

Distilling Causal Effect from Miscellaneous Other-Class for Continual Named Entity Recognition

Junhao Zheng, Zhanxian Liang, Haibin Chen, Qianli Ma*

School of Computer Science and Engineering,
South China University of Technology, Guangzhou, China
junhaozheng47@outlook.com, qianlima@scut.edu.cn*

Abstract

Continual Learning for Named Entity Recognition (CL-NER) aims to learn a growing number of entity types over time from a stream of data. However, simply learning *Other-Class* in the same way as new entity types amplifies the catastrophic forgetting and leads to a substantial performance drop. The main cause behind this is that *Other-Class* samples usually contain old entity types, and the old knowledge in these *Other-Class* samples is not preserved properly. Thanks to the causal inference, we identify that the forgetting is caused by the missing causal effect from the old data. To this end, we propose a unified causal framework to retrieve the causality from both new entity types and *Other-Class*. Furthermore, we apply curriculum learning to mitigate the impact of label noise and introduce a self-adaptive weight for balancing the causal effects between new entity types and *Other-Class*. Experimental results on three benchmark datasets show that our method outperforms the state-of-the-art method by a large margin. Moreover, our method can be combined with the existing state-of-the-art methods to improve the performance in CL-NER.¹

1 Introduction

Named Entity Recognition (NER) is a vital task in various NLP applications (Ma and Hovy, 2016). Traditional NER aims at extracting entities from unstructured text and classifying them into a fixed set of entity types (e.g., *Person*, *Location*, *Organization*, etc). However, in many real-world scenarios, the training data are streamed, and the NER systems are required to recognize new entity types to support new functionalities, which can be formulated into the paradigm of continual learning (CL, a.k.a. *incremental learning* or *lifelong learning*)

*Corresponding author

¹Our codes are publicly available at <https://github.com/zzz47zzz/CFNER>

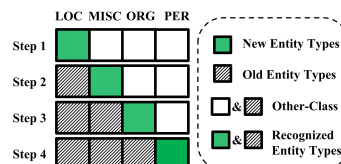


Figure 1: An illustration of *Other-class* in CL-NER. Suppose that a model learns four entity types in CoNLL2003 sequentially. “LOC”: Location; “MISC”: Miscellaneous; “ORG”: Organisation; “PER”: Person.

(Thrun, 1998; Parisi et al., 2019). For instance, voice assistants such as Siri or Alexa are often required to extract new entity types (e.g. *Song*, *Band*) for grasping new intents (e.g. *GetMusic*) (Monaikul et al., 2021).

However, as is well known, continual learning faces a serious challenge called *catastrophic forgetting* in learning new knowledge (McCloskey and Cohen, 1989; Robins, 1995; Goodfellow et al., 2013; Kirkpatrick et al., 2017). More specifically, simply fine-tuning a NER system on new data usually leads to a substantial performance drop on previous data. In contrast, a child can naturally learn new concepts (e.g., *Song* and *Band*) without forgetting the learned concepts (e.g., *Person* and *Location*). Therefore, continual learning for NER (CL-NER) is a ubiquitous issue and a big challenge in achieving human-level intelligence.

In the standard setting of continual learning, only new entity types are recognized by the model in each CL step. For CL-NER, the new dataset contains not only new entity types but also *Other-class* tokens which do not belong to any new entity types. For instance, about 89% tokens belongs to *Other-class* in OntoNotes5 (Hovy et al., 2006). Unlike accuracy-oriented tasks such as the image/text classification, NER inevitably introduces a vast number of *Other-class* samples in training data. As a result, the model strongly biases towards *Other-class* (Li et al., 2020). Even worse, the meaning of *Other-class* varies along with the continual learning pro-

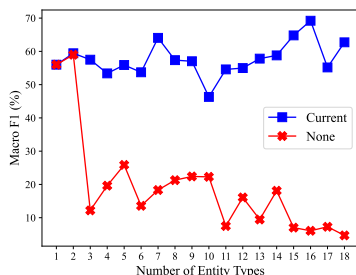


Figure 2: An illustration of the impact of *Other*-class samples on OntoNotes5. We consider two scenarios with different extra annotation levels on *Other*-class samples: (1) annotate all recognized entity types on the data in the current CL step (Current); (2) no extra annotations on *Other*-class samples (None).

cess. For example, “Europe” is tagged as *Location* if and only if the entity type *Location* is learned in the current CL step. Otherwise, the token “Europe” will be tagged as *Other*-class. An illustration is given in Figure 1 to demonstrate *Other*-class in CL-NER. In a nutshell, the continually changing meaning of *Other*-class as well as the imbalance between the entity and *Other*-class tokens amplify the forgetting problem in CL-NER.

Figure 2 is an illustration of the impact of *Other*-class samples. We divide the training set into 18 disjoint splits, and each split corresponds to one entity type to learn. Then, we only retain the labels of the corresponding entity type in each split while the other tokens are tagged as *Other*-class. Next, the NER model learns 18 entity types one after another, as in CL. To eliminate the impact of forgetting, we assume that all recognized training data can be stored. Figure 2 shows two scenarios where *Other*-class samples are additionally annotated with ground-truth labels or not. Results show that ignoring the different meanings of *Other*-classes affects the performance dramatically. The main cause is that *Other*-class contains old entities. From another perspective, the old entities in *Other*-class are similar to the reserved samples of old classes in the data replay strategy (Rebuffi et al., 2017). Therefore, we raise a question: *how can we learn from Other-class samples for anti-forgetting in CL-NER?*

In this study, we address this question with a **Causal Framework for CL-NER (CFNER)** based on causal inference (Glymour et al., 2016; Schölkopf, 2022). Through causal lenses, we determine that the crux of CL-NER lies in establishing causal links from the old data to new entity types

and *Other*-class. To achieve this, we utilize the old model (*i.e.*, the NER model trained on old entity types) to recognize old entities in *Other*-class samples and distillate causal effects (Glymour et al., 2016) from both new entity types and *Other*-class simultaneously. In this way, the causality of *Other*-class can be learned to preserve old knowledge, while the different meanings of *Other*-classes can be captured dynamically. In addition, we design a curriculum learning (Bengio et al., 2009) strategy to enhance the causal effect from *Other*-class by mitigating the label noise generated by the old model. Moreover, we introduce a self-adaptive weight to dynamically balance the causal effects from *Other*-class and new entity types. Extensive experiments on three benchmark NER datasets, *i.e.*, OntoNotes5, i2b2 (Murphy et al., 2010) and CoNLL2003 (Sang and De Meulder, 2003), validate the effectiveness of the proposed method. The experimental results show that our method outperforms the previous state-of-the-art method in CL-NER significantly. The main contributions are summarized as follows:

- We frame CL-NER into a causal graph (Pearl, 2009) and propose a unified causal framework to retrieve the causalities from both *Other*-class and new entity types.
- We are the first to distillate causal effects from *Other*-class for anti-forgetting in CL, and we propose a curriculum learning strategy and a self-adaptive weight to enhance the causal effect in *Other*-class.
- Through extensive experiments, we show that our method achieves the state-of-the-art performance in CL-NER and can be implemented as a plug-and-play module to further improve the performances of other CL methods.

