GenCNER: A Generative Framework for Continual Named Entity Recognition

Anonymous ACL submission

Abstract

Traditional named entity recognition aims to identify text mentions into pre-defined entity types. Continual Named Entity Recognition (CNER) is introduced since entity categories are continuously increasing in various real-world scenarios. However, existing continual learning (CL) methods for NER face challenges of catastrophic forgetting and semantic shift of non-entity type. In this paper, we propose GenCNER, a simple but effective Generative framwork for CNER to mitigate the above drawbacks. Specifically, we skillfully convert the CNER task into sustained entity triplet label generation problem and utilize a powerful pre-trained seq2seq model to solve it. Additionally, we design a type-specific confidence-based pseudo labeling strategy along with knowledge distillation (KD) to preserve learned knowledge and alleviate the impact of label noise at the triplet level. Experimental results on two benchmark datasets show that our framework outperforms previous state-of-the-art methods in multiple CNER settings, and achieves the smallest gap compared with non-CL results[1].

1 Introduction

Named Entity Recognition (NER) is one fundamental task in NLP fields due to its wide application in entity linking (Le and Titov, 2018; Ganea and Hofmann, 2017), relation extraction (Hu et al., 2021; Liu et al., 2022) and knowledge graph (Yang et al., 2022). Conventionally, NER involves recognizing text mentions into fixed pre-defined entity categories (e.g., "Person", "Organization") and the NER model only experiences one-time complete training process. While in real world, new entity types continue to emerge over time, which requires the model to perform incremental learning rather than training from scratch. As an example, voice assistants like Siri and Xiao Ai are frequently expected to extract new entity types for accurate understanding of user intents (Xia et al., 2022; Zhang et al., 2023). Thus the technique for Continual Named Entity Recognition (CNER) has attracted sustained attention to learn new entity types incrementally without forgetting the old ones.

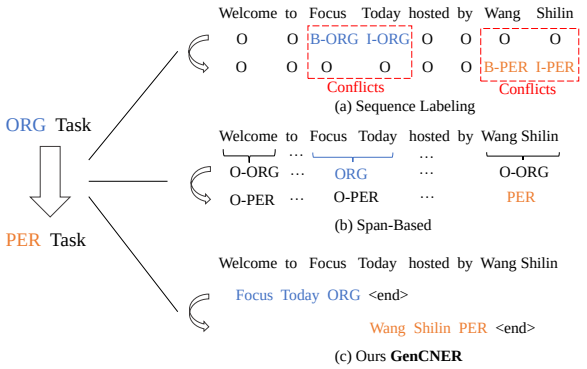

Figure 1: Changes of current ground-truth labels for sequence labeling, span-based and ours GenCNER method as CNER tasks increase. The red dashed box indicates conflicts between training targets, which leads to the semantic shift problem of the non-entity.

Some efforts have been devoted to solve CNER tasks effectively. Existing work could be mainly divided into two strategies: sequence labeling and span-based models. The sequence labeling methods (Monaikul et al., 2021; Xia et al., 2022; Zheng et al., 2022) assign each token one tag to indicate the entity boundary and type, and employ knowledge distillation (KD) to maintain old knowledge with entity type increasing. Nevertheless, such mechanism often encounters the semantic shift of non-entity type (Zhang et al., 2023). In conventional NER labeling paradigm, tokens marked as non-entity do not belong to any entity type. While in CNER setting, the non-entity mark only indicates the corresponding tokens do not belong to current entity type. Consequently, the non-entity type will inevitably encompass the context token, previously learned entity types and future ones not yet encountered. As illustrated in Figure 1, be-

[1]The code will be released for future research.

sides true non-entity tokens, the non-entity mark "O" also covers future entity type "PER" in the current "ORG" task. The problem of semantic shift leads to inconsistent training objectives and exacerbates catastrophic forgetting.

The span-based methods (Zhang and Chen, 2023; Chen and He, 2023) formulate CNER as a continuous binary classification task of the candidate entity spans and leverage knowledge distillation (KD) to retain memory. By changing from token-level labeling to span-level binary classification, the semantic shift of non-entity type could be avoided. As depicted in Figure 1, the non-entity mark regards entity span as non-(certain)-entity ("O-ORG"), which has no conflict with future entity type ("PER"). However, the basic framework proposed by Zhang and Chen (2023) ignores label noise brought by previous teacher model. To address catastrophic forgetting and label noise problems, Chen and He (2023) introduces Reinforcement Learning to span-based Knowledge Distillation for further improvement, but increases model complexity and computation costs.

The success of generative pre-trained language models (BART (Lewis et al., 2020), T5 (Raffel et al., 2020), etc.) has recently aroused interest in leveraging generative methods to solve information extraction tasks. Inspired by this, we propose **GenCNER**, a **Gen**erative framework with confidence-based pseudo labeling and knowledge distillation (KD) to reduce catastrophic forgetting and label noise. To be specific, we formulate continual NER task as sustained entity triplet label generation problem and employ the pre-trained seq2seq model BART with pointer network to generate target triplet sequence. In order to retain previously learned entity types and mitigate label noise, we first develop a confidence-based pseudo labeling strategy which removes pseudo entity triplets (predicted by teacher model) with lower confidence than type-specific thresholds, then empirically adopt knowledge distillation (KD) to transfer old knowledge to student model more accurately.

