Few-Shot Class-Incremental Learning for Named Entity Recognition

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

Abstract

Previous work of class-incremental learning for Named Entity Recognition (NER) relies on the assumption that there exists abundance of labeled data for the training of new classes. In this work, we study a more challenging but practical problem, *i.e.*, few-shot class-incremental learning for NER, where an NER model is trained with only few labeled samples of the new classes, without forgetting knowledge of the old ones. To alleviate the problem of catastrophic forgetting in few-shot class-incremental learning, we generate synthetic data of the old classes using the trained NER model, augmenting the training of new classes. We further develop a framework that distills from the NER model from previous steps with both synthetic data, and real data from the current training set. Experimental results show that our approach achieves significant improvements over existing baselines.

1 Introduction

Existing models of Named Entity Recognition (NER) are usually trained on a large scale dataset with predefined entity classes, then deployed for entity extraction on the test data without further adaptation or refinement. In practice, data of new entity classes that the NER model has not seen during training arrives constantly, thus it is desirable that the NER model can be incrementally updated over time with knowledge of data for these new classes. In this case, one challenge is that the training data of old entity classes may not be available due to privacy concerns or memory limitations (Ma et al., 2020). Then, the model can easily degrade in terms of the performance on old classes when being fine-tuned with only annotations of new entity classes, *i.e.*, *catastrophic forgetting*. In addressing this problem, previous work in class-incremental learning for NER (Monaikul et al., 2021) regularizes the current model by distilling from the previous model trained on old (existing) classes, using text from the training dataset of new classes. However, this requires abundance of data in the new training dataset being used for distillation. Such an assumption is usually unrealistic since the token-level annotations required by NER training are labor-consuming and scarce, especially for the new unseen classes. In this paper, we study a more realistic setting, *i.e.*, few-shot class-incremental learning for NER, where the model (i) incrementally learns on new classes with few annotations, and (ii) without requiring access to training data for old classes.

There is very limited work in few-shot class-incremental learning for NER. Such a setting is more challenging compared with class-incremental learning for NER. First, the few-shot datasets in few-shot class-incremental learning may not contain enough information for the trained model to generalize during testing. Second, it is more challenging to solve the catastrophic forgetting problem in few-shot class-incremental learning when data for old classes is not available and new data is scarce. In class-incremental learning for NER (Monaikul et al., 2021), the same training sequence may contain entities of different classes. Therefore, when the training dataset for new classes is sufficiently large, its context, *i.e.*, words labeled as not from entities of new classes, may also contain abundant entities of the old classes. That is, the new training data can be regarded as an unlabeled *replay* dataset of the existing entity classes. In such case, we can simply address the problem of catastrophic forgetting by distilling from the previous model (trained on old classes) to the current one, using text from such a replay dataset (Monaikul et al., 2021). However, in few-shot class-incremental learning, we cannot expect to avoid catastrophic forgetting by distilling with only the few samples from the new training dataset, since

*Corresponding Author

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 571 - 582
May 22-27, 2022 ©2022 Association for Computational Linguistics

there may not exist sufficient (if any) entities of the old classes.

In this paper, we propose a framework to enable few-shot class-incremental learning for NER. As mentioned above, since the few-shot dataset may not contain enough entities of old classes as *replay* data for distilling from the previous model, which leads to catastrophic forgetting, we consider generating synthetic data of the old entity classes for distillation. Such data is termed as *synthetic replay*. Specifically, we generate synthetic data samples of old classes by inverting the NER model. Given the previous model trained on the old classes, we optimize the token embeddings of the synthetic data, so that predictions from the previous model can contain old entity classes, given the synthetic data as input. In this way, the synthetic data is likely to contain entities of old classes, and distilling from the previous model with such data will thus encourage knowledge preservation of old classes. Additionally, to ensure the synthetic (reconstructed) data to be realistic, we propose to leverage the readily available real text data for new classes, via adversarially matching the hidden features of tokens from the synthetic data and those from the real data. Note that the synthetic data generated from such adversarial match with real data will contain semantics that are close to the real text data for new classes. Consequently, compared with training with only the few samples of new classes, the synthetic data will provide more diverse context that are close to the samples of the few-shot dataset, augmenting the few-shot training for the new classes. Further, with the generated synthetic data, we propose a framework that trains the NER model with annotations of the new classes, while distilling from the previous model with both the synthetic data and real text from the new training data. Our contributions of this work are summarized as follows:

- We present the first work of studying few-shot incremental learning for Named Entity Recognition (NER), a more practical but challenging problem compared with class-incremental learning for NER.

- We approach the problem by proposing a framework that distills from the existing model with both, real data of new entity classes and synthetic data reconstructed from the model as replay data of old entity classes.

Input Sequence:	***Emily***	***from***	***California***	***was***	***born***	***in***	***1990***	***.***
Step 1 :	***PER***	***O***	***O***	***O***	***O***	***O***	***O***	***O***
Step 2 :	***O***	***O***	***LOC***	***O***	***O***	***O***	***O***	***O***
Step 3 :	***O***	***O***	***O***	***O***	***O***	***O***	***TIME***	***O***
Prediction :	***PER***	***O***	***LOC***	***O***	***O***	***O***	***TIME***	***O***

Figure 1: Hypothetical annotations for different time steps in NER few-shot class-incremental learning and the expected model prediction after training at step 3. In our experiments, we do not assume the same sentence is shared by datasets from different time steps.

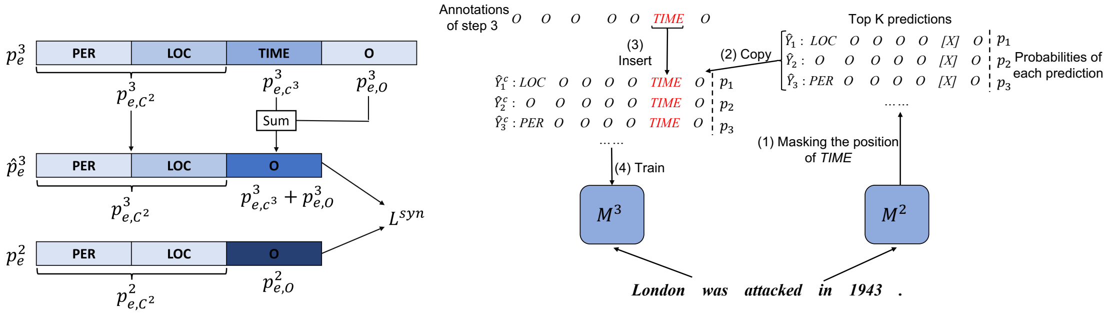
- Experiments show that our method significantly improves over existing baselines for the task of few-shot class-incremental learning in NER.