2 Related Work

2.1 Continual Learning for NER

Despite the fast development of CL in computer vision, most of these methods (Douillard et al., 2020; Rebuffi et al., 2017; Hou et al., 2019) are devised for accuracy-oriented tasks such as image classification and fail to preserve the old knowledge in *Other*-class samples. In our experiment, we find that simply applying these methods to CL-NER does not lead to satisfactory performances.

In CL-NER, a straightforward solution for learning old knowledge from *Other*-class samples is

self-training (Rosenberg et al., 2005; De Lange et al., 2019). In each CL step, the old model is used to annotate the *Other*-class samples in the new dataset. Next, a new NER model is trained to recognize both old and new entity types in the dataset. The main disadvantage of *self-training* is that the errors caused by wrong predictions of the old model are propagated to the new model (Monaikul et al., 2021). Monaikul et al. (2021) proposed a method based on knowledge distillation (Hinton et al., 2015) called *ExtendNER* where the old model acts as a teacher and the new model acts as a student. Compared with *self-training*, this distillation-based method takes the uncertainty of the old model’s predictions into consideration and reaches the state-of-the-art performance in CL-NER.

Recently, Das et al. (2022) alleviates the problem of *Other*-tokens in few-shot NER by contrastive learning and pretraining techniques. Unlike them, our method explicitly alleviates the problem brought by *Other*-Class tokens through a causal framework in CL-NER.

2.2 Causal Inference

Causal inference (Glymour et al., 2016; Schölkopf, 2022) has been recently introduced to various computer vision and NLP tasks, such as semantic segmentation (Zhang et al., 2020), long-tailed classification (Tang et al., 2020; Nan et al., 2021), distantly supervised NER (Zhang et al., 2021) and neural dialogue generation (Zhu et al., 2020). Hu et al. (2021) first applied causal inference in CL and pointed out that the vanishing old data effect leads to forgetting. Inspired by the causal view in (Hu et al., 2021), we mitigate the forgetting problem in CL-NER by mining the old knowledge in *Other*-class samples.

3 Causal Views on (Anti-) Forgetting

In this section, we explain the (anti-) forgetting in CL from a causal perspective. First, we model the causalities among data, feature, and prediction at any consecutive CL step with a causal graph (Pearl, 2009) to identify the forgetting problem. The causal graph is a directed acyclic graph whose nodes are variables, and directed edges are causalities between nodes. Next, we introduce how causal effects are utilized for anti-forgetting.

3.1 Causal Graph

Figure 3a shows the causal graph of CL-NER when no anti-forgetting techniques are used. Specifically, we denote the old data as S ; the new data as D ; the feature of new data extracted from the old and new model as X_0 and X ; the prediction of new data as \hat{Y} (*i.e.*, the probability distribution (scores)). The causality between nodes is as follows: (1) $D \rightarrow X \rightarrow \hat{Y}$: $D \rightarrow X$ represents that the feature X is extracted by the backbone model (*e.g.*, BERT (Devlin et al., 2019)), and $X \rightarrow \hat{Y}$ indicates that the prediction \hat{Y} is obtained by using the feature X with the classifier (*e.g.*, a fully-connected layer); (2) $S \rightarrow X_0 \leftarrow D$: these links represent that the old feature representation of new data X_0 is determined by the new data D and the old model trained on old data S . Figure 3a shows that the forgetting happens because there are no causal links between S and \hat{Y} . More explanations about the forgetting in CL-NER are demonstrated in Appendix A.

3.2 Colliding Effects

In order to build cause paths from S to \hat{Y} , a naive solution is to store (a fraction of) old data, resulting in a causal link $S \rightarrow D$ is built. However, storing old data contradicts the scenario of CL to some extent. To deal with this dilemma, Hu et al. (2021) proposed to add a causal path $S \leftrightarrow D$ between old and new data by using *Colliding Effect* (Glymour et al., 2016). Consequently, S and D will be correlated to each other when we control the collider X_0 . Here is an intuitive example: a causal graph $sprinkler \rightarrow pavement \leftarrow weather$ represents the pavement’s condition (wet/dry) is determined by both the weather (rainy/sunny) and the sprinkler (on/off). Typically, the weather and the sprinkler are independent of each other. However, if we observe that the pavement is wet and know that the sprinkler is off, we can infer that the weather is likely to be rainy, and vice versa.

4 A Causal Framework for CL-NER

In this section, we frame CL-NER into a causal graph and identify that learning the causality in *Other*-class is crucial for CL-NER. Based on the characteristic of CL-NER, we propose a unified causal framework to retrieve the causalities from both *Other*-class and new entity types. We are the first to distillate causal effects from *Other*-class for anti-forgetting in CL. Furthermore, we introduce

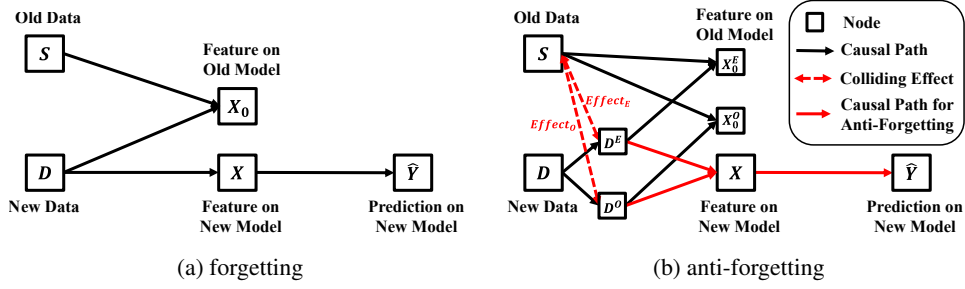


Figure 3: The causal graph for CL-NER: (a) forgetting happens when there are no causal paths from old data to new predictions; (b) anti-forgetting is to build causal paths from old data to new predictions through new entities (D^E) and *Other*-class samples (D^O). We call the causal effects in these two links $Effect_E$ and $Effect_O$, respectively.

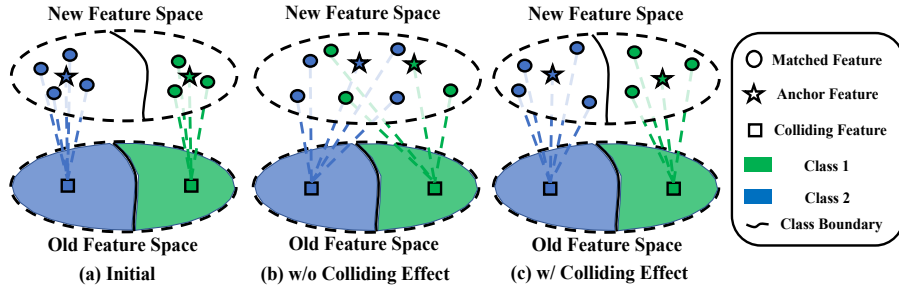


Figure 4: A demonstration of the colliding effect. The anchor token’s feature (*anchor features*) collides with matched tokens’ features (*matched features*) on the *colliding feature* in the old feature space. (a) initial: the class boundary is retained since the new model is initialized by the old model condition. (b) w/o Colliding Effect: the class boundary is forgot in the new feature space since there are no causal effects from old data to new predictions. (c) w/ Colliding Effect: the class boundary is preserved after an CL step since the anchor and matched tokens *collide* in the old feature space.

a curriculum-learning-based strategy and a self-adaptive weight to allow the model to better learn the causalities from *Other*-class samples.