Compared with sequence labeling methods for CNER, our generative framework GenCNER well avoids the problem of semantic shift since it has no need of non-entity marks in the generative background. As shown in Figure 1, the entity triplet is defined as "(start token, end token, entity type)" and the target sequence will continuously append triples of new entity types as tasks increase. We can observe that the generative method updates learning targets for CNER task in a more natural and reasonable manner. Furthermore, GenCNER even supports both nested and discontinuous entity structures while other methods cannot or struggle. Experimental results on OntoNotes and Few-NERD show that GenCNER achieves the highest F1 score at each CL step as well as the smallest performance gap to non-CL settings. The main contributions of this work can be summarized as follows:

- We propose an effective generative framework to tackle CNER task, which innovatively transforms continual named entity recognition into consecutive entity triplet generation process. To the best of our knowledge, we are the first to introduce the generative mechanism to explorations of CNER tasks.

- We develop a confidence-based pseudo labeling strategy with type-specific thresholds filtering to preserve high-quality entity triplets generated by old model for knowledge distillation (KD), weakening the influence of catastrophic forgetting and label noise.

- We conduct adequate experiments on two public NER datasets in class incremental settings. Experimental results show that our proposed method outperforms strong baselines consistenly on each dataset and achieves new state-of-the-art in different CL settings.

2 Related Work

Named Entity Recognition aims to locate text mentions with specific meanings and classify them into predefined entity categories (Li et al., 2022a). Recent works for various NER subtasks could be mainly divided into sequence labeling, span-based, hypergraph-based and seq2seq methods (Shen et al., 2021; Li et al., 2022b). The sequence labeling methods (Metke-Jimenez and Karimi, 2015; Straková et al., 2019) annotate each token with one tag indicating the entity boundary and type. Common tagging schemas include BIO and BIOES, where "B" represents the start of an entity, "O" stands for the context token not belonging to any entity. The span-based methods (Yu et al., 2020; Tan et al., 2020) consider NER as a classification task of the candidate entity spans extracted from input sentence. Hypergraph-based methods (Katiyar and Cardie, 2018) construct hypergraphs based on the entity structure and introduce graph neural network. Most recently, the seq2seq framework

which converts entity extraction into sequence generation has been proposed to solve multi-type NER subtasks uniformly (Yan et al., 2021), achieving surprising performance.

Continual Learning NER solves continuous tasks where training targets of new entity types emerge as a stream. Existing works treat Continual NER as the class-incremental problem where the training data only contains labels of current entity type(s). Early explorations like AddNER and ExtendNER (Monaikul et al., 2021) employed the basic sequence labeling framework to tag tokens and knowledge distillation to prevent model from forgetting old entity types. To reduce the dependency of type co-occurrence during KD, Xia et al. (2022) proposed L&R, which provides ExtendNER with synthetic samples of old types for further distillation. Considering the semantic shift of non-entity type, Zheng et al. (2022) designed a causal framework to retrieve the causality from both new entity types and Other-Class, Zhang et al. (2023) developed a confidence-based pseudo-labeling for the non-entity type and Ma et al. (2023) proposed to learn discriminative representations for entity types and "O" labels. On the other hand, SpanKL (Zhang and Chen, 2023) and SKDNER (Chen and He, 2023) introduced span-based classification to CNER, which avoided the problem of semantic shift and achieved satisfactory results.

Seq2seq Framework The seq2seq model with encoder-decoder structure has been successfully adopted in various NLP tasks. It was first proposed by Cho et al. (2014) in order to solve machine translation tasks. Bahdanau et al. (2014) applied the attention mechanism to the decoder and obtained better translation performance. On this basis, the Pointer Network (Vinyals et al., 2015) and CopyNet (Gu et al., 2016) were proposed to get the probability distribution for input sentences. With the increase of pre-trained language model, several attempts have been made to pre-train a seq2seq model, including MASS (Song et al., 2019), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020). In this paper, we focus on BART since its simplicity and better performance.

3 Preliminary

3.1 Problem Formulation

We follow previous works (Monaikul et al., 2021) to consider CNER as the class-incremental problem. We train the model on a sequence of tasks $T_1, T_2, ..., T_l$, where each task $T_k (1 \leq k \leq l)$ has its own training dataset D_k annotated only for new entity types E_k. Note that entity types involved in different tasks are non-overlapping (i.e., $E_i \cap E_j = \emptyset$ when $i \neq j$). Meanwhile, sentences in each specific training dataset potentially also contain other entity types in previous or future steps, which are certainly not annotated in current task. At the first step, we train model M_1 on D_1 from scratch to identify entity types in E_1. At the k-th ($k > 1$) incremental step, our goal is to train a model M_k based on the training set D_k and previously learned model M_{k-1} to recognize entities of all types seen so far, represented as $\cup_{i=1}^{k} E_i$.

