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).

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-
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). This process is illustrated in Figure 1 to demonstrate Other-class in CL-NER. 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 notes 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 $\text{sprinkler} \rightarrow \text{pavement} \leftarrow \text{weather}$ 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.
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.

4.1 Problem Formulation

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

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 distill 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.
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 $X_i^E$ and $X_i^O$ in the old feature space. In this way, we build two causal paths from old data $S$ to new predictions $\tilde{Y}$. 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:

$$Effect = Effect_E + Effect_O$$  \hspace{1cm} \text{(1)}$$

$$-\sum_i CE(Y_i, \hat{Y}_i) - \sum_j KL(Y_j, \tilde{Y}_j),$$  \hspace{1cm} \text{(2)}$$

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

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, $Y$ 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 follows:

$$\tilde{Y}_i = W_i \hat{Y}_i + \sum_{k=1}^K W_{ik} \tilde{Y}_k$$  \hspace{1cm} \text{(3)}$$

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

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

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 \( \text{Effect}_O \) in Eq.(1) as follow:

\[
\text{Effect}_O = -\Sigma_j KL(\hat{Y}_j) - \Sigma_n KL(\hat{Y}_n)
\]

\[
\text{s.t. } D_j \in D^O, \quad D_n \in D^{UO},
\]

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 \( \text{Effect}_O \)

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 \( \text{Effect}_O \). 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 \( \delta \) \((\delta \in [0, 1])\) to encourage the model to learn first from clean Other-class samples and then noisier ones. Specifically, when calculating \( \text{Effect}_O \), we only select Defined-Other-Class tokens whose predictive confidences are larger than \( \delta \) for distilling colliding effects while others are for knowledge distillation. The value of \( \delta \) changes along with the training process and the value of \( \delta \) in the \( i \)-th epoch is calculated as follow:

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

where \( m, \delta_1 \) and \( \delta_m \) are the predefined hyperparameters and \( \delta_m \) should be smaller than \( \delta_1 \).

### 4.5 Balancing \( \text{Effect}_E \) and \( \text{Effect}_O \)

Figure 5 shows that the total causal effect \( \text{Effect} \) consists of \( \text{Effect}_E \) and \( \text{Effect}_O \), where \( \text{Effect}_E \) is for learning new entities while \( \text{Effect}_O \) 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 \( \text{Effect}_E \) and \( \text{Effect}_O \):

\[
\lambda = \lambda_{\text{base}} \sqrt{C_O/C_N}
\]

where \( \lambda_{\text{base}} \) 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:

\[
\max \text{Effect} = \text{Effect}_E + \lambda \ast \text{Effect}_O
\]