4.1 Problem Formulation

In the i -th CL step, given an NER model M_i which is trained on a set of entities $E_i = \{e_1, e_2 \dots, e_{n_i}\}$, the target of CL-NER is to learn the best NER model M_{i+1} to identify the extended entity set $E_{i+1} = \{e_1, e_2 \dots, e_{n_{i+1}}\}$ with a training set annotated only with $E_i^{New} = \{e_{n_i+1}, e_{n_i+2}, \dots, e_{n_{i+1}}\}$.

Suppose the model consists of a backbone network for feature extraction and a classifier for classification. As a common practice, M_{i+1} is first initialized by the parameter of M_i , and then the dimensions of the classifier are extended to adapt to new entity types. Then, M_i will guide the learning process of M_{i+1} through knowledge distillation (Monaikul et al., 2021) or regularization terms (Douillard et al., 2020) to preserve old knowledge. Our method is based on knowledge distillation where the old model M_i acts as a teacher and the

new model M_{i+1} acts as a student. Our method further distillates causal effects in the process of knowledge distillation.

4.2 Distilling Colliding Effects in CL-NER

Based on the causal graph in Figure 3a, we figure out that the crux of CL lies in building causal paths between old data and prediction on the new model. If we utilize colliding effects, the causal path between old and new data can be built without storing old data.

To distillate the colliding effects, we first need to find tokens in new data which have the same feature representation X_0 in the old feature space, *i.e.*, condition on X_0 . However, it is almost impossible to find such *matched tokens* since features are sparse in high dimensional space (Altman and Krzywinski, 2018). Following Hu et al. (2021), we approximate the colliding effect using K-Nearest Neighbor (KNN) strategy. Specifically, we select a token as *anchor token* and search the k-nearest neighbor tokens whose features bear a resemblance to the *anchor token’s* feature in the old feature

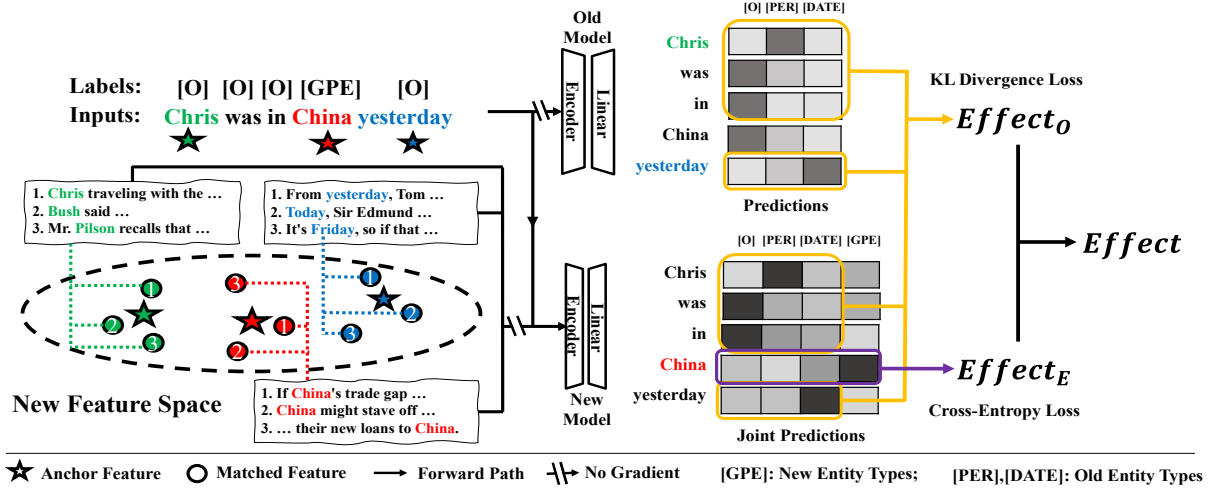


Figure 5: A demonstration of the proposed causal framework for CL-NER.

space. Next, when calculating the prediction of the anchor token, we use *matched tokens* for joint prediction. Note that in backpropagation, only the gradient of the anchor token is computed. Figure 4 shows a demonstration for distilling colliding effects.

Although *Other*-class tokens usually do not directly guide the model to recognize new entity types, they contain tokens from old entity types, which allow models to recall what they have learned. Naturally, we use the old model to recognize the *Other*-class tokens which actually belong to old entity types. Since these *Other*-class tokens belong to the predefined entity types, we call them as *Defined-Other-Class* tokens, and we call the rest tokens in *Other*-class as *Undefined-Other-Class* tokens.

Based on the characteristics of NER, we extend the causal graph in Figure 3a to Figure 3b. The key adjustment is that the node of new data is split into two nodes, including new entity tokens D^E and *Defined-Other-Class* tokens D^O . Then, we apply the colliding effects on D^E and D^O respectively, resulting in that D^E and D^O collide with S on nodes X_0^E and X_0^O in the old feature space. In this way, we build two causal paths from old data S to new predictions \hat{Y} . In the causal graph, we ignore the token from *Undefined-Other-Class* since they do not help models learn new entity types or review old knowledge. Moreover, we expect the model to update instead of preserving the knowledge about *Other*-class in each CL step. Here, we consider two paths separately because the colliding effects are distilled from different kinds of data and calculated

in different ways.

4.3 A causal framework for CL-NER

Formally, we define the total causal effects *Effect* as follow:

$$Effect = Effect_E + Effect_O \quad (1)$$

$$= -\sum_i CE(\bar{Y}_i, Y_i) - \sum_j KL(\bar{Y}_j, \tilde{Y}_j), \quad (2)$$

$$s.t. \quad D_i \in D^E, \quad D_j \in D^O.$$

In Eq.(1), $Effect_E$ and $Effect_O$ denote the colliding effect of new entity types and *Defined-Other-Class*. In Eq.(2), $CE(\cdot, \cdot)$ and $KL(\cdot, \cdot)$ represent the cross-entropy and KL divergence loss, and D_i, D_j are the i -th, j -th token in new data. In cross-entropy loss, Y_i is the ground-truth entity type of the i -th token in new data. In KL divergence loss, \tilde{Y}_j is the soft label of the j -th token given by the old model over old entity types. In both losses, \bar{Y} represents the weighted average of prediction scores over *anchor* and *matched* tokens. When calculating $KL(\cdot, \cdot)$, we follow the common practice in knowledge distillation to introduce temperature parameters T_t, T_s for teacher (old model) and student model (new model) respectively. Here, we omit T_t, T_s in $KL(\cdot, \cdot)$ for notation simplicity. The weighted average scores of the i -th token is calculated as follow:

$$\bar{Y}_i = W_i \hat{Y}_i + \sum_{k=1}^K W_{ik} \hat{Y}_{ik} \quad (3)$$

$$s.t. \quad W_i \geq W_{i1} \geq W_{i2} \geq \dots \geq W_{iK}$$

$$W_i + \sum_{k=1}^K W_{ik} = 1,$$

where the i -th token is the *anchor* token and K *matched* tokens are selected according to the KNN

strategy. We sort the K *matched* tokens in ascending order according to the distance to the *anchor* token in the old feature space. \hat{Y}_i, \hat{Y}_{ik} are the prediction scores of the i -th token and its k -th *matched* token, and W_i, W_{ik} are the weight for \hat{Y}_i, \hat{Y}_{ik} , respectively. The weight constraints ensure that the token closer to the *colliding* feature has a more significant effect.