3.2 Learning Targets Construction

In order to apply the generative framework to handle CNER task, we empirically represent each target entity as the triplet containing start token, end token and entity type. Given an input training sentence of n tokens $X = [x_1, x_2, ..., x_n]$, the learning targets of different CNER tasks could be formulated as follows: At the first step, we expect to identify entities of types in E_1 and construct the target sequence $Y_1 = [s_1, e_1, t_1, s_2, e_2, t_2, ..., s_g, e_g, t_g]$ to train the model M_1, where g is the total number of entities, $s_i, e_i (1 \leq i \leq g)$ indicate the start and end token of the i-th entity respectively and $t_i \in E_1$ represents the entity type. At the k-th ($k > 1$) learning step with the previous model M_{k-1} available, we first adopt M_{k-1} to predict pseudo triplet sequence Y_k^p including all the previously learned entity types $\cup_{i=1}^{k-1} E_i$, then connect the ground truth of triplet sequence Y_k^g only containing E_k types at the end of Y_k^p to form the final target sequence $Y_k = [Y_k^p, Y_k^g]$ for model M_k training.

4 Method

The architecture of our framework is illustrated in Figure 2, consisting of two necessary components: Entity Triplet Sequence Generation (ETSG) and Confidence-based Pseudo Labeling (CBPL). In the first task of CNER, we train the model M_1 on D_1 with CE loss to make ETSG generate entity triplets of E_1 types. In other incremental tasks, we first use previous model to make a one-off prediction on training data, then prune the pseudo triplet sequence by confidence-based threshold filtering. We train the current model with KD loss and CE loss together to make ETSG capable of generating entity triplets of all seen types.

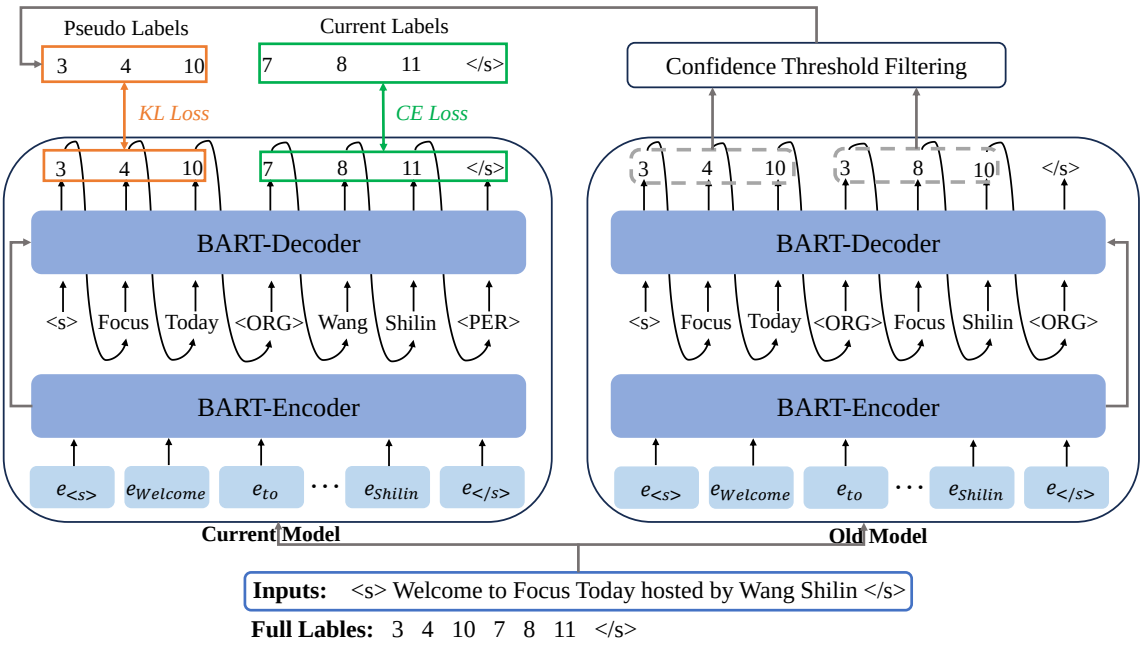

Figure 2: Overview of the proposed generative framework for Continual NER. Since the input sentence has 10 tokens, we conduct the 10 shift to the entity type index. Thus index 0-9 indicates boundary token of entities, and index larger than 9 represents different entity categories. $e_{<word>}$ denotes token embedding of BART.

4.1 Entity Triplet Sequence Generation

As mentioned above, we need to predict entity triplet sequence in each incremental task of CNER. Considering previous works (Yan et al., 2021; Yu et al., 2023) which effectively employ generative framework to solve the unified or multimodal NER tasks, we adopt the pre-trained seq2seq model BART with encoder-decoder structure to generate entity triplet sequence. Additionally, we introduce the idea of pointer network to BART decoder since the start and end tokens of each entity triplet both come from the input sentence. The BART encoder deals with the raw sentence $X = [x_1, x_2, ..., x_n]$ to get the contextual sentence-level representation $\mathbf{H}^e \in \mathbb{R}^{n \times d}$ as:

$$\mathbf{H}^e = \text{BartEncoder}(X), \tag{1}$$

where e represents the encoder output, n is the sentence length and d is the hidden dimension.