### 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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>FG-1-PG-1</th>
<th>FG-2-PG-2</th>
<th>FG-8-PG-1</th>
<th>FG-8-PG-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mi-F1</td>
<td>Ma-F1</td>
<td>Mi-F1</td>
<td>Ma-F1</td>
<td>Mi-F1</td>
</tr>
<tr>
<td>I2B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>17.43</td>
<td>13.81</td>
<td>28.57</td>
<td>21.43</td>
<td>20.83</td>
</tr>
<tr>
<td></td>
<td>±0.54</td>
<td>±1.14</td>
<td>±0.26</td>
<td>±0.41</td>
<td>±1.78</td>
</tr>
<tr>
<td></td>
<td>±0.35</td>
<td>±1.03</td>
<td>±2.65</td>
<td>±1.76</td>
<td>±1.38</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>43.86</td>
<td>31.31</td>
<td>64.32</td>
<td>43.53</td>
<td>57.86</td>
</tr>
<tr>
<td></td>
<td>±2.43</td>
<td>±1.62</td>
<td>±0.76</td>
<td>±0.59</td>
<td>±0.87</td>
</tr>
<tr>
<td></td>
<td>±0.74</td>
<td>±1.31</td>
<td>±4.78</td>
<td>±3.13</td>
<td>±1.35</td>
</tr>
<tr>
<td>ST</td>
<td>31.98</td>
<td>14.76</td>
<td>55.44</td>
<td>33.38</td>
<td>49.51</td>
</tr>
<tr>
<td></td>
<td>±2.12</td>
<td>±1.31</td>
<td>±4.78</td>
<td>±3.13</td>
<td>±1.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>42.85</td>
<td>24.05</td>
<td>57.01</td>
<td>35.29</td>
<td>43.95</td>
</tr>
<tr>
<td></td>
<td>±2.86</td>
<td>±1.35</td>
<td>±4.14</td>
<td>±1.38</td>
<td>±2.01</td>
</tr>
<tr>
<td>ExtenderNER</td>
<td>62.73</td>
<td>36.26</td>
<td>71.98</td>
<td>49.09</td>
<td>59.79</td>
</tr>
<tr>
<td></td>
<td>±3.62</td>
<td>±2.24</td>
<td>±0.50</td>
<td>±1.38</td>
<td>±1.70</td>
</tr>
<tr>
<td>CFNER(Ours)</td>
<td>15.27</td>
<td>10.85</td>
<td>25.85</td>
<td>20.55</td>
<td>17.63</td>
</tr>
<tr>
<td></td>
<td>±0.26</td>
<td>±1.11</td>
<td>±0.11</td>
<td>±0.24</td>
<td>±0.57</td>
</tr>
<tr>
<td>OntoNotes5</td>
<td>9.06</td>
<td>8.36</td>
<td>34.67</td>
<td>24.62</td>
<td>29.00</td>
</tr>
<tr>
<td></td>
<td>±0.96</td>
<td>±0.57</td>
<td>±0.85</td>
<td>±0.86</td>
<td>±0.91</td>
</tr>
<tr>
<td></td>
<td>±0.56</td>
<td>±0.57</td>
<td>±1.08</td>
<td>±0.86</td>
<td>±0.91</td>
</tr>
<tr>
<td></td>
<td>±0.96</td>
<td>±0.14</td>
<td>±1.79</td>
<td>±1.11</td>
<td>±0.46</td>
</tr>
<tr>
<td></td>
<td>28.18</td>
<td>21.11</td>
<td>64.32</td>
<td>43.53</td>
<td>66.46</td>
</tr>
<tr>
<td></td>
<td>±1.15</td>
<td>±0.90</td>
<td>±1.08</td>
<td>±0.86</td>
<td>±0.91</td>
</tr>
<tr>
<td></td>
<td>±0.79</td>
<td>±1.06</td>
<td>±1.67</td>
<td>±1.66</td>
<td>±0.66</td>
</tr>
<tr>
<td></td>
<td>50.71</td>
<td>33.24</td>
<td>68.93</td>
<td>50.63</td>
<td>73.59</td>
</tr>
<tr>
<td></td>
<td>±0.86</td>
<td>±0.34</td>
<td>±1.49</td>
<td>±1.09</td>
<td>±0.77</td>
</tr>
<tr>
<td>ST</td>
<td>50.53</td>
<td>32.84</td>
<td>67.61</td>
<td>49.26</td>
<td>73.12</td>
</tr>
<tr>
<td></td>
<td>±0.80</td>
<td>±0.34</td>
<td>±1.49</td>
<td>±1.09</td>
<td>±0.77</td>
</tr>
<tr>
<td></td>
<td>±0.57</td>
<td>±1.10</td>
<td>±0.48</td>
<td>±0.69</td>
<td>±0.58</td>
</tr>
<tr>
<td>ExtendNER</td>
<td>58.94</td>
<td>42.22</td>
<td>72.59</td>
<td>55.96</td>
<td>78.92</td>
</tr>
<tr>
<td>CFNER(Ours)</td>
<td>±0.57</td>
<td>±1.10</td>
<td>±0.48</td>
<td>±0.69</td>
<td>±0.58</td>
</tr>
</tbody>
</table>