Until now, we calculate the effects in D^E and D^O and ignore the *Undefined-Other-Class*. Following Monaikul et al. (2021), we apply the standard knowledge distillation to allow new models to learn from old models. To this end, we just need to rewrite $Effect_O$ in Eq.(1) as follow:

$$Effect_O = -\sum_j KL(\bar{Y}_j, \tilde{Y}_j) - \sum_n KL(\hat{Y}_n, \tilde{Y}_n) \quad (4)$$

$$s.t. \quad D_j \in D^O, \quad D_n \in D^{UO},$$

where D^{UO} is the data belong to the *Undefined-Other-Class*. The Eq.(4) can be seen as calculating samples from D^O and D^{UO} in the same way, except that samples from D^{UO} have no *matched* tokens. We summarize the proposed causal framework in Figure 5.

4.4 Mitigating Label Noise in $Effect_O$

In our method, we use the old model to predict the labels of *Defined-Other-Class* tokens. However, it inevitably generates label noise when calculating $Effect_O$. To address this problem, we adopt curriculum learning to mitigate the label noise in the proposed method.

Curriculum learning has been widely used for handling label noises (Guo et al., 2018) in computer vision. Arazo et al. (2019) empirically find that networks tend to fit correct samples before noisy samples. Motivated by this, we introduce a confidence threshold δ ($\delta \in [0, 1]$) to encourage the model to learn first from clean *Other-class* samples and then noisier ones. Specifically, when calculating $Effect_O$, we only select *Defined-Other-Class* tokens whose predictive confidences are larger than δ for distilling colliding effects while others are for knowledge distillation. The value of δ changes along with the training process and the value of δ in the i -th epoch is calculated as follow:

$$\delta_i = \begin{cases} \delta_1 + \frac{i-1}{m-1}(\delta_m - \delta_1), & 1 \leq i \leq m \\ \delta_m, & i > m, \end{cases} \quad (5)$$

where m, δ_1 and δ_m are the predefined hyperparameters and δ_m should be smaller than δ_1 .

4.5 Balancing $Effect_E$ and $Effect_O$

Figure 5 shows that the total causal effect $Effect$ consists of $Effect_E$ and $Effect_O$, where $Effect_E$ is for learning new entities while $Effect_O$ is for reviewing old knowledge. With the learning process of CL, the need to preserve old knowledge varies (Hou et al., 2019). For example, more efforts should be made to preserve old knowledge when there are 15 old classes and 1 new class v.s. 5 old classes and 1 new class. In response to this, we introduce a self-adaptive weight for balancing $Effect_E$ and $Effect_O$:

$$\lambda = \lambda_{base} \sqrt{C_O/C_N} \quad (6)$$

where λ_{base} is the initial weight and C_O, C_N are the numbers of old and new entity types respectively. In this way, the causal effects from new entity types and *Other-class* are dynamically balanced when the ratio of old classes to new classes changes. Finally, the objective of the proposed method is given as follow:

$$\max Effect = Effect_E + \lambda * Effect_O \quad (7)$$

5 Experiments

5.1 Settings

Datasets. We conduct experiments on three widely used datasets, *i.e.*, OntoNotes5 (Hovy et al., 2006), i2b2 (Murphy et al., 2010) and CoNLL2003 (Sang and De Meulder, 2003). To ensure that each entity type has enough samples for training, we filter out the entity types which contain less than 50 training samples. We summarize the statistics of the datasets in Table 5 in Appendix F.

Following Monaikul et al. (2021), we split the training set into disjoint slices, and in each slice, we only retain the labels which belong to the entity types to learn while setting other labels to *Other-class*. Different from Monaikul et al. (2021), we adopt a greedy sampling strategy to partition the training set to better simulate the real-world scenario. Specifically, the sampling algorithm encourages that the samples of each entity type are mainly distributed in the slice to learn. We provide more explanations and the detailed algorithm in Appendix B.

Training. We use bert-base-cased (Devlin et al., 2019) as the backbone model and a fully-connected layer for classification. Following previous work in CL (Hu et al., 2021), we predefine a fixed order of

Table 1: Comparisons with state-of-the-art methods on I2B2 and OntoNotes5. The average results as well as standard derivations are provided. *Mi-F1*: micro-F1; *Ma-F1*: macro-F1; *Forget*: Forgetting; : higher is better; : lower is better. The best F1 results are bold.

Dataset	Method	FG-1-PG-1		FG-2-PG-2		FG-8-PG-1		FG-8-PG-2	
		Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
I2B2	Finetune Only	17.43	13.81	28.57	21.43	20.83	18.11	23.60	23.54
		± 0.54	± 1.14	± 0.26	± 0.41	± 1.78	± 1.66	± 0.15	± 0.38
	PODNet	12.31	17.14	34.67	24.62	39.26	27.23	36.22	26.08
		± 0.35	± 1.03	± 2.65	± 1.76	± 1.38	± 0.93	± 12.9	± 7.42
	LUCIR	43.86	31.31	64.32	43.53	57.86	33.04	68.54	46.94
		± 2.43	± 1.62	± 0.76	± 0.59	± 0.87	± 0.39	± 0.27	± 0.63
	ST	31.98	14.76	55.44	33.38	49.51	23.77	48.94	29.00
	± 2.12	± 1.31	± 4.78	± 3.13	± 1.35	± 1.01	± 6.78	± 3.04	
ExtendNER	42.85	24.05	57.01	35.29	43.95	23.12	52.25	30.93	
	± 2.86	± 1.35	± 4.14	± 3.38	± 2.01	± 1.79	± 5.36	± 2.77	
CFNER(Ours)	62.73	36.26	71.98	49.09	59.79	37.3	69.07	51.09	
	± 3.62	± 2.24	± 0.50	± 1.38	± 1.70	± 1.15	± 0.89	± 1.05	
OntoNotes5	Finetune Only	15.27	10.85	25.85	20.55	17.63	12.23	29.81	20.05
		± 0.26	± 1.11	± 0.11	± 0.24	± 0.57	± 1.08	± 0.12	± 0.16
	PODNet	9.06	8.36	34.67	24.62	29.00	20.54	37.38	25.85
		± 0.56	± 0.57	± 1.08	± 0.85	± 0.86	± 0.91	± 0.26	± 0.29
	LUCIR	28.18	21.11	64.32	43.53	66.46	46.29	76.17	55.58
		± 1.15	± 0.84	± 1.79	± 1.11	± 0.46	± 0.38	± 0.09	± 0.55
	ST	50.71	33.24	68.93	50.63	73.59	49.41	77.07	53.32
	± 0.79	± 1.06	± 1.67	± 1.66	± 0.66	± 0.77	± 0.62	± 0.63	
ExtendNER	50.53	32.84	67.61	49.26	73.12	49.55	76.85	54.37	
	± 0.86	± 0.84	± 1.53	± 1.49	± 0.93	± 0.90	± 0.77	± 0.57	
CFNER(Ours)	58.94	42.22	72.59	55.96	78.92	57.51	80.68	60.52	
	± 0.57	± 1.10	± 0.48	± 0.69	± 0.58	± 1.32	± 0.25	± 0.84	