The BART decoder aims to generate boundary token or entity type index at each time step $P_t = P(y_t|H^e, Y_{<t})$. However, the previous triplet sequence $Y_{<t}$ cannot be put into the decoder module directly since it contains boundary token as well as entity type index. To tackle this problem, we continuously add new entity types as special tokens to the BART Tokenizer so that the sentence tokens and entity categories always share the same vocabulary. Then we develop an Index2Token mode to convert predicted indexes into practical tokens at each generative time step:

$$\hat{y}_t = \begin{cases} X_{y_t}, & if\ y_t \leq n \\ [\cup_{i=1}^{k} E_i]_{y_t - n}, & if\ y_t > n, \end{cases} \tag{2}$$

where $[\cup_{i=1}^{k} E_i]$ represents all the entity types learned so far in the k-th incremental CNER task.

After converting indexes into tokens in this way, we can put the encoder output and previous triplet sequence into the decoder network and obtain the last hidden state, which formulates as:

$$\mathbf{h}_t^d = \text{BartDecoder}(\mathbf{H}^e; \hat{Y}_{<t}), \tag{3}$$

where $\hat{Y}_{<t} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_{t-1}]$, representing the generated index triplet sequence for time step t.

Then we apply the pointer mechanism to generate index probability distribution P_t as:

$$\hat{\mathbf{H}}^e = \text{MLP}(\mathbf{H}^e), \tag{4}$$

$$[\cup_{i=1}^{k} \mathbf{E}_i]^d = \text{BartTokenEmbed}(\cup_{i=1}^{k} \mathbf{E}_i), \tag{5}$$

$$\mathbf{L}_t = [\hat{\mathbf{H}}^e \otimes \mathbf{h}_t^d; [\cup_{i=1}^{k} \mathbf{E}_i]^d \otimes \mathbf{h}_t^d], \tag{6}$$

$$P_t = \text{Softmax}(\mathbf{L}_t), \tag{7}$$

where the MLP module makes a linear transformation on the encoder output, BartTokenEmbed function means the token embeddings shared between encoder and decoder. Suppose the number of entity types in $\cup_{i=1}^{k} E_i$ is ek, then $\hat{\mathbf{H}}^e \in \mathbb{R}^{n \times d}$; $[\cup_{i=1}^{k} \mathbf{E}_i]^d \in \mathbb{R}^{ek \times d}$; $\mathbf{L}_t \in \mathbb{R}^{t \times (n+ek)}$; $\otimes$ means the dot product of multi-dimension vectors and $[\cdot;\cdot]$ concats the two vectors in the first dimension.

4.2 Confidence-based Knowledge Distillation

GenCNER uses knowledge distillation to avoid forgetting old types of entities while learning the new ones. At the k-th incremental task, we make a one-off prediction with the previously learned model M_{k-1} on the current training dataset for all the seen types $\cup_{i=1}^{k-1} E_i$. Next, we connect the pseudo triplet sequence generated by model M_{k-1} with the annotated triplet sequence of current entity type(s) E_k, to construct target triplet sequence for model M_k optimization. During KD process, the previous model M_{k-1} plays the role of teacher to distill old knowledge into the current student M_k.

It is worth noting that the pseudo triplet sequence possibly brings label noise which interferes the training of current model. In order to reduce model noise and improve distillation efficiency, we empirically design a confidence-based pseudo labeling mechanism for the filtering of poor predictions. Specifically, towards each training sentence in D_k, we first compute the probability of each token predicted by M_{k-1}. Since it is difficult to evaluate the quality of one single token, we split the probability sequence every three generative steps, corresponding to each predicted entity triplet. Then we obtain the minimum value of each probability triplet to stand for the generation quality of different entity triplets. Since different entity types have diverse learning difficulties, we define the confidence threshold in a type-specific way referring to previous semi-supervised works (Zhang et al., 2021; Wang et al., 2023).

$$\delta^{(c)} = \min(\delta, \text{median}\{\min(p(s), p(e), p(c)) | (s, e, c) \in S^{k-1}\}), \tag{8}$$

where δ is the type-specific hyper-parameter, s (or e) also denotes the start (or end) token of entity, $c \in \cup_{i=1}^{k-1} E_i$ represents one specific entity type and S^{k-1} denotes the whole entity triplet set predicted by M_{k-1}. Such a threshold ensures that we keep at least 50% pseudo triplets of each entity type to train a new model.

For noise minimization, entity triplets with token probability lower than the type-specific threshold are removed from predicted entity triplet sequence, forming the new pseudo label sequence as:

$$Y_k^{pru} = \{(s_i, e_i, t_i) \mid \min(p(s_i), p(e_i), p(t_i)) \geq \delta^{(t_i)}\}, \tag{9}$$

where $(s_i, e_i, t_i) \in Y_k^p$ denotes the i-th entity triplet generated by old model M_{k-1}.

The pruned pseudo triplet sequence Y_k^{pru} is used to compute the KL divergence loss of soft labels with the current model M_k, which formulates as:

$$\mathcal{L}_{KD} = \frac{1}{L} \sum_{t=1}^{L} P(y_t^{pru})(\log P(y_t^{pru}) - \log f_\theta(X, \hat{Y}_{<t})), \tag{10}$$

where L is the length of pruned pseudo triplet sequence, $y_t^{pru} \in Y_k^{pru}$ stands for the boundary token or entity type predicted by old model M_{k-1} at the t-th generative step, $P(y_t^{pru})$ means the corresponding probability distribution of y_t^{pru}, f_θ represents the current student model M_k.