2 Background

2.1 Problem Definition

Assume there is a stream of NER datasets $\mathcal{D}^1, \ldots, \mathcal{D}^t, \ldots$, annotated with disjoint entity classes, where t is the time step and $\mathcal{D}^t = \{(X_i^t, Y_i^t)\}_{i=1}^{|\mathcal{D}^t|}$ contains c^t entity classes. Here $X_i^t = [x_{i,1}^t, \cdots, x_{i,N_i}^t]$ and $y_i^t = [y_{i,1}^t, \cdots, y_{i,N_i}^t]$ are the NER token and label sequences, respectively, with length N_i, and $|\mathcal{D}^t|$ is the size of the dataset. Dataset $\mathcal{D}^1$ is the base dataset, assumed of reasonably large size for classes of step $t = 1$. The datasets $\{\mathcal{D}^t\}_{t>1}$ are the few-shot datasets with about K samples for each class. In few-shot class-incremental learning, the NER model will be incrementally trained with $\mathcal{D}^1, \mathcal{D}^2, \ldots$, over time, with data from $\mathcal{D}^t$ only available at the t^{th} time step. After being trained with $\mathcal{D}^t$, the model will be evaluated jointly on all entity classes encountered in $\mathcal{D}^1, \cdots, \mathcal{D}^t$, *i.e.*, we do not learn separate prediction modules for each time step. Figure 1 shows an example of annotations for different incremental learning steps on classes of *PER*, *LOC*, and *TIME*.

In Figure 1, we should note that tokens that are labeled as *O* in the current step are likely to contain abundant entities from the previous classes. For instance, tokens annotated as *O* in step 3 include entities of previous classes, *i.e.*, *PER* and *LOC*. Therefore, when a large amount of training data is available for the new classes, the new dataset can be regarded as unlabeled replay data of previous classes. As an example, in Monaikul et al. (2021),


p_e^3
PER
LOC
TIME
O
p^3_{e,c^2}
p^3_{e,c^3}
$p^3_{e,O}$
Sum
$\hat{p}_e^3$
PER
LOC
O
p^3_{e,c^2}
$p^3_{e,c^3} + p^3_{e,O}$
L^{syn}
p_e^2
PER
LOC
O
p^2_{e,c^2}
$p^2_{e,O}$
Annotations of step 3
O O O O O TIME O
(3) Insert
(2) Copy
Top K predictions
$\hat{Y}_1$: LOC O O O O [X] O p_1
$\hat{Y}_2$: O O O O O [X] O p_2
$\hat{Y}_3$: PER O O O O [X] O p_3
Probabilities of each prediction
$\hat{Y}_1^c$: LOC O O O O TIME O p_1
$\hat{Y}_2^c$: O O O O O TIME O p_2
$\hat{Y}_3^c$: PER O O O O TIME O p_3
……
(4) Train
(1) Masking the position of TIME
M^3
M^2
London was attacked in 1943 .

(a) (b)

Figure 2: (a) An example of L^{syn} of Eq (4) at step 3 of Figure 1. (b) An example of distilling with D^t at step 3 of Figure 1. M^2 and M^3 are models from step 2 and 3, respectively. We replace the predictions on the position of "1943" from M^2 with the correct annotation, "TIME", from D^3 before training on M^3.

their performance of class-incremental learning on CoNLL2003 has been comparable or even better than training with full annotations of all the classes encountered, by just distilling with the training data of the new classes. However, in few-shot class-incremental learning, the few training samples of the current step may not contain enough entities of the previous classes. In Section 4, we also discuss the difference between few-shot class-incremental and few-shot learning for NER.

3 Few-Shot Class-Incremental Learning for NER

3.1 The Proposed Framework

Following Beltagy et al. (2019); Souza et al. (2019), we use the BERT-CRF as our NER model, which consists of a BERT$_{base}$ (Devlin et al., 2018) encoder with a linear projection and a conditional random field (CRF) (Lafferty et al., 2001) layer for prediction. We denote M^t as the NER model for step t. M^t is initialized from M^{t-1} to preserve knowledge of old classes. For time step $t > 1$, M^t is expected to learn about the new classes from $\mathcal{D}^t$, while not forgetting the knowledge from $\{\mathcal{D}^k\}_{k=1}^{t-1}$. Assume we have already obtained a synthetic dataset $\mathcal{D}_r^t = \{E_i^{t,r}, Y_i^{t,r}\}_{i=1}^{|\mathcal{D}_r^t|}$ of previous entity classes from $\{\mathcal{D}^k\}_{k=1}^{t-1}$, where $E_i^{t,r} = [e_{i,1}^{t,r}, \cdots, e_{i,N_i}^{t,r}]$ and $Y_i^{t,r} = [y_{i,1}^{t,r}, \cdots, y_{i,N_i}^{t,r}]$ are the reconstructed token embeddings and reference label sequence. $Y_i^{t,r}$ is a randomly sampled label sequence containing classes from the previous steps and $E_i^{t,r}$ is optimized so the output from M^{t-1} with $E_i^{t,r}$ matches $Y_i^{t,r}$. We will discuss the construction of the synthetic $\mathcal{D}_r^t$ in Section 3.2. Given the current training data $\mathcal{D}^t$ and M^{t-1} that has been trained on $\mathcal{D}^{t-1}$, we propose to train M^t by distilling from M^{t-1} with both the real data from D^t and synthetic data from $\mathcal{D}_r^t$. The challenge of such distillation is that the predictions from M^t and M^{t-1} are likely to contain different set of labels, *i.e.*, M^t should also predict with the new entity classes from D^t. This is different from the standard setting of distillation, where the teacher and student models share the same label space (Hinton et al., 2015). In tackling such a problem of label space discrepancy, we propose separate approaches of distillation for $\mathcal{D}^t$ and $\mathcal{D}_r^t$, respectively.