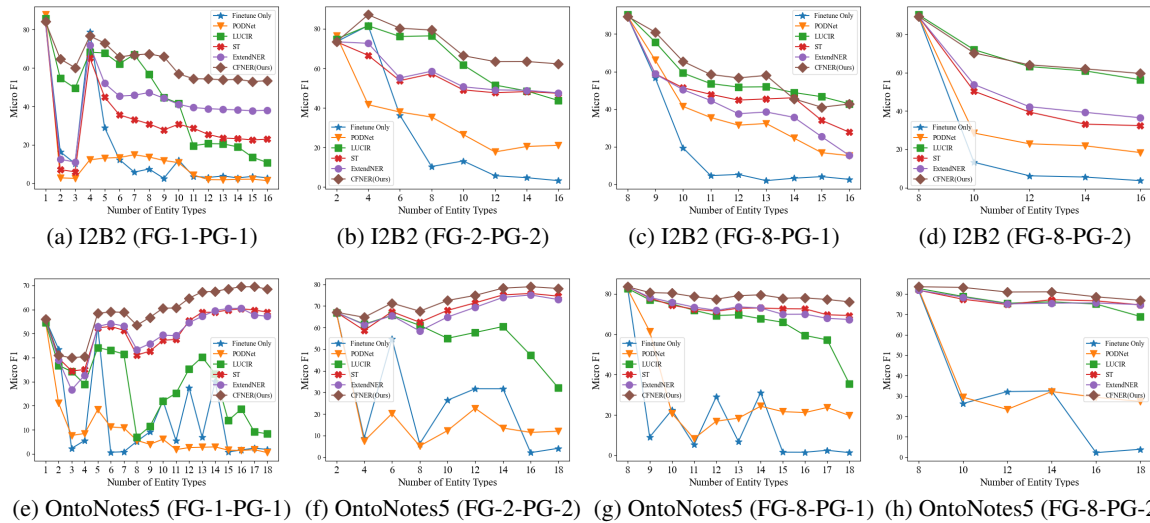


Figure 6: Comparison of the step-wise micro-f1 score on I2B2 (16 entity types), OntoNotes5 (18 entity types).

classes (alphabetically in this study) and train models with the corresponding data slice sequentially. Specifically, *FG* entity types are used to train a model as the initial model, and every *PG* entity types are used for training in each CL step (denoted as *FG-a-PG-b*). For evaluation, we only retain the new entity types' labels while setting other labels to *Other*-class in the validation set. In each CL step, we select the model with the best validation

performance for testing and the next step's learning. For testing, we retain the labels of all recognized entity types while setting others to *Other*-class in the test set.

Metrics. Considering the class imbalance problem in NER, we adopt Micro F1 and Macro F1 for measuring the model performance. We report the average result on all CL steps (including the first step) as the final result.

Baselines. We consider four baselines: ExtendNER (Monaikul et al., 2021), Self-Training (ST) (Rosenberg et al., 2005; De Lange et al., 2019), LUCIR (Hou et al., 2019) and PODNet (Douillard et al., 2020). ExtendNER is the previous state-of-the-art method in CL-NER. LUCIR and PODNet are state-of-the-art CL methods in computer vision. Detailed descriptions of the baselines and their training settings are demonstrated in Appendix C.

Hyper-Parameters. We set the number of matched tokens $K = 3$, the weights $W_i = 1/2$ and $W_{ik} = \frac{1}{2K}$. For parameters in the curriculum learning strategy, we set $\delta_1 = 1$, $\delta_m = 0$ and $m = 10$. We set the initial value of balancing weight $\lambda_{base} = 2$. More training details are shown in Appendix D.

5.2 Results and Analysis

Comparisons with State-Of-The-Art. We consider two scenarios for each dataset: (1) training the first model the same as the following CL steps; (2) training the first model with half of all entity types. The former scenario is more challenging, whereas the latter is closer to the real-world scenario since it allows models to learn enough knowledge before incremental learning. Apart from that, we consider fine-tuning without any anti-forgetting techniques (Finetune Only) as a lower bound for comparison.

The results on I2B2 and OntoNotes5 are summarized in Table 1 and Figure 6. Due to the space limitation, we provide the results on CoNLL2003 in Table 6 and Figure 9 in Appendix F. In most cases, our method achieves the best performance. Especially, our method outperforms the previous state-of-the-art method in CL-NER (*i.e.*, ExtendNER) by a large margin. Besides, we visualize the features of our method and ExtendNER for comparison in Appendix E. The performances of PODNet and LUCIR are much worse than our methods when more CL steps are performed. The reason could be that neither of them differentiates *Other*-class from entity types, and the old knowledge in *Other*-class is not preserved. Our method encourages the model to review old knowledge from both new entity types and *Other*-class in the form of distilling causal effects.

Ablation Study. We ablate our method, and the results are summarized in Table 2. To validate the effectiveness of the proposed causal framework, we only remove the colliding effects in *Other*-class and new entity types for the settings *w/o Effect_O*

Table 2: The ablation study of our method on three datasets in the setting FG-1-PG-1. *AW*: adaptive weight; *CuL*: curriculum learning strategy; *Mi-F1*: micro-F1; *Ma-F1*: macro-F1.

Methods	I2B2		OntoNotes5		CoNLL2003	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
CFNER(Ours)	62.73	36.26	58.94	42.22	80.91	79.11
w/o AW	61.65	35.86	57.63	40.28	80.75	78.43
w/o CuL	61.21	34.79	57.95	39.54	80.32	78.71
w/o AW & CuL	60.78	33.15	56.83	38.95	79.89	77.54
w/o <i>Effect_O</i>	59.68	30.56	53.09	34.99	78.68	76.15
w/o <i>Effect_E</i>	53.62	28.75	54.88	37.29	79.83	77.45
w/o <i>Effect_O</i> & <i>Effect_E</i>	42.85	24.05	50.53	32.84	76.36	73.04

and *w/o Effect_E*, respectively. Specifically, in the *w/o Effect_O* setting, we apply knowledge distillation for all *Other*-class samples, while in *w/o Effect_E* setting, we calculate the cross-entropy loss for classification. Note that our model is the same as ExtendNER when no causal effects are used (*i.e.*, *w/o Effect_O* & *Effect_E*). The results show that both *Effect_O* and *Effect_E* play essential roles in our framework. Furthermore, the adaptive weight and the curriculum-learning strategy help model better learn causal effects in new data.

Table 3: Hyper-parameter analysis on I2B2 (FG-8-PG-2). *Mi-F1*: micro-F1; *Ma-F1*: macro-F1.

K	Mi-F1	Ma-F1	δ_1	Mi-F1	Ma-F1	λ_{base}	Mi-F1	Ma-F1
1	65.48	46.82	0	65.25	46.33	0.5	69.27	48.71
2	67.12	49.37	0.5	66.09	48.51	1	69.46	52.45
3	69.07	51.09	0.9	68.23	49.1	2	69.07	51.09
5	70.25	52.51	0.95	68.64	49.66	5	64.4	46.65
10	70.69	52.26	1	69.07	51.09	10	54.12	40.66

Hyper-Parameter Analysis. We provide hyper-parameter analysis on I2B2 with the setting FG-8-PG-2. We consider three hyper-parameters: the number of matched tokens K , the initial value of balancing weight λ_{base} and the initial value of confidence threshold δ_1 . The results in Table 3 shows that a larger K is beneficial. However, as K becomes larger, the run time increases correspondingly. The reason is that more forward passes are required during training. Therefore, We select $K = 3$ by default to balance effectiveness and efficiency. Results also show that $\delta_1 = 1$ reaches the best result, which indicates that it is more effective to learn *Effect_E* first and then gradually introduce *Effect_O* during training. Otherwise, the old model’s wrong predictions will significantly affect the model’s performance. Additionally, we find that the performance drops substantially when λ_{base} is too large. Note that we did not carefully search for the best hyper-parameters, and the default ones are used throughout the experi-

ments. Therefore, elaborately adjusting the hyper-parameters may lead to superior performances on specific datasets and scenarios.