Considering we expect the model to learn new entity types by extending entity triplet sequence, the cross entropy loss is computed as:

$$\mathcal{L}_{CE} = -\frac{1}{G} \sum_{t=L+1}^{L+G} P_{one-hot}(y_t^g) \log f_\theta(X, \hat{Y}_{<t}), \tag{11}$$

where G represents the length of current ground truth of triplet sequence, $y_t^g \in Y_k^g$ means the $(t - L)$-th value of current annotated triplet sequence and f_θ represents the current model M_k, as above.

Following previous continue learning methods for NER (Monaikul et al., 2021; Zhang and Chen, 2023), the final loss in the k-th incremental task $(k > 1)$ is the weighted sum as:

$$\mathcal{L} = \alpha \mathcal{L}_{KD} + \beta \mathcal{L}_{CE}, \tag{12}$$

where α and β are both the weights of different losses to balance the learning of past and present.

5 Experiments

We conduct extensive experiments on two public datasets to prove the effectiveness of our proposed model in recognizing entities for growing CNER tasks. Then the detailed analyses are given to show the strengths of each component.

5.1 Datasets

For the evaluation of proposed generative framework, we conduct main experiments on two public NER datasets: OntoNotes (Hovy et al., 2006) and Few-NERD (Ding et al., 2021). To imitate the class-incremental setting, we follow previous works (Monaikul et al., 2021; Xia et al., 2022) to split the above standard NER corpora into separate datasets used in a series of CL tasks. Compared to OntoNotes which only annotates one single entity type within each task, the fine-grained dataset Few-NERD allows learning multiple types per task. Some basic information of them is described below. **OntoNotes-5.0 English** is one flat NER dataset in the field of news, containing 18 en-

tity categories. We follow recent works (Zhang and Chen, 2023) to select 6 types to ensure sufficient training samples, including Organization, Person, GPE, Date, Cardinal and NORP. Each entity type is annotated only in the corresponding CL task. **Few-NERD** is first proposed for few-shot NER research, which consists of 8 coarse-grained and 66 fine-grained entity types listed in Appendix A. For simplicity, we employ the supervised full version of Few-NERD and construct a sequence of CL tasks via each coarse-grained type. Consequently, each task can be seen as a semantic domain with some relative fine-grained entity types, more similar to the complicated applications in real world.

5.2 CNER Settings

Before experimental evaluation, we need to split the original dataset reasonably to construct a series of CL dataset. Nevertheless, we discover the obvious diversity of different split methods in previous works, thus making their results difficult to compare. Following Zhang and Chen (2023), the diversity of CL dataset includes two aspects: **(1)** To divide the initial train/dev dataset into a series of train/dev dataset for each CL task, Monaikul et al. (2021) separates original samples randomly into disjoint datasets, while Xia et al. (2022) filters samples having entity types designated to be learned in that corresponding task. The above division patterns are defined as *Split* and *Filter*, respectively. **(2)** When constructing the test dataset evaluated in each CL task, Monaikul et al. (2021) directly adopt the full version of initial test set, while Xia et al. (2022) still filters test samples having one or more entity types learned so far. The different splits of test set are represented as *All* and *Filter*, Similarly.

Armed with these observations, there exist four combinations of CNER training and testing: *Split-All*, *Split-Filter*, *Filter-All*, *Filter-Filter*. We select ***Split-All*** and ***Filter-Filter*** for comprehensive evaluation of proposed framework and reliable comparations with previous works.

5.3 Baselines and Evaluation metrics

We compare GenCNER with typical baselines using different recognizing mechanisms. (1) Sequence labeling: **AddNER** and **ExtendNER** (Monaikul et al., 2021) are the early state-of-the-art works to apply sequence labeling and knowledge distillation to CNER task. **L&R** (Xia et al., 2022) enhances ExtendNER with an extra reviewing stage of synthetic augmented samples. We can not compare with CFNER (Zheng et al., 2022), CPFD (Zhang et al., 2023) and O_CILNER (Ma et al., 2023) since they employ data sampling operations and different CL settings. (2) Span-based: **SpanKL** (Zhang and Chen, 2023) is the first to introduce span-based classification to achieve coherent optimization in CNER. **SKD-NER** (Chen and He, 2023) employs reinforcement learning strategies during span-based KD process to optimize the soft labeling and distillation loss.

For the evaluation metrics, we employ the Macro F1 score of all seen types up to each learning step, and report the average results of tests on different pre-defined learning orders (See Appendix B). Since the fine-grained types in Few-NERD are not balanced, we first compute the Micro F1 of each coarse-grained type then fairly report the Macro F1 of all seen coarse-grained types. Additionally, we report the gap between non-CL and CL in each incremental task. The non-CL setting adds all the previous training datasets to the current one and provides full annotations of entity types learned so far, which serves as the upper bound of model performance at each CL step.

5.4 Implementation Details

During data preparation, we continuously add entity types as special tokens to the BART Tokenizer with CL tasks increasing. Meanwhile, we apply the n shift (n is the length of input sentences after padding) to each entity type index, which helps us identify whether the predicted index is a boundary token or entity type. See Appendix C for more experimental and hyper-parameter settings.