3.1.1 Distilling with Real Data $\mathcal{D}^t$

The distillation from M^{t-1} to M^t involves matching the output distributions between M^t to M^{t-1}. However, given an input sequence X from $\mathcal{D}^t$, the CRF layer outputs correspond to a sequence-level distribution $P_\theta(Y|X)$, *i.e.*, probabilities for all possible label sequences of X, the cardinality of which, grows exponentially large with the length of X. Therefore, it is infeasible to match with the exact output distributions of CRF. Following the current state-of-the-art approach of NER distillation (Wang et al., 2020b), we approximate the sequence-level output distribution of CRF with only its top S predictions. Specifically, for model M^{t-1}, we have,

$$\hat{P}_{M^{t-1}}(Y|X) = [P_{M^{t-1}}(\hat{Y}_1|X), \ldots, \\ P_{M^{t-1}}(\hat{Y}_S|X), 1 - \sum_{s=1}^{S} P_{M^{t-1}}(\hat{Y}_s|X)], \quad (1)$$

where $\{\hat{Y}_s\}_{s=1}^S$ are the top S most probable predictions of label sequence from M^{t-1}. We set $S = 10$. In this way, the output from the CRF of M^{t-1} becomes tractable. However, M^t still cannot be trained with such an output from M^{t-1}. This is because M^{t-1} was not trained with the new classes in $\mathcal{D}^t$. Therefore, when X is from $\mathcal{D}^t$, M^{t-1} will have wrong predictions on the tokens labeled as being from entities of new classes. In order to distill with M^{t-1}, we propose a correction for $\{\hat{Y}_s\}_{s=1}^S$. Figure 2(b) shows an example of such a process. Specifically, on the positions of the sequence where $\mathcal{D}^t$ has labeled as new classes, we replace the predictions in $\{\hat{Y}_s\}_{s=1}^S$ with the annotations from $\mathcal{D}^t$. We denote the corrected set of predictions as $\{\hat{Y}_s^c\}_{s=1}^S$. For training of M^t, we first calculate the predicted distribution of M^t with respect to $\{\hat{Y}_s^c\}_{s=1}^S$, as

$$\begin{aligned}\hat{P}_{M^t}(Y|X) =& [P_{M^t}(\hat{Y}_1^c|X), \cdots, P_{M^t}(\hat{Y}_S^c|X),\\ & 1 - \sum_{s=1}^{S} P_{M^t}(\hat{Y}_s^c|X)],\end{aligned} \tag{2}$$

where we compute the predicted probabilities from M^t with regard to $\{\hat{Y}_s^c\}_{s=1}^S$ from M^{t-1}. Then, M^t can be trained by minimizing the cross entropy between $\hat{P}_{M^{t-1}}(Y|X)$ and $\hat{P}_{M^t}(Y|X)$ via

$$\begin{aligned}&L^{real}(\mathcal{D}^t) = \\ &- \frac{1}{|D^t|}\sum_{X \in D^t} CE(\hat{P}_{M^{t-1}}(Y|X), \hat{P}_{M^t}(Y|X)),\end{aligned} \tag{3}$$

where $CE(\cdot,\cdot)$ is the cross entropy function. Note that the definition of *O* is different in M_{t-1} and M_t. Take Figure 2(b) as an example, the prediction of *O* in step 2 corresponds to both *O* and *TIME* for step 3, since *TIME* is not in the target entity classes of step 2. However, from the annotation of step 3, we know that tokens annotated as *O* are not *TIME*. Therefore, we can safely assume that the prediction of *O* in $\{\hat{Y}_s^c\}_{s=1}^S$ from M_2 matches the definition of *O* in M_3, *i.e.*, the semantics of *O* in $\{\hat{Y}_s^c\}_{s=1}^S$ is the same for M_t and M_{t-1}.

3.1.2 Distilling with Synthetic Data D_r^t

Different from data in $\mathcal{D}^r$, in which we know tokens annotated as *O* are not from the new classes, data from $\mathcal{D}_r^t$ is reconstructed from M_{t-1} and only contains labels for the previous classes. Any token predicted with "O" from M^{t-1} can be potentially labeled as *O* or the new classes by M^t. Therefore, with $\mathcal{D}_r^t$, it is unclear how to correct the output of CRF from M^{t-1}, *i.e.* $\{\hat{Y}_s\}_{s=1}^S$, for training of M^t. By considering the above, we resort to another approach that decomposes the output from CRF, *i.e.*, sequence level label distribution, into marginal label prediction for each token, using the forward and backward method in Lafferty et al. (2001). Figure 2(a) shows a graphic example of our distillation loss L^{syn} with D_r^t. Specifically, let C^t be the cumulative number of possible labels for any given token in NER at step t, *i.e.*, $C^t = \sum_{k=1}^t c^t$, with c^t be the number of class in $\mathcal{D}^t$. For each token with embedding e, we define $p_e^t = [p_{e,O}^t; p_{e,C^{t-1}}^t; p_{e,c^t}^t]$ and $p_e^{t-1} = [p_{e,O}^{t-1}; p_{e,C^{t-1}}^{t-1}]$ as the predicted marginal distribution of a token from M^t and M^{t-1}, respectively. $p_{e,O}^t, p_{e,O}^{t-1} \in \mathbb{R}$ are the probabilities for class O, whereas $p_{e,C^{t-1}}^t, p_{e,C^{t-1}}^{t-1} \in \mathbb{R}^{C^{t-1}}$ are the probabilities for entity classes encountered up to step $t-1$. Further, $p_{e,c^t}^t \in \mathbb{R}^{c^t}$ are probabilities for the new classes in step t. Since *O* from step $t-1$ corresponds to the O and the c^t new classes in step t, we first collapse p_e^t by computing $\hat{p}_e^t = [\text{sum}(p_{e,O}^t, p_{e,c^t}^t); p_{e,C^{t-1}-1}^t]$, where we merge the predictions of *O* and c^t new classes. In this way, $\hat{p}_e^t$ will have the same dimension as p_e^{t-1}. Let E_r^t be the set of embeddings for all tokens contained in $\mathcal{D}_r^t$. The distillation loss for $\mathcal{D}_r^t$ is

$$L^{syn}(D_r^t) = \mathbb{E}_{e \in E_r^t}\text{KL}(\hat{p}_e^t || p_e^{t-1}), \tag{4}$$

where $\text{KL}(\cdot||\cdot)$ is the KL divergence.