Table 4: Combining our methods with other baselines on three datasets in the setting FG-1-PG-1. *Mi-F1*: micro-F1; *Ma-F1*: macro-F1. *CF* represents applying causal effects.

Methods	I2B2		OntoNotes5		CoNLL2003	
	Mi-F1	Ma-F1	Mi-F1	Ma-F1	Mi-F1	Ma-F1
CFNER(Ours)	62.73	36.26	58.94	42.22	80.91	79.11
LUCIR	43.86	31.31	28.18	21.11	74.15	70.48
LUCIR+CF	66.27	38.52	62.03	44.34	81.28	79.56
ST	31.98	14.76	50.71	33.24	76.17	72.88
ST+CF	61.41	33.43	57.06	41.28	80.59	79.11

Combining with Other Baselines. Furthermore, the proposed causal framework can be implemented as a plug-and-play module (denoted as *CF*). As shown in Figure 5, $Effect_E$ is based on the cross-entropy loss for classifying new entities, while $Effect_O$ is based on the KL divergence loss for the prediction-level distillation. We use LUCIR and ST as the baselines for a demonstration. To combine LUCIR with our method, we substitute the classification loss in LUCIR with $Effect_E$ and substitute the feature-level distillation with $Effect_O$. When combining with ST, we only replace the soft label with the hard label when calculating $Effect_O$. The results in Table 4 indicate that our method improves LUCIR and ST substantially. It is worth noticing that *LUCIR+CF* outperforms our method consistently, indicating our method has the potential to combine with other CL methods in computer vision and reach a superior performance in CLNER. In addition, *CFNER* outperforms *ST+CF* due to the fact that soft labels convey more knowledge than hard labels.

6 Conclusion

How can we learn from *Other*-class for anti-forgetting in CL-NER? This question is answered by the proposed causal framework in this study. Although *Other*-class is “useless” in traditional NER, we point out that in CL-NER, *Other*-class samples are naturally reserved samples from old classes, and we can distillate the causal effects from them to preserve old knowledge. Moreover, we apply curriculum learning to alleviate the noisy label problem to enhance the distillation of $Effect_O$. We further introduce a self-adaptive weight for dynamically balancing causal effects between new entity

types and *Other*-class. Experimental results show that the proposed causal framework not only outperforms state-of-the-art methods but also can be combined with other methods to further improve performances.

Ethics Statement

For the consideration of ethical concerns, we would make description as follows: (1) We conduct all experiments on existing datasets, which are derived from public scientific researches. (2) We describe the statistics of the datasets and the hyper-parameter settings of our method. Our analysis is consistent with the experimental results. (3) Our work does not involve sensitive data or sensitive tasks. (4) We will open source our code for reproducibility.

Limitations

Although the proposed method alleviates the catastrophic forgetting problem to some extent, its performances are still unsatisfactory when more CL steps are performed. Additionally, calculating causal effects from *Other*-class depends on the old model predictions, resulting in errors propagating to the following CL steps. Moreover, the proposed method requires more computation and a longer training time since the predictions of *matched* samples are calculated.

Acknowledgements

This paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 62272173, 61872148), the Natural Science Foundation of Guangdong Province (Grant Nos. 2022A1515010179, 2019A1515010768). We thank Xichen Shang, Zipeng Chen for helpful discussions on the manuscript.

References

- Naomi Altman and Martin Krzywinski. 2018. The curse (s) of dimensionality. *Nat Methods*, 15(6):399–400.
- Eric Arazo, Diego Ortego, Paul Albert, Noel O’Connor, and Kevin McGuinness. 2019. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, pages 312–321. PMLR.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

- Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca Passonneau, and Rui Zhang. 2022. CONTaiNER: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland. Association for Computational Linguistics.
- Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*, 2(6).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. 2020. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *ECCV 2020-16th European Conference on Computer Vision*, volume 12365, pages 86–102. Springer.
- Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. 2016. *Causal inference in statistics: A primer*. John Wiley & Sons.
- Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.
- Sheng Guo, Weilin Huang, Haozhi Zhang, Chenfan Zhuang, Dengke Dong, Matthew R Scott, and Dinglong Huang. 2018. Curriculumnet: Weakly supervised learning from large-scale web images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).
- Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839.
- Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.
- Xinting Hu, Kaihua Tang, Chunyan Miao, Xian-Sheng Hua, and Hanwang Zhang. 2021. Distilling causal effect of data in class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3957–3966.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.
- Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. Dice loss for data-imbalanced nlp tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 465–476.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.
- Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.
- 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*, pages 13570–13577.
- Shawn N Murphy, Griffin Weber, Michael Mendis, Vivian Gainer, Henry C Chueh, Susanne Churchill, and Isaac Kohane. 2010. Serving the enterprise and beyond with informatics for integrating biology and the bedside (i2b2). *Journal of the American Medical Informatics Association*, 17(2):124–130.
- Guoshun Nan, Jiaqi Zeng, Rui Qiao, Zhijiang Guo, and Wei Lu. 2021. Uncovering main causalities for long-tailed information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9683–9695.
- German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.
- Judea Pearl. 2009. *Causality*. Cambridge university press.

Judea Pearl. 2014. Interpretation and identification of causal mediation. *Psychological methods*, 19(4):459.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

Anthony Robins. 1995. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION’05)-Volume 1-Volume 01*, pages 29–36.

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

Bernhard Schölkopf. 2022. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pages 765–804.

Kaihua Tang, Jianqiang Huang, and Hanwang Zhang. 2020. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems*, 33:1513–1524.

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Dong Zhang, Hanwang Zhang, Jinhui Tang, Xian-Sheng Hua, and Qianru Sun. 2020. Causal intervention for weakly-supervised semantic segmentation. *Advances in Neural Information Processing Systems*, 33:655–666.

Wenkai Zhang, Hongyu Lin, Xianpei Han, and Le Sun. 2021. De-biasing distantly supervised named entity recognition via causal intervention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4803–4813.

Qingfu Zhu, Weinan Zhang, Ting Liu, and William Yang Wang. 2020. Counterfactual off-policy training for neural dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3438–3448.

A Forgetting in CL-NER

To identify the cause of forgetting, we consider the differences in prediction Y when the old data S exists or not. For each CL step, the effect of old data S can be calculate as:

$$Effect_S = P(\hat{Y} = \hat{y}|do(S = s)) \quad (8)$$

$$\begin{aligned} & - P(\hat{Y} = \hat{y}|do(S = 0)) \\ & = P(\hat{Y} = \hat{y}|S = s) - P(\hat{Y} = \hat{y}|S = 0) \end{aligned} \quad (9)$$

$$= P(\hat{Y} = \hat{y}) - P(\hat{Y} = \hat{y}) \quad (10)$$

$$= 0, \quad (11)$$

where $do(\cdot)$ is the *causal intervention* (Pearl, 2014, 2009) representing that assigning a certain value to a variable without considering all parent nodes (causes). In the first equation, $do(S = 0)$ represents null intervention, *i.e.*, setting old data to null. In the second equation, $P(\hat{Y} = \hat{y}|do(S) = P(\hat{Y} = \hat{y}|S))$ due to the fact that S has no parent nodes. In the third equation, $P(\hat{Y} = \hat{y}|S) = P(\hat{Y} = \hat{y})$ since all causal paths from S to Y are blocked by the collider X_0 . From Eq.(11), we find that the missing old data effect causes the forgetting. We neglect the effect of initial parameters adopted from the old model since it will be exponentially decayed towards 0 during learning (Kirkpatrick et al., 2017).