5.5 Main Results

We evaluate GenCNER performance on OntoNotes in two CNER settings: *Split-All* and *Filter-Filter*. The former randomly splits the original training dataset into disjoint subsets and adopts the full version of initial test set, which is the most common setup for CNER. The latter selects samples having current entity types from initial training set and filters samples with types seen so far in original test set, lacking training samples without any entity.

As shown in Table 1, GenCNER significantly outperforms other baselines in both CNER settings. It not only achieves the highest Macro F1 scores at each step, but also has the smallest gap to the non-CL performance. Compared with previous SOTA method SpanKL, GenCNER achieves +0.97%, +0.69% F1 higher, and narrows the per-

Method	Train(**Split**)-Test(**All**)						Train(**Filter**)-Test(**Filter**)					
	Step1	Step2	Step3	Step4	Step5	Step6	Step1	Step2	Step3	Step4	Step5	Step6
non-CL	82.52	86.42	87.32	88.84	89.62	89.27	90.78	91.54	90.76	90.60	90.50	90.48
AddNER	82.52	83.90	84.66	85.02	85.48	85.03	90.78	89.82	88.92	87.20	86.16	85.82
Δ	-0.00	-2.52	-2.66	-3.82	-4.14	-4.24	-0.00	-1.72	-1.84	-2.40	-3.34	-4.66
non-CL	82.79	86.49	87.70	88.46	89.02	89.19	90.62	91.70	91.02	90.79	90.92	90.10
ExtendNER	82.79	83.54	84.48	84.67	85.12	84.96	90.62	88.92	87.55	86.30	84.77	81.37
Δ	-0.00	-2.95	-3.22	-3.79	-3.90	-4.23	-0.00	-2.78	-3.47	-4.49	-6.15	-8.73
non-CL	-	-	-	-	-	-	92.06	91.16	90.50	89.69	89.57	89.30
L&R	-	-	-	-	-	-	92.06	88.09	85.69	83.79	83.38	83.02
Δ	-	-	-	-	-	-	-0.00	-3.07	-4.81	-5.90	-6.19	-6.28
SKD-NER	87.33	91.47	92.36	90.76	88.54	88.17	-	-	-	-	-	-
non-CL	85.60	88.16	88.64	89.39	89.69	89.74	92.37	92.65	92.78	92.06	92.10	91.90
SpanKL	85.60	87.92	88.22	88.76	89.02	88.98	92.37	90.81	90.38	89.50	89.18	88.07
Δ	-0.00	-0.24	-0.42	-0.63	-0.67	-0.76	-0.00	-1.84	-2.40	-2.56	-2.92	-3.83
non-CL	86.37	89.26	89.41	89.93	90.40	90.57	92.69	93.48	93.67	92.96	92.91	92.23
GenCNER	86.37	88.84	89.06	89.42	89.71	**89.95**	92.69	91.86	91.39	90.13	89.73	**88.76**
Δ	-0.00	-0.42	-0.35	-0.51	-0.69	**-0.62**	-0.00	-1.62	-2.28	-2.83	-3.18	**-3.47**

Table 1: Macro F1 of different models at each CL step in *Split-All* and *Filter-Filter* settings on OntoNotes. Δ = CL − non-CL. The highest F1 score at final learning step is bolded along with its gap. All methods adopt the large version of BERT or BART as encoder, ensuring their parameter sizes are at the same level.

Method	Train(**Split**)-Test(**All**)							
	Step1	Step2	Step3	Step4	Step5	Step6	Step7	Step8
non-CL	64.01	62.25	61.88	61.07	61.28	62.43	63.83	63.65
AddNER	64.01	61.32	60.54	59.43	58.74	59.32	60.41	59.32
Δ	-0.00	-0.93	-1.34	-1.64	-2.54	-3.11	-3.42	-4.33
non-CL	64.06	62.08	61.76	60.94	61.29	62.02	63.62	63.32
ExtendNER	64.06	59.02	57.05	55.72	55.46	55.96	56.85	56.16
Δ	-0.00	-3.06	-4.71	-5.22	-5.83	-6.06	-6.77	-7.16
L&R	68.13	66.72	64.51	63.44	60.97	61.23	60.88	60.32
non-CL	67.81	65.22	64.97	64.18	64.22	64.94	66.10	65.76
SpanKL	67.81	64.16	63.62	62.31	61.67	62.17	63.24	62.15
Δ	-0.00	-1.06	-1.35	-1.86	-2.55	-2.77	-2.86	-3.61
non-CL	68.27	65.73	65.36	64.63	64.92	65.48	66.52	66.31
GenCNER	68.27	64.75	64.10	63.02	62.28	62.95	63.74	**62.96**
Δ	-0.00	-0.98	-1.26	-1.61	-2.64	-2.53	-2.78	**-3.35**

Table 2: Macro F1 on Few-NERD in *Split-All* CL settings. Each step involves one coarse-grained entity type containing some coarse-grained ones.

formance gap by the absolute value of 0.14, 0.36 at the final learning step in *Split-All* and *Filter-Filter* settings, respectively. Experimental results on OntoNotes demonstrate the effectiveness of GenCNER to recognize entities of growing types. We also discover that different methods perform similarly in non-CL setup but quite diversely for CNER, revealing they may be strong enough to solve traditional NER but have varying degrees of difficulty in handling a series of CL tasks.