3.1.3 General Objective

The general objective of M^t for training at step t is given by

$$L^t = L^{real}(\mathcal{D}^t) + \alpha L^{syn}(\mathcal{D}_r^t), \tag{5}$$

where $L^{real}(\cdot)$ and $L^{syn}(\cdot)$ corresponds to distillation with the real data in $\mathcal{D}^t$ and synthetic data in $\mathcal{D}_r^t$, respectively, and α is a parameter balancing between the losses for $\mathcal{D}^t$ and $\mathcal{D}_r^t$. We set $\alpha = 1$ in the experiment.

3.2 Synthetic Data Reconstruction

Now we describe how to reconstruct $\mathcal{D}_r^t$ from M^{t-1}. Given a randomly sampled label sequence Y containing the old entity classes from $\{\mathcal{D}^k\}_{k<t}$, we seek to reconstruct the embedding sequence E corresponding to its training data. In doing so, we randomly initialize embeddings E, then optimize the parameters of E with gradient descent so that its output with M^{t-1} matches the expected label sequence Y. Formally, we optimize E by minimizing the training loss of the CRF as

$$L^{crf} = -\log P_{M^{t-1}}(Y|E). \tag{6}$$

One problem of such reconstruction is that *the resulting synthetic E may not be realistic.* This will result in a domain gap of training on the synthetic data of old entities but testing on the real data. To alleviate this problem, we propose to encourage synthetic data to be more realistic by leveraging the real data from $\mathcal{D}^t$.

Let $h_l^{t-1,syn}(E_r^t)$ be the hidden state from the l^{th} layer of the BERT encoder in M^{t-1}, regarding the set of synthetic token embeddings, E_r^t. Similarly, let $h_l^{t-1,real}(emb(X^t))$ be the output hidden states from the l^{th} layer of M^{t-1}, regarding the set of real tokens, X^t, from $\mathcal{D}_r^t$. Moreover, $emb(\cdot)$ is the embedding layer. We propose to adversarially match $h_l^{t-1,syn}(E_r^t)$ and $h_l^{t-1,real}(emb(X^t))$ so that hidden states from the real and synthetic are not far away from each other. In this way, the reconstructed embeddings from $\mathcal{D}_r^t$ are likely to be more realistic. Specifically, let M_l be a binary discriminator module, *i.e.*, one layer linear projection with sigmoid output, whose inputs are the real and synthetic hidden states,

$$\begin{aligned} M_l^* &= \text{argmin}_{M_l} - \mathbb{E}_{h \in h_l^{t-1,syn}(E_r^t)} \log M_l(h) \\ &\quad - \mathbb{E}_{h \in h_l^{t-1,real}(emb(X^t))} \log(1 - M_l(h)), \\ L_l^{adv} &= \mathbb{E}_{h \in h_l^{t-1,syn}(E_r^t)} \log(1 - M_l^*(h)). \end{aligned} \tag{7}$$

Finally, the loss for reconstructing D_r^t is

$$L^r = L^{crf} + \beta \sum_{l \in l^s} L_{adv}^l, \tag{8}$$

where $l^s = 2, 4, \cdots, 12$, *i.e.*, we match every two layers of the BERT encoder in M^{t-1}. β is a balancing parameter and is default to 10 in the experiment. Since we train M^t with the reconstructed token embeddings from M^{t-1}, we freeze the BERT token embedding layer during training, so that M^{t-1} and M^t can share the same token embeddings. This is also reasonable for the setting of few shot learning, since tuning all the model parameters with few samples will result in overfitting.

Another problem we should consider is that *the real data $\mathcal{D}^t$ and synthetic data $\mathcal{D}_r^t$ may contain different sets of entity classes*, *i.e.*, the few-shot dataset $\mathcal{D}^t$ may not contain entities of old classes in $\mathcal{D}_r^t$. In this case, for the token embeddings of old classes in $\mathcal{D}_r^t$, *s.t.*, $\{e_{i,j}|y_{i,j}^{t,r} \neq O\}$, matching the hidden states of these embeddings with those from D^t will distract these embedding from being optimized into the entities of old classes, which we will show in the experiments. Therefore, we overload the definition of E_r^t in (4) by excluding embeddings of the old entity classes in $\mathcal{D}_r^t$ from matching, *i.e.*, $E_r^t = \{e_{i,j}|y_{i,j}^{t,r} = O\}$, while X^t contains all the real tokens from $\mathcal{D}^t$. Algorithm 1 shows the complete procedure for constructing $\mathcal{D}_r^t$.

Since $\mathcal{D}_r^t$ contains entities of old classes from previous steps, distilling with $L^{syn}(D_r^t)$ will help preserving knowledge of old entity classes, *i.e.*, avoiding catastrophic forgetting, without accessing the real data from previous steps. Additionally, with $\mathcal{D}_r^t$, M^t is no longer trained with only few samples from $\mathcal{D}^t$, thus is less likely to overfit. This is because $\mathcal{D}_r^t$ can construct a relative larger scale, *e.g.*, several thousand sentences, within the computation limit. Additionally, the semantics of $\mathcal{D}_r^t$ can be close to $\mathcal{D}^t$, since their token embeddings are closely matched. Thus, compared with training only with $\mathcal{D}^t$, $\mathcal{D}_r^t$ provides more diverse text information for M^t during training. Moreover, the entity of old classes from $\mathcal{D}_r^t$ can be regarded as negative samples for training of the new classes in $\mathcal{D}^t$, thus reducing the confusion between old and new classes for M^t during training.