B Greedy Sampling Algorithm for CL-NER

In real-world scenarios, the new data should focus on the new entity types, *i.e.*, most sentences contain the tokens from new entity types. Suppose we randomly partition a dataset for CL-NER as in Monaikul et al. (2021). In that case, each slice contains a large number of sentences whose tokens all belong to *Other*-class, resulting in that models tend to bias to *Other*-class when inference. A straightforward solution is to filter out all sentences with only *Other*-class tokens. However, it brings a new problem: the slices’ sizes are imbalanced.

To address this problem, we propose a sampling algorithm for partitioning a dataset in CL-NER (Algorithm 1). Simply put, we allocate the sentence containing low-frequency entity types to the corresponding slice in priority until the slice contains the required number of sentences. If a sentence contains no entity types or the corresponding slices are full, we randomly allocate the sentence to an incomplete slice. In this way, we partition the dataset into

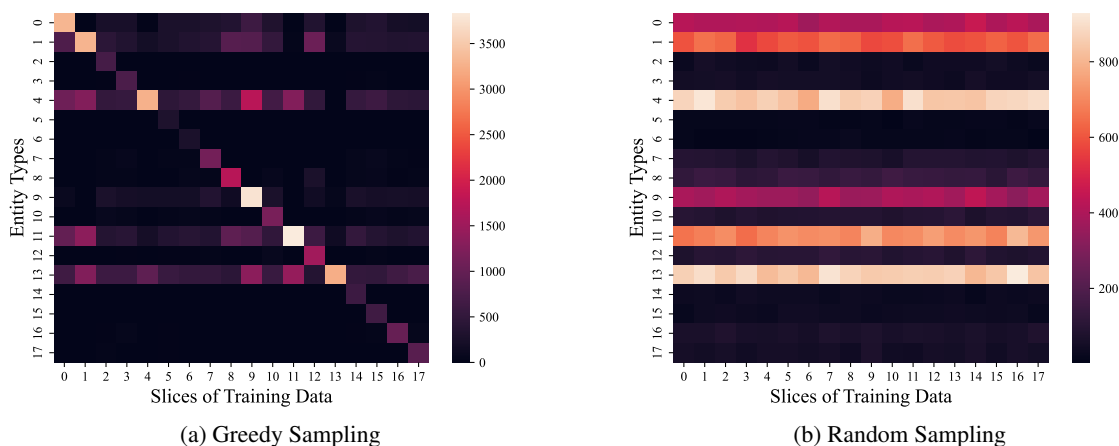


Figure 7: Comparison of the greedy sampling and random sampling on OntoNotes5. Each slice contains one entity types to learn.

slices with balanced sizes, and each slice mainly contains the entity types to learn.

For comparing Algorithm 1 and the random sampling as in Monaikul et al. (2021), we provide the label distributions in each slices of training data in Figure 7. Figure 7 shows that the greedy sampling generates more realistic datasets for CL-NER. When we use the randomly partitioned dataset for training in the setting FG-1-PG-1, the micro-f1 score of our method is 16.12, 17.43, and 12.75 (%) on OntoNotes5, I2B2, and CoNLL2003, respectively, indicating that the number of entities in each slice is inadequate for learning a NER model. Therefore, the greedy sampling alleviates the need for data in CL-NER.

C Baselines Introductions and Settings

The introductions about the baselines in experiments and their experimental settings are as follows. Note that we do not apply any reserved samples from old classes in LUCIR and PODNet for a fair comparison since our method requires no old data.

- Self-Training (ST) (Rosenberg et al., 2005; De Lange et al., 2019): ST first utilizes the old model to annotate the *Other*-class tokens with old entity types. Then, the new model is trained on new data with annotations of all entity types. Finally, the cross-entropy loss on all entity types is minimized.
- ExtendNER (Monaikul et al., 2021): ExtendNER has a similar idea to ST, except that the old model provides the soft label (*i.e.*,

probability distribution) of *Other*-class tokens. Specifically, the cross-entropy loss is computed for entity types’ tokens, and KL divergence loss is computed for *Other*-class tokens. During training, the sum of cross-entropy loss and KL divergence loss is minimized. Following Monaikul et al. (2021), we set the temperature of the teacher (old model) and student model (new model) to 1 and 2, respectively.

- LUCIR (Hou et al., 2019): LUCIR develops a framework for incrementally learning a unified classifier for the continual image classification tasks. The total loss consists of three terms: (1) the cross-entropy loss on the new classes samples; (2) the distillation loss on the features extracted by the old model and those by the new one; (3) the margin-ranking loss on the reserved samples for old classes. In our experiments, we compute the cross-entropy loss for new entity types, the distillation loss for all entity types, and the margin-ranking loss for *Other*-class samples instead of the reserved samples. Following (Hou et al., 2019), λ_{base} (*i.e.*, loss weight for the distillation loss) is set to 50, K (*i.e.*, the top- K new class embeddings are chosen for the margin-ranking loss) is set to 1 and m (*i.e.*, the threshold of margin ranking loss) is set to 0.5 for all the tasks.
- PODNet (Douillard et al., 2020): PODNet has a similar idea to LUCIR to combat the catastrophic forgetting in continual learning

Algorithm 1: Greedy Sampling Algorithm for CL-NER

Input: $\mathcal{D} = \{(s_i, t_i)\}_{i=1}^n$: a training set contains n sentences s and their label sequences t ;

Require: G : the number of slices; \mathcal{E} : the entity set; \mathcal{E}_j : the entity set in the j -th slice;

Output: G slices of datasets:
 $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \dots, \mathcal{D}^{(G)}$

- 1 Initialize $\mathcal{D}^{(1)}, \dots, \mathcal{D}^{(G)} = \{\}, \dots, \{\}$;
- 2 Initialize $cnt_1, \dots, cnt_G = 0, \dots, 0$;
// Calculate the number of sentences to allocate in each slice
- 3 **for** j in range($1, G+1$) **do**
- 4 | $n_j = n * |\mathcal{E}_j| / |\mathcal{E}|$
- 5 **end**
- 6 Sort \mathcal{E} in ascending order based on the frequency in \mathcal{D} ;
- 7 **for** (s_i, t_i) in \mathcal{D} **do**
- 8 | $is_select = \text{False}$;
// Allocate low frequency entity types in priority
- 9 | **for** e in \mathcal{E} **do**
- 10 | | **if not** e in t_i **then**
- 11 | | | continue;
- 12 | | **end**
- 13 | | j is index of \mathcal{E}_j s.t. e belongs to \mathcal{E}_j ;
- 14 | | **if** $cnt_j < n_j$ **then**
- 15 | | | $\mathcal{D}^{(j)} = \mathcal{D}^{(j)} \cup (s_i, t_i)$;
- 16 | | | $cnt_j += 1$;
- 17 | | | $is_select = \text{True}$;
- 18 | | | break;
- 19 | | **end**
- 20 | **end**
- 21 | | **if not** is_select **then**
- 22 | | | randomly choice j s.t. $cnt_j < n_j$;
- 23 | | | $\mathcal{D}^{(j)} = \mathcal{D}^{(j)} \cup (s_i, t_i)$;
- 24 | | | $cnt_j += 1$;
- 25 | | **end**
- 26 **end**
- 27 **return** $\mathcal{D}^{(1)}, \mathcal{D}^{(2)}, \dots, \mathcal{D}^{(G)}$;

for image classification. The total loss consists of the classification loss and distillation loss. To compute the distillation loss, PodNet