Following recent works (Zhang and Chen, 2023), we test GenCNER on a more complicated dataset Few-NERD under typical *Split-All* CL setting and display the results in Table 2. Each step represents one coarse-grained entity type, which contains several fine-grained ones and well imitates real-life scenarios. From Table 2, we observe that although facing the challenge of learning multiple fine-grained types per task, GenCNER still performs better than other baselines. To be specific, GenCNER improves the F1 score by 0.81% at the final step and reduces the gap from -3.61 to -3.35.

We briefly analyze the core factors leading to performance improvement. As we convert CNER into sequence generation of rising entity triplets, the problem of semantic shift no longer exists, thus achieving coherent optimization. During KD stage, we abandon pseudo triplets with lower confidence in a type-specific way, which reduces the label noise and prevents subsequent error accumulation.

5.6 Further Analysis

Ablation Study To explore the effectiveness of confidence-based filtering strategy and KL divergence loss of soft pseudo labels, we conduct necessary ablation experiments on OntoNotes. As shown in Table 3, when we remove confidence threshold filtering from GenCNER, the F1 score decreases by 0.44%, 1.35% at the final step. Meanwhile, we find the *Filter-Filter* setting suffers more obvious performance decline. That may be because *Filter-Filter* constructs training sets by filtering samples with specific entity types, causing intersection between different sets. Thus there exist more samples of type co-occurence in *Filter-Filter* CL setting, potentially bringing more noise during old model prediction. With regard to pseudo label format and optimized objectives, the F1 score reveals a slight

Method	Train(**Split**)-Test(**All**)						Train(**Filter**)-Test(**Filter**)					
	Step1	Step2	Step3	Step4	Step5	Step6	Step1	Step2	Step3	Step4	Step5	Step6
GenCNER	86.37	88.84	89.06	89.42	89.71	**89.95**	92.69	91.86	91.39	90.13	89.73	**88.76**
w/o *CTF*	86.37	88.56	88.67	89.10	89.38	89.51	92.69	91.14	90.43	89.07	88.68	87.41
w/o $\mathcal{L}_{KD}$	86.37	88.65	88.81	89.26	89.32	89.73	92.69	91.46	90.72	89.62	89.23	88.29

Table 3: The ablation study on OntoNotes in two CL settings. "*CTF*" denotes Confidence Threshold Filtering.

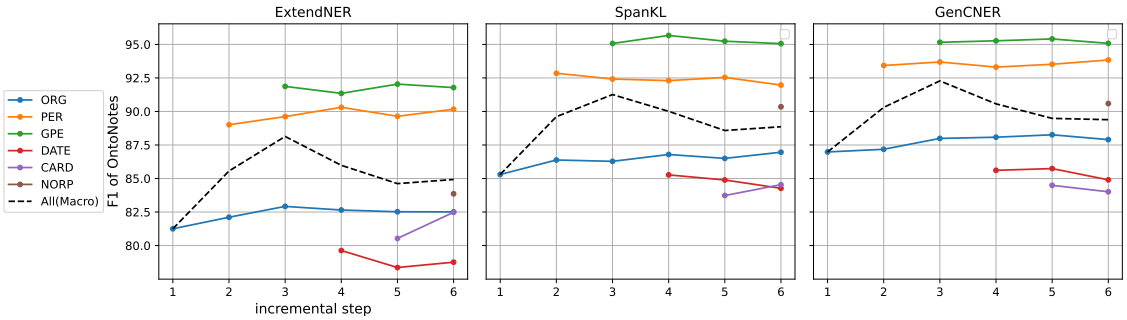

Figure 3: F1 curves of involved OntoNotes entity type(s) at each step with a certain learning order in *Split-All* setup.

decrease if we replace KL divergence with cross entropy loss, indicating the strength of soft label learning, as expected.

Type-Level Comparison We further study the instant performance of involved entity types at each CL step. Following a certain learning order, We plot their F1 curves on OntoNotes in *Split-All* setting and add an extra dashed curve to denote the overall Macro F1 up to per step. As illustrated in Figure 3, GenCNER outperforms other baselines in the following aspects: (1) The starting point of each curve in GenCNER is always superior to that in other methods, indicating the stronger power of GenCNER in acquiring new type knowledge. (2) The trend of most F1 curves in GenCNER is obviously smoother with CL tasks growing, which reveals it effectively prevents the new model from forgetting already learned entity types.

Effect of triplet compositions During entity triplet sequence generation, we utilize the classic autoregressive framework, where current decoder input depends on the output token from last time step. Considering the influence of conditional probability in seq2seq models, we evaluate three compositions of entity triplet on Ontonotes in *Split-All*: TSE (type, start, end), STE and SET. From Table 4, we find that the composition of triples has a certain impact on model performance and SET achieves the best, indicating the recognition of boundary tokens promotes the classification of entity types.

Analysis of Threshold Selection We finally study the selection of type-specific threshold δ on two datasets at the second CL step in *Split-All* setting. Figure 4 reveals that F1 score first increases then

Method	Train(**Split**)-Test(**All**)					
	Step1	Step2	Step3	Step4	Step5	Step6
TSE	86.05	88.46	88.77	88.89	89.32	89.63
STE	86.16	88.53	88.70	89.22	89.48	89.79
SET	86.37	88.84	89.06	89.42	89.71	**89.95**

Table 4: Macro F1 of different entity triplet compositions on OntoNotes in *Split-All* setting.