4 Related Work

Class-Incremental Learning: Different from continual learning, *e.g.*, (Hu et al., 2018), which sequentially learn on different tasks (usually with different classes) and requires task labels for prediction, class-incremental learning aims at jointly predicting with all the encountered classes without knowing task labels. Sun et al. (2019); Ke et al. (2021) have study continual learning for different tasks of NLP. Recently, Monaikul et al. (2021) studies class-incremental learning for NER, building a unified NER classifier for all the classes encountered over time. There are two problems regarding this method. Firstly, Monaikul et al. (2021) only works with a non-CRF-based model. However, many current state-of-the-art NER models are built with a CRF module (Liu et al., 2019; Chen et al., 2020; Wang et al., 2021). Secondly, it assumes that a large amount of data for the new classes is available, which is unrealistic since annotations for unseen classes are usually scarce. In this work, we assume only few-shot datasets are available for the new classes, *i.e.*, few-shot class-incremental learning, which was proposed in Tao et al. (2020); Mazumder et al. (2021), yet not studied in NER. Also, note that class-incremental learning is different meta-learning with episode training (Ding et al., 2021; Finn et al., 2017), since tasks/classes of meta-leaning may appear multiple times in episode training, while we assume the dataset of each class only appear once in class-incremental learning.

Few-Shot Learning: Models of few-shot learning are generally trained with a base dataset, then learned to predict unseen target classes with few samples. One branch of the works is based on metric learning. These generally involves predicting by learning to compare token features with class prototypes (Hou et al., 2020) or stored query samples (training data) of target classes (Yang and Katiyar, 2020). The latter violates our setting of class-incremental learning, for which it is prohibitive to store the training data for *e.g.*, privacy issue. Alternatively, Huang et al. (2020) avoids overfitting of few-shot learning by augmenting with noisy or unlabeled data from the web. Our approach is similar to Huang et al. (2020), in that we also augment few-shot training of the current step with additional data, except we use generated synthetic instead of real data. Recently, (Cui et al., 2021) proposes *Template NER*, a few-shot friendly model for NER that convert NER into a sequence-to-sequence problem. Our few-shot class-incremental learning is different from few-shot learning in that *i)* Few-shot learning requires data of different classes arrives at the same time and with complete annotations for all the target classe, while data of few-shot class-incremental learning arrives sequentially, containing annotation of only classes of the current step. *ii)* Existing works of few-shot NER build separate prediction modules for the target and base classes and ignore the performance of base classes during evaluation, thus incompatible with class-incremental learning.

Data-Free Distillation: Data-free distillation refers to the case in which we distill from a teacher model to a student model with the training data of the teacher not available. A typical solution is to reconstruct synthetic training data from the trained teacher model for distillation. Such a setting was previously explored for model compression of image classification (Yin et al., 2020) and text classification (Ma et al., 2020). However, it has not been studied for NER scenarios. We use data-free distillation for transferring knowledge between models from the current and previous steps for few-shot class-incremental learning.

5 Experiments

5.1 Datasets and Implementation

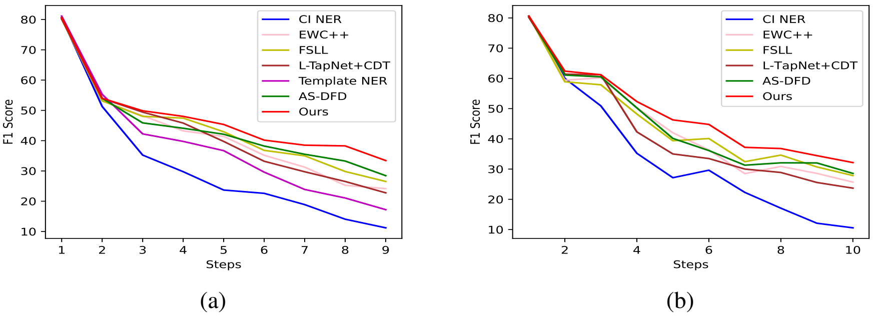
Following the previous work of class-incremental learning for NER (Monaikul et al., 2021), we experiment with two datasets: CoNLL2003 and Ontonote 5.0. For CoNLL2003, our results are average over eight ordering of entity classes for each step as in Monaikul et al. (2021). For Ontonote 5.0, we rank the entities in alphabetic order and experiment with two combinations of different entity classes for different steps. Table 3 and 4 in the Appendix list the entity classes used for each step. Since CoNLL2003 is a relative smaller dataset, we conduct both 5-shot and 10-shot experiments for CoNLL2003 and 5-shot experiments for OntoNote 5.0. Following Yang and Katiyar (2020), our base datasets, *i.e.*, dataset of step 1, is the training data of CoNLL2003 and OntoNote 5.0, labeled with only entity classes included in step 1. The few-shot datasets are sampled from the evaluation dataset with greedy sampling (Yang and Katiyar, 2020). The resulting NER model of each step is tested on the entire test set. Please refer to the Appendix for addition details.

Algorithm 1 Algorithm for constructing D_r^t from M^{t-1}.

```
Input: Model from the previous step, $M^{t-1}$, set of old classes up to $t-1$, $V = \{v_i\}_{i=1}^{C^{t-1}}$.
Output: The reconstructed data $D_r^t$.
$D_r^t = \emptyset$
for $v$ in $V$ do
  for $i$ in $1 \cdots N$ do
    Uniformly sample $n_e \in [1, n_e^{max}]$.
    Uniformly sample $n_s \in [n_e, n_s^{max}]$.
    Uniformly sample $k \in [1, n_s - n_e + 1]$.
    Construct a target label sequence $Y$ of length $n_s$, with a length $n_e$ entity of class $v$ starting from position $k$.
    Randomly initialize an embedding sequence $E$ of length $n_s$.
    while not converge do
      Update $E$ with (8)
    end while
    Add $\{E, Y\}$ to $D_r^t$.
  end for
end for
```

5.2 Baselines and Ablation Study

We compare with the state-of-the art work of class-incremental learning for NER (*CI NER*). Additionally, we implement EWC++ (Chaudhry et al., 2018) with $\alpha = 0$, *i.e.*, using weights regularization to avoid forgetting instead of generating synthetic data. We also implement FSLL (Mazumder et al.,



(a)

(b)

Figure 3: 5-shot Class-Incremental few-shot learning for OntoNote 5.0 data with two combination of classes for each step. (a) P_1 (b) P_2. In the appendix, we include the class combinations, P_1 and P_2, for OntoNote 5.0 and corresponding ablation study.

Table 1: Results for CoNLL2003 5-shot learning.