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 others 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 validation 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: Extend-NER (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$_{iO}$ and w/o Effect$_E$, respectively. Specifically, in the w/o Effect$_{iO}$ 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$_{iO}$ & Effect$_E$). The results show that both Effect$_{iO}$ 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 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>I2B2</th>
<th>OntoNotes5</th>
<th>CoNLL2003</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mi-F1</td>
<td>Ma-F1</td>
<td>Mi-F1</td>
</tr>
<tr>
<td>CFNER(Our)</td>
<td>62.73</td>
<td>36.26</td>
<td>58.94</td>
</tr>
<tr>
<td>w/o AW</td>
<td>61.65</td>
<td>35.86</td>
<td>57.63</td>
</tr>
<tr>
<td>w/o CuL</td>
<td>61.21</td>
<td>34.79</td>
<td>57.85</td>
</tr>
<tr>
<td>w/o AW &amp; CuL</td>
<td>60.78</td>
<td>33.15</td>
<td>56.83</td>
</tr>
<tr>
<td>w/o Effect$_{iO}$</td>
<td>59.68</td>
<td>30.86</td>
<td>53.09</td>
</tr>
<tr>
<td>w/o Effect$_E$</td>
<td>53.62</td>
<td>28.73</td>
<td>54.08</td>
</tr>
<tr>
<td>w/o Effect$_{iO}$ &amp; Effect$_E$</td>
<td>42.85</td>
<td>24.05</td>
<td>50.53</td>
</tr>
</tbody>
</table>

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 $\lambda_{base}$ and the initial value of confidence threshold $\delta_1$. 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$_{iO}$ 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 $\lambda_{base}$ 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 hyperparameters 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.

<table>
<thead>
<tr>
<th>Methods</th>
<th>I2B2</th>
<th>OntoNotes5</th>
<th>CoNLL2003</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFNER (Ours)</td>
<td>62.73</td>
<td>36.26</td>
<td>58.94</td>
</tr>
<tr>
<td>LUCIR</td>
<td>43.86</td>
<td>31.31</td>
<td>28.18</td>
</tr>
<tr>
<td>LUCIR+CF</td>
<td>66.27</td>
<td>38.52</td>
<td>62.03</td>
</tr>
<tr>
<td>ST</td>
<td>31.98</td>
<td>14.76</td>
<td>50.71</td>
</tr>
<tr>
<td>ST+CF</td>
<td>61.41</td>
<td>33.43</td>
<td>57.06</td>
</tr>
</tbody>
</table>

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 CL-NER. 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 distill 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 hyperparameter 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


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 calculated as:

$$\text{Effect}_S = P(\hat{Y} = \hat{y}|\text{do}(S = s)) - P(\hat{Y} = \hat{y}|S = 0)$$

$$= P(\hat{Y} = \hat{y}|S = s) - P(\hat{Y} = \hat{y}|S = 0)$$

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

where $\text{do}(\cdot)$ 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, $\text{do}(S = 0)$ represents null intervention, i.e., setting old data to null. In the second equation, $P(\hat{Y} = \hat{y}|\text{do}(S) = P(\hat{Y} = \hat{y}|S)$ 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).

### 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 $\text{Other}$-class, resulting in that models tend to bias to $\text{Other}$-class when inference. A straightforward solution is to filter out all sentences with only $\text{Other}$-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

\[A \quad \text{Forgetting in CL-NER}\]

\[B \quad \text{Greedy Sampling Algorithm for CL-NER}\]
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), $\lambda_{base}$ (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: \( D = \{(s_i, t_i)\}_{i=1}^n \): a training set contains \( n \) sentences \( s \) and their label sequences \( t \);

Require: \( G \): the number of slices; \( E \): the entity set; \( E_j \): the entity set in the \( j \)-th slice;

Output: \( G \) slices of datasets: 
\[
D^{(1)}, D^{(2)}, \ldots, D^{(G)}
\]

1. Initialize \( D^{(1)}, \ldots, D^{(G)} = \{\}, \ldots, \{\} \);
2. Initialize \( cnt_1, \ldots, cnt_G = 0, \ldots, 0 \);

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

for image classification. The total loss consists of the classification loss and distillation loss. To compute the distillation loss, PodNet 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 \( \lambda_c \) (i.e., loss weight for the POD-spatial loss) to 3 and \( \lambda_f \) (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.

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

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