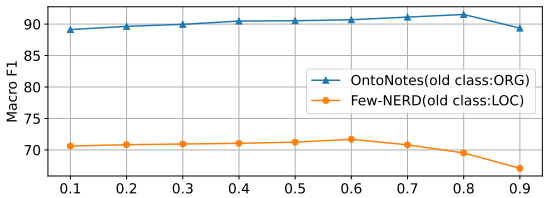

Figure 4: Effect of type-specific threshold δ selection.

decreases as the threshold increases progressively. Such trend indicates that reasonable threshold settings can help reduce label noise, while excessive thresholds may lead to insufficient training samples.

6 Conclusion

In this paper, we propose a novel generative framework for continual NER. By formulating CNER into sequence generation of increasing entity triplets, GenCNER averts the semantic shift of non-entity and solves a series of CL tasks excellently. We perform type-specific confidence threshold filtering to select high-quality pseudo triplets generated by old model, which effectively reduces the potential noise and enhances knowledge distillation. Experiments on two NER datasets show GenCNER achieves new SOTA performance over strong baselines in multiple CL settings.

Limitations

Compared with other methods, GenCNER costs more training and inference time since it adopts the autoregressive structure to generate triplet sequence in each CL task. In addition, the knowledge distillation greatly depend on the co-occurrence of entity types between different tasks. The model performance will significantly decrease if current task contains fewer previously learned entity types. We plan to explore the above issues for future work.

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Computer Science*.

Yi Chen and Liang He. 2023. SKD-NER: Continual named entity recognition via span-based knowledge distillation with reinforcement learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6689–6700, Singapore. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. 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 3198–3213, Online. Association for Computational Linguistics.

Octavian-Eugen Ganea and Thomas Hofmann. 2017. Deep joint entity disambiguation with local neural attention. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2619–2629, Copenhagen, Denmark. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Eduard Hovy, Mitchell 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, New York City, USA. Association for Computational Linguistics.

Xuming Hu, Fukun Ma, Chenyao Liu, Chenwei Zhang, Lijie Wen, and Philip S Yu. 2021. Semi-supervised relation extraction via incremental meta self-training. In *Proc. of EMNLP: Findings*.

Arzoo Katiyar and Claire Cardie. 2018. Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Phong Le and Ivan Titov. 2018. Improving entity linking by modeling latent relations between mentions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1595–1604, Melbourne, Australia. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2022a. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.

Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022b. Unified named entity recognition as word-word relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Shuliang Liu, Xuming Hu, Chenwei Zhang, Shu'ang Li, Lijie Wen, and Philip S. Yu. 2022. Hiure: Hierarchical exemplar contrastive learning for unsupervised relation extraction. In *Proc. of NAACL-HLT*, pages 5970–5980.

Ruotian Ma, Xuanting Chen, Zhang Lin, Xin Zhou, Junzhe Wang, Tao Gui, Qi Zhang, Xiang Gao, and Yun Wen Chen. 2023. Learning "O" helps for learning more: Handling the unlabeled entity problem for class-incremental NER. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5959–5979, Toronto, Canada. Association for Computational Linguistics.

Alejandro Metke-Jimenez and Sarvnaz Karimi. 2015. Concept extraction to identify adverse drug reactions in medical forums: A comparison of algorithms. *ArXiv*, abs/1504.06936.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *International Conference on Machine Learning*, pages 5926–5936.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:9016–9023.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, et al. 2023. Freematch: Self-adaptive thresholding for semi-supervised learning. In *Eleventh International Conference on Learning Representations*. OpenReview.net.

Yu Xia, Quan Wang, Yajuan Lyu, Yong Zhu, Wenhao Wu, Sujian Li, and Dai Dai. 2022. Learn and review: Enhancing continual named entity recognition via reviewing synthetic samples. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2291–2300, Dublin, Ireland. Association for Computational Linguistics.

Hang Yan, Tao Gui, Junqi Dai, Qipeng Guo, Zheng Zhang, and Xipeng Qiu. 2021. A unified generative framework for various NER subtasks. 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 5808–5822, Online. Association for Computational Linguistics.

Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge graph contrastive learning for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1434–1443, New York, NY, USA. Association for Computing Machinery.

Jianfei Yu, Ziyan Li, Jieming Wang, and Rui Xia. 2023. Grounded multimodal named entity recognition on social media. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9141–9154, Toronto, Canada. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. 2021. Flexmatch: boosting semi-supervised learning with curriculum pseudo labeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA. Curran Associates Inc.

Duzhen Zhang, Wei Cong, Jiahua Dong, Yahan Yu, Xiuyi Chen, Yonggang Zhang, and Zhen Fang. 2023. Continual named entity recognition without catastrophic forgetting. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8186–8197, Singapore. Association for Computational Linguistics.

Yunan Zhang and Qingcai Chen. 2023. A neural span-based continual named entity recognition model. In *AAAI Conference on Artificial Intelligence*.

Junhao Zheng, Zhanxian Liang, Haibin Chen, and Qianli Ma. 2022. Distilling causal effect from miscellaneous other-class for continual named entity recognition. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3602–3615, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Fine-grained Entity Types of Few-NERD

We evaluate all