Method	Step 1	Step 2	Step 3	Step 4	Avg $\geq$ 2
Continual NER	87.89	59.54	51.09	42.98	51.20
EWC++	88.35	68.23	60.34	50.97	59.85
L-TapNet+CDT	88.03	68.57	61.54	51.72	60.61
FSLL	88.35	68.49	61.66	52.71	60.95
AS-DFD	88.35	68.87	60.32	52.99	60.73
Ours ($\alpha = 0$)	88.35	60.09	52.16	44.31	52.19
Ours ($\alpha = 0$, **marg**)	88.35	58.47	51.22	43.19	50.96
Ours ($\beta = 0$)	88.35	69.11	60.54	53.86	61.13
Ours (all tokens)	88.35	69.78	62.33	58.74	63.62
Ours	88.35	**71.31**	**63.76**	**59.37**	**64.18**

Table 2: Results for CoNLL2003 10-shot learning.

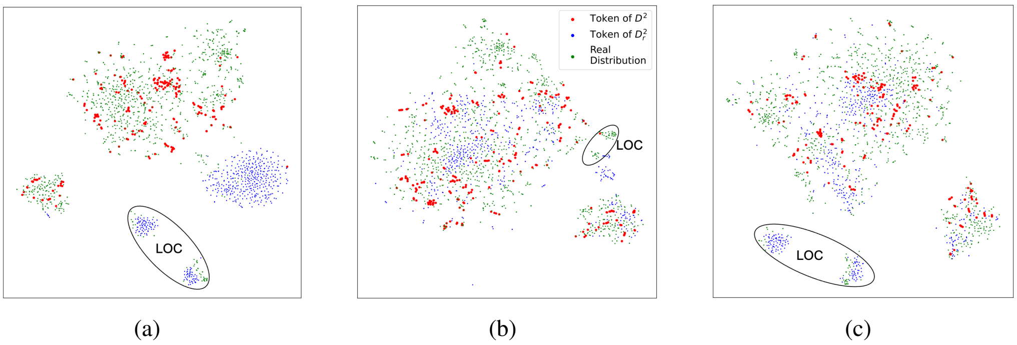
Method	Step 1	Step 2	Step 3	Step 4	Avg $\geq$ 2
Continual NER	87.89	59.77	54.03	46.94	53.58
EWC++	88.35	66.32	62.69	55.14	61.38
L-TapNet+CDT	88.03	66.45	62.43	54.89	61.23
FSLL	88.35	68.34	63.59	56.00	62.71
AS-DFD	88.35	68.95	59.54	53.22	60.57
Ours ($\alpha = 0$)	88.35	60.26	55.46	47.69	54.47
Ours ($\alpha = 0$, **marg**)	88.35	59.67	54.40	46.83	53.63
Ours ($\beta = 0$)	88.35	69.60	60.56	54.59	61.68
Ours (all tokens)	88.35	70.26	61.25	58.69	63.40
Ours	88.35	**70.75**	**64.60**	**60.02**	**65.12**

2021), a state-of-the-art method of few-shot class-incremental learning for *image classification* with metric learning. As mentioned in the related work section, our method can be considered as data-free distillation. Therefore, we also include *AS-DFD* (Ma et al., 2020), the state-of-the-art method of data-free distillation in *text classification*. Specifically, we construct D_r^t with the adversarial regularization described in *AS-DFD* instead of (8). We also adapt *L-TAPNet+CDT* (Hou et al., 2020) for comparison. *L-TAPNet+CDT* is a state-of-the-art work of few-shot learning for sequence labeling with CRF module. Please refer to Appendix for how we adapt it for class-incremental learning.

As an ablation study, we compare our method with: *i) Ours (*$\alpha = 0$*)*, only train with only D^t with $\alpha = 0$. *ii) Ours (*$\alpha = 0$*, marg)*, which is also training with $\alpha = 0$. The difference is that instead of using the sequence-level distillation with $L^{real}(D^t)$, we decompose the output of CRF into marginal predictions for each token, as described before (4). In this way, we can directly apply the token-level distillation in *CI NER* (Monaikul et al., 2021) for the CRF-based NER model. Compared with *Ours (*$\alpha = 0$*)*, this is included to show the performance of directly applying token-level distillation for CRF-based model. *iii)* In *Ours (*$\beta = 0$*)*, we examine the usefulness of L^{adv} by setting $\beta = 0$. *iv) Ours (all tokens)*, which matches all the synthetic tokens in D_r^t with real tokens in D^t, instead of matching with only those labeled as *O* in D_r^t, as described after eq (8).

5.3 Results of Few-Shot Class-Incremental Learning

Table 1 and 2 show the F1 scores from different steps of few-shot class-Incremental learning on CoNLL2003. The values are averaged over eight permutations as in (Monaikul et al., 2021). Our methods outperform all the considered baselines for both 5-shot and 10-shot learning. Especially, *CI NER* (Monaikul et al., 2021) has the worst result among all the methods. This is because the performance of *CI NER* relies on a large amount of data from D^t for replay of previous entities. Therefore, it does not work well in the few-shot scenario, where D^t with only few samples may not contain entities of old classes for replay. Additionally, we find that the performance of *AS-DFD* (Ma et al., 2020) is slightly lower than *Ours (*$\beta = 0$*)*, *i.e.*,



(a)

(b)

(c)

Figure 4: T-SNE plots of real and synthetic token embeddings with 10-shot *LOC*→*PER*, *i.e.*, training on *LOC* for step 1 and *PER* for step 2. D^2 is a 10-shot dataset for *PER*. We visualize the token embedding from the the last layer of the BERT encoder in M^1, *i.e.*, trained only on *LOC*. (a) *Ours* $(\beta = 0)$, no adversarial matching between tokens from D^2 and D_r^2. (b) *Ours (all tokens)*, matching all the tokens from D^2 and all the tokens from D_r^2. (c) *Ours*, excluding the synthetic tokens that are labeled as of entities from old classes, *i.e.*, *LOC*, from adversarial matching. The *real distribution* refers to tokens from the testing dataset, which are not available during training. We use black ellipses to mark tokens from the *real distribution* that are predicted as *LOC*, *i.e.*, the old entities.

distilling using data reconstructed with only L^{crf}. *AS-DFD* is designed for text classification, where they use the feature of the special token *[CLS]* from BERT for classification, while features of the non-special tokens (within text) are trained with an augmented task of language modeling. However, in NER, features of the non-special tokens are directly used for prediction. Thus, simultaneously training such features with language modeling may distract the model from learning the task specific information needed for NER.