Table 5: The statistics and the entity sequence for each dataset

	# Class	# Sent	# Entity	Entity sequence (Alphabetical Order)
CoNLL2003	4	21k	35k	LOCATION, MISC, ORGANISATION, PERSON
I2B2	16	141k	29k	AGE, CITY, COUNTRY, DATE, DOCTOR, HOSPITAL, IDNUM, MEDICALRECORD, ORGANIZATION, PATIENT, PHONE, PROFESSION, STATE, STREET, USERNAME, ZIP
OntoNotes5	18	77k	104k	CARDINAL, DATE, EVENT, FAC, GPE, LANGUAGE, LAW, LOC, MONEY, NORP, ORDINAL, ORG, PERCENT, PERSON, PRODUCT, QUANTITY, TIME, WORK_OF_ART

constrains the output of each intermediate convolutional layer while LUCIR only considers the final feature embeddings. For classification, PODNet used NCA loss instead of the cross-entropy loss. In our experiments, we constrain the output of each intermediate stage of BERT as PODNet constrains each stage of a ResNet. Following (Douillard et al., 2020), we set the λ_c (*i.e.*, loss weight for the POD-spatial loss) to 3 and λ_f (*i.e.*, loss weight for the POD-flat loss) to 1.

D Training Details

The models were implemented in Pytorch (Paszke et al., 2019) on top of the BERT Huggingface implementation (Wolf et al., 2019). We use the default hyper-parameters in BERT: hidden dimensions are 768, and the max sequence length is 512. Following Hu et al. (2021), we normalize the feature vector output by BERT and compute the cosine similarities between the feature vector and class representations for predictions.

We use the BIO tagging schema for all three datasets. For CoNLL2003, we train the model for ten epochs in each CL step. For OntoNotes5 and I2B2, we train the model for 20 epochs when PG=2, and 10 epochs when PG=1. The batch size is set as 8, and the learning rate is set as 4e-4. The experiments are run on GeForce RTX 2080 Ti GPU. Each experiment is repeated 5 times.

E T-SNE Visualizations

To deepen the understanding of the forgetting in CL-NER, we visualize the feature representations from BERT in Figure 8. Results show that our method preserves more old knowledge and learns better feature representations than ExtendNER.

F Additional Experimental Results

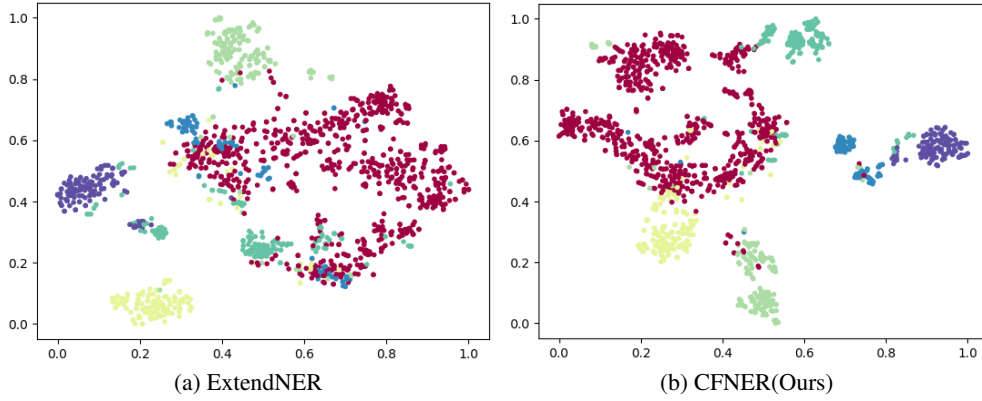
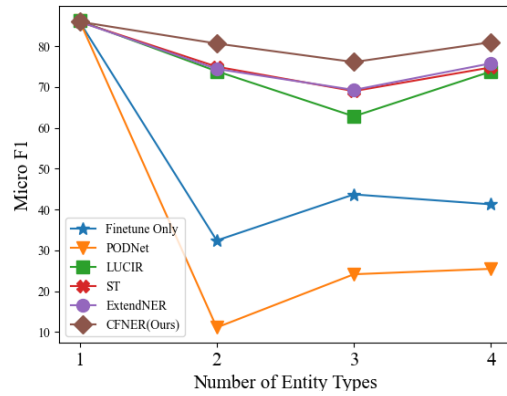


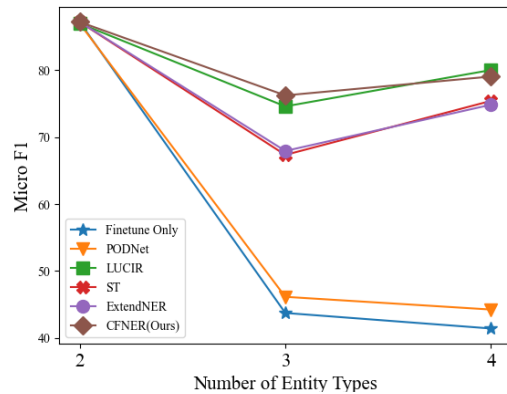
Figure 8: The T-SNE visualization of the feature representations from BERT. The model is trained on OntoNotes5 in the setting FG-1-PG-1. We randomly select six classes for a demonstration.

Table 6: Comparisons with state-of-the-art methods on CoNLL2003. The average results as well as standard derivations are provided. *Mi-F1*: micro-F1; *Ma-F1*: macro-F1. The best F1 results are bold.

Dataset	Method	FG-1-PG-1		FG-2-PG-1	
		Mi-F1	Ma-F1	Mi-F1	Ma-F1
CoNLL2003	Finetune Only	50.84 ±0.10	40.64 ±0.16	57.45 ±0.05	43.58 ±0.18
	PODNet	36.74 ±0.52	29.43 ±0.28	59.12 ±0.54	58.39 ±0.99
	LUCIR	74.15 ±0.43	70.48 ±0.66	80.53 ±0.31	77.33 ±0.31
	ST	76.17 ±0.91	72.88 ±1.12	76.65 ±0.24	66.72 ±0.11
	ExtendNER	76.36 ±0.98	73.04 ±1.8	76.66 ±0.66	66.36 ±0.64
	CFNER(Ours)	80.91 ±0.29	79.11 ±0.50	80.83 ±0.36	75.20 ±0.32



(a) CoNLL2003 (FG-1-PG-1)



(b) CoNLL2003 (FG-2-PG-1)

Figure 9: Comparison of the step-wise micro-f1 score on CoNLL2003 (4 entity types).