$ denote the output from self-attention. $[;]_r$ denotes row concatenation. $\boldsymbol{\Phi}_K$ and $\boldsymbol{\Phi}_V$ are projected from the prompt $\boldsymbol{q} \in \mathbb{R}^{N_{\boldsymbol{q}} \times d}$ in Section 3,

$$[\boldsymbol{\Phi}_K; \boldsymbol{\Phi}_V]_c = \boldsymbol{W}_2^l Tanh(\boldsymbol{W}_1^l \boldsymbol{q}) \tag{15}$$

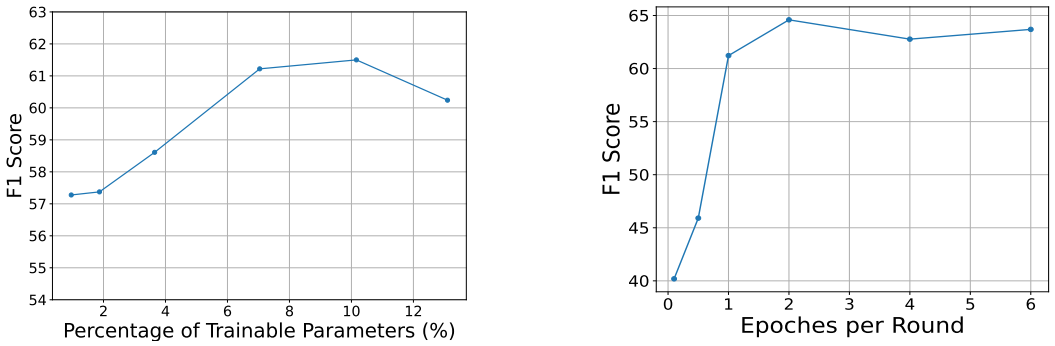
where $[;]_c$ denotes column concatenation. $Tanh$ is the tangent activation. $\boldsymbol{W}_1^l \in \mathbb{R}^{d \times h}$ and $\boldsymbol{W}_1^l \in \mathbb{R}^{h \times 2d}$ are two trainable linear projections for a transformer layer. h is the hidden dimension, controlling the size of trainable parameters.

C Experiment Details

In the experiments, we show results with $N_{\boldsymbol{q}} = 10$, $d = 768$ and $h = 400$. With such configuration, the trainable parameters (those need to be communicated) only takes up 7.04% ($\eta = 7.04$) of the model size, significantly reducing the communication cost compared to finetuning the full model. The model is locally trained for 1 epoch before being upload for aggregation, *i.e.*, $E_{loc} = 1$ (Section 2.2), and train with 25 rounds of communication. We fix the pretrained BART parameters in LightNER, only training and communicating the trainable parameters for federated learning. Our model is trained with learning rate 3e-5 and batch size 8. We empirically set the momentum value $\alpha = 0.9$. We train with a single GPU with pytorch 1.7.0 and python 3.8. For the weights of aggregation in equation (1), $\{m_k\}_{k=1}^K$, we initially tried with FedAvg

Table 3: Statistics of the Clinic Datasets.

Dataset	# Sentences	# Drug	# ADE	# Disease	# Finding	# Symptom
ADE (Gurulingappa et al., 2012)	4258	4077	4652	1169	89	126
SMM4H (Weissenbacher et al., 2019)	1226	1471	1414	135	26	20



(a) (b)

Figure 3: Hyperparameter analysis with (a) the percentage of trainable parameters, η, and (b) epoches of local training per round, E_{loc}. We show the averaged F1 score over the six domains of OntoNote 5.0.

(Wang et al., 2020a) that set m_k as proportional to the size of the dateset in its corresponding domain. However, we found this will lead to inferior results for platforms whose dataset is small in size. Therefore, we set the weights $\{m_k\}_{k=1}^{K}$ as uniform.