In the ablation study, we find that our adversarial matching indeed improves the quality of the synthetic data (*Ours vs. Ours* $(\beta = 0)$), especially when excluding tokens of the reconstructed old entities from matching (*Ours vs. Ours (all tokens)*). Further, *Ours* $(\alpha = 0$, *marg)* has lower performance than *Ours* $(\alpha = 0)$, showing that it might not be optimal to directly apply *CI NER* (Monaikul et al., 2021) with CRF based models.

Figure 3 shows the results of class-incremental with OntoNote 5.0. Since there are more steps relative to the experiments for CoNLL2003, following previous works in few-shot class-incremental learning (Tao et al., 2020; Mazumder et al., 2021), we plot the F1 scores as curves, to highlight the relative difference of different methods over time. Our method consistently outperforms the baselines. Note that with larger number of steps of incremental learning, the curves may not be necessarily monotonically decreasing. This may indeed happen because training with some classes can benefit the performance of other downstream classes, thus (locally) increasing the overall performance. In Appendix, we also present the ablation study with OntoNote 5.0.

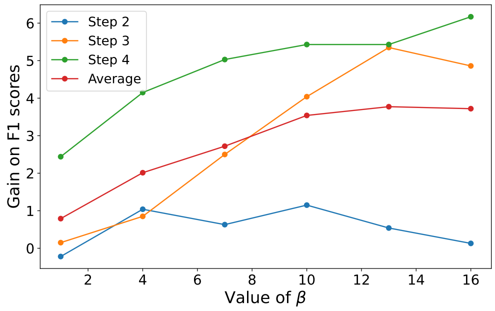


Figure 5: F1 score gains of 10-shot learning with different values of β on CoNLL2003, relative to $\beta = 0$.

5.4 Visualization of Token Embeddings

Figure 4 shows the t-sne plots of hidden states of tokens from 10-shot *LOC*→*PER* (explained in the caption). In (a), we can find that there are synthetic tokens that are very close to the real *LOC* tokens (green dots in the black ellipse). These synthetic tokens (within the black ellipse) are the reconstructed *LOC*. On the contrary, the synthetic context, *i.e.*, the rest of the synthetic tokens outside the ellipse, are far away from the real distribution. This may because the context contains more diverse information, which makes it more difficult to be reconstructed. Such a difference between real and synthetic tokens may cause a domain shift between training and testing, since we are training on synthetic token and testing on real tokens.

Note that there is no tokens from D^2 (red dots) in the black ellipse of *LOC*, indicating that there may not be *LOC* entities in the few-shot dataset D^2, unlike in non-few-shot learning where D^2 can contain a lot of entities of the old classes (*LOC*). (b) shows the result of matching all the synthetic tokens from D^2_r with all the real ones from D^2. In this way, most of the synthetic tokens are matched with the real ones, except that only few synthetic tokens are aligned with the real *LOC* tokens. This is because the few-shot dataset D^2 may not contain entities from the old classes *LOC*. In this case, the adversarial matching will distract synthetic tokens from being reconstructed as *LOC*. Then, the reconstructed embedding sequences will contain less information from the old classes (*LOC*). In (c), we exclude synthetic tokens that are intended to be reconstructed as the old class *LOC*, *i.e.*, labeled as *LOC* in the target label sequence Y in Algorithm 1. As a result, the synthetic tokens contain both *LOC* and context that is aligned with the real distribution.

5.5 Results of Varying β

We investigate the relationship between model performance and the value of β, *i.e.*, the parameter controlling the degree of adversarial matching. Figure 5 shows the the F1 scores from different steps on CoNLL2003, with different values of β. We experiment with 10-shot and report the gain of F1 score compared with $\beta = 0$. We noticed that there is positive gain of the average F1 score on the whole experiment for a range of β values, *i.e.*, $[1, 16]$. These results demonstrate that the proposed adversarial matching between the real and synthetic data (D^t and D^t_r) is generally beneficial and is not sensitive to the selection of β.

6 Conclusion

We present the first work of few-shot class-Incremental learning for NER To address the problem of catastrophic forgetting, we proposed to reconstruct synthetic training of the old entity classes from the model trained at the previous time step. Additionally, the synthetic data allows the model to be trained with a more diverse context, thus less likely to overfit to the few training samples of current step. Experimental results showed that our method outperforms the baselines, enabling the NER model to incrementally learning from new classes with few samples.

Acknowledgements

This work was carried out during an internship at Adobe Research. Further, it was supported by NIH (NINDS 1R61NS120246), DARPA (FA8650-18-2-7832-P00009-12) and ONR (N00014-18-1-2871-P00002-3). We thank all the researchers involved from Adobe Research and the support from Duke University.

References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547.

Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020. Seqvat: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. 2021. Few-nerd: A few-shot named entity recognition dataset. *arXiv preprint arXiv:2105.07464*.

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.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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. *arXiv preprint arXiv:2006.05702*.

Wenpeng Hu, Zhou Lin, Bing Liu, Chongyang Tao, Zhengwei Tao, Jinwen Ma, Dongyan Zhao, and Rui Yan. 2018. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International Conference on Learning Representations*.

Jiaxin Huang, Chunyuan Li, Krishan Subudhi, Damien Jose, Shobana Balakrishnan, Weizhu Chen, Baolin Peng, Jianfeng Gao, and Jiawei Han. 2020. Few-shot named entity recognition: A comprehensive study. *arXiv preprint arXiv:2012.14978*.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting bert for continual learning of a sequence of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4746–4755.

Fanjie Kong and Ricardo Henao. 2021. Efficient classification of very large images with tiny objects. *arXiv preprint arXiv:2106.02694*.

Fanjie Kong, Xiao-Yang Liu, and Ricardo Henao. 2021. Quantum tensor network in machine learning: An application to tiny object classification. *arXiv preprint arXiv:2101.03154*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

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. *arXiv preprint arXiv:1906.02437*.

Xinyin Ma, Yongliang Shen, Gongfan Fang, Chen Chen, Chenghao Jia, and Weiming Lu. 2020. Adversarial self-supervised data-free distillation for text classification. *arXiv preprint arXiv:2010.04883*.

Pratik Mazumder, Pravendra Singh, and Piyush Rai. 2021. Few-shot lifelong learning. *arXiv preprint arXiv:2103.00991*.

Natawut Monaikul, Giuseppe Castellucci, Simone Filice, and Oleg Rokhlenko. 2021. Continual learning for named entity recognition. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning. *arXiv preprint arXiv:1909.03329*.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12183–12192.

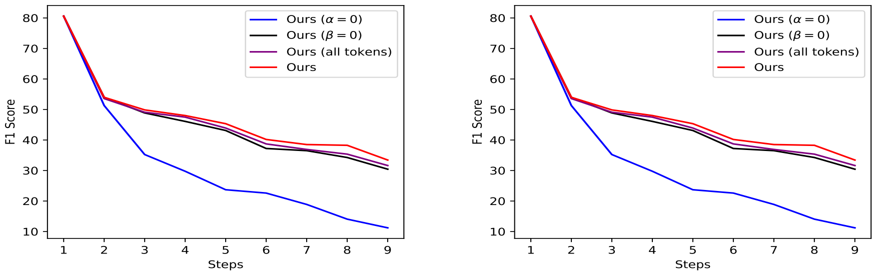
Dong Wang, Yuewei Yang, Chenyang Tao, Zhe Gan, Liqun Chen, Fanjie Kong, Ricardo Henao, and Lawrence Carin. 2020a. Proactive pseudo-intervention: Causally informed contrastive learning for interpretable vision models. *arXiv preprint arXiv:2012.03369*.

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*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Improving named entity recognition by external context retrieving and cooperative learning. *arXiv preprint arXiv:2105.03654*.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. *arXiv preprint arXiv:2010.02405*.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. 2020. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724.



(a)

(b)

Figure 6: Ablation study of 5-shot Class-Incremental few-shot learning for OntoNote 5.0 data with two combination of classes for each step.(a) P_1 (b) P_2.

A Entity Classes for each step

Table 3: Entity classes in each step with CoNLL2003.

Permutations for CoNLL2003
P_1: PER → LOC → ORG → MISC
P_2: PER → MISC → LOC → ORG
P_3: LOC → PER → ORG → MISC
P_4: LOC → ORG → MISC → PER
P_5: ORG → LOC → MISC → PER
P_6: ORG → MISC → PER → LOC
P_7: MISC → PER → LOC → ORG
P_8: MISC → ORG → PER → LOC

Table 4: Entity classes in each step with OntoNote 5.0. Classes within the brackets are trained in the same step.

Permutations for OntoNote 5.0
P_1: [CARDINAL, DATE, EVENT, FAC] → [GPE, LANGUAGE] → LAW → [LOC, MONEY] → NORP → [ORDINAL, ORG]→ PERCENT → [PERSON, PRODUCT] → [QUANTITY, TIME, WORK_OF_ART]
P_2: [CARDINAL, DATE, EVENT, FAC] → GPE → LANGUAGE→ LAW → LOC → [MONEY, NORP] → [ORDINAL, ORG] → [PERCENT, PERSON] → [PRODUCT, QUANTITY] → [TIME, WORK_OF_ART]

B Algorithm for Data Reconstruction

Algorithm 1 shows the procedure of sampling synthetic label sequences and reconstructing the token embedding sequence. Specifically, we construct a label sequence by sampling a length for the token sequence and entity span, respectively.

C Ablation Study for OntoNote 5.0

Figure 6 shows the ablation study on OntoNote 5.0. $\alpha = 0$ corresponds to training only with eq $\mathcal{L}^{real}$. $\beta = 0$ means reconstructing without adversarial training.

D Additional Details

We set the learning rate for our NER model is 5e-5. The model is trained with 50 epochs of D^t, with learning batch size of 1 for 5-shot and 2 for 10-shot experiments. The batch size of training with D^t_r is 5 for each reconstructed class. Following Ma et al. (2020), the token embeddings in D^t_r are initialized with $N(0, 0.35)$, then optimized with a learning rate of 1e-2. We have $n_e^{max} = 4$ and $n_s^{max} = 30$. Our code is modified based on Huggingface[*] with python 3.7 and pytorch 1.7.0, run on 8 P100 GPUs, each with a 16GB memory. Following the prior work[*], when constructing D^t_r, we add the tokens of *[CLS]* and *[SEP]* before and after the sequence of token embeddings that are intended to be reconstructed, so that it is consistent with the input format of BERT. Then, we append after *[SEP]* with *[PAD]* to construct a padded embedding sequence of length 128. In algorithm 1, we freeze the embedding of *[CLS]*, *[SEP]* and *[PAD]*, *i.e.*, the gradients from eq (10) in the main paper are not backpropagated into embeddings of *[CLS]*, *[SEP]* and *[PAD]*. At each time step, we reconstructed 150 samples for each previous class. These samples are discarded after training of the current step.

E Adapting *L-TapNet+CDT* for Class-Incremental Learning

There are two challenge adapting *L-TapNet+CDT* for class-incremental learning. *i)* Unlike dew-shot

[*] https://github.com/huggingface/transformers

[*] Ma, X.; Shen, Y.; Fang, G.; Chen, C.; Jia, C.; and Lu, W.2020. Adversarial Self-Supervised Data-Free Distillationfor Text Classification.arXiv preprint arXiv:2010.04883.

learning, where the few-shot dataset contains complete annotations for all the target classes, dataset of class-incremental leanring only contains data of the current classes, *i.e.*, entities of old or future classes are labeled as O. *ii)* The sematics of O changes with different time step. Without a complete annotation of both current and previous classes, it is unclear how to collect O tokens for constructing the prototype of O for the current class, since tokens labeled as O in the current step may be within span if old entity classes.

To solve the above problem, for each time step, we samples a *fake* few-shot dataset labeled with both current and old classes. We use entities of new classes to construct corresponding prototypes. These prototypes are saved for future steps, *i.e.*, fixed as classification weights for classification of these classes. We use tokens labeled as O in the *fake* dataset for constructing the current O prototypes. Labels for the old classes in the *fake* dataset are not directly used for prediction. Note that this can be a easier problem compared with few-shot class-incremental learning, since the *fake* dataset contains labels of old classes, though we avoid using them for prediction.