D The Clinic Datasets

The labeling procedure: We annotate the text corpous of ADE (Gurulingappa et al., 2012) and SMM4H (Weissenbacher et al., 2019) with a tag set of 5 entity types, *i.e.*, $\mathcal{T} = \{Drug, ADE, Disease, Finding, Symptom\}$, following the definition as in the original paper of CADEC (Karimi et al., 2015). Following (Gurulingappa et al., 2010),we have two annotators that can discuss on the disagreement. We split the text of ADE (Gurulingappa et al., 2012) and SMM4H (Weissenbacher et al., 2019) into batches of 100 sentences. The annotators will work on streaming of batches, and annotating each batch takes about an hour. To ensure the quality of the resulting annotation, we also include a medical student from a clinical institution, in addition to the two annotators, to decide on sample for which the two annotators are not confident. The medical student and the two annotators are all student volunteers, who are also contributing to the methodology and experiments of this research project and credited with their names included in the paper author list. Table 3 show the statistics of our annotations, regarding the number of sentences and identified entities. We have also removed some of the duplicated sentences in SMM4H.

Simulating heterogeneous tag sets for different platforms: As in Section 5.3, our experiments with the clinic datasets consider three platforms for federated learning. During the experiments, we specify different sets of annotated entity types ($\mathcal{T}^k$) for different platforms to simulate local training with heterogeneous tag sets. For instance, if $\mathcal{T}^k$ is specified as annotated in platform k, then annotations of $\mathcal{T}^{\backslash k}$ will be ignored in this platform. $\{\mathcal{T}^k\}_{k=1}^{K}$ are specified such that each platform contains at least one annotated entity types whose annotations are not available in the other platforms. Formally, for each platform k, there exist at least one $s \in \mathcal{T}^k$, *s.t.*, $s \notin \mathcal{T}^{k'}$, $k' \neq k$. In this way, we simulate a practical scenario that each platform will have its unique contribution to the federated learning system, via enabling the global model to recognize at least one entity types whose annotations are only available in this platform. Such a setting is based on the consideration that including more platforms in the federated learning system may increase the risk of backdoor attack (Bagdasaryan et al., 2020) and privacy leakage (Li et al., 2021). Therefore, it is realistic that a platform is allow to participate in federated learning only if it can make unique contributions to the global model, *i.e.*, enabling the global model to recognize entity types that are not

annotated in other platforms. Additionally, since there are 3 platforms, we allow each entity types to be annotated in at most 2 platforms. This is because it is less necessary for knowledge of a certain entity type to be transferred across platforms, if all the three platforms have already had its annotation.

As in Section 5.3, we experiment with 3 platforms ($K = 3$) using the clinic datasets, with text of each platform being from a unique clinic dataset. In determining the $\mathcal{T}^k$ for each platform, we first randomly (uniformly) sample three different entity types (*Drug*, *ADE* and *Disease* as an example) from $\mathcal{T}$, one for each platform. Each of the sampled entity types is specified as uniquely annotated in its associated platform. Then, for each of the rest of the entity types, denoted as s, ($s \in \{$*Finding*, *Symptom*$\}$ in this example), we first randomly decide whether the it is annotated in $n \in \{1, 2\}$ platforms, with a bernoulli distribution of probability 0.5 for each case. Then, we randomly (uniformly) sample n platforms, and assume s is annotated within these platforms. We randomly sample 5 sets of $\{\mathcal{T}^k\}_{k=1}^K$ with the above process. Since the three clinic datasets do not come with training and testing splits. We follow (Ge et al., 2020) that randomly sample 10% of the data in each dataset for testing, while the rest is for local training. We have 3 random split per sampled $\{\mathcal{T}^k\}_{k=1}^K$, and run the experiment with each split and sampled $\{\mathcal{T}^k\}_{k=1}^K$. Following (Ge et al., 2020; Chen et al., 2022) We report the average F1 score of all the experiment runs.

ACL 2023 Responsible NLP Checklist

A For every submission:

☐ A1. Did you describe the limitations of your work?
Left blank.

☐ A2. Did you discuss any potential risks of your work?
Left blank.

☐ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☐ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☐ Did you use or create scientific artifacts?

Left blank.

☐ B1. Did you cite the creators of artifacts you used?
Left blank.

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Left blank.

☐ 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)?
Left blank.

☐ 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?
Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Left blank.

☐ 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.
Left blank.

C ☐ Did you run computational experiments?

Left blank.

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

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?
Left blank.

☐ 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?
Left blank.

☐ 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.)?
Left blank.

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.?
Left blank.

☐ 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)?
Left blank.

☐ 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?
Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Left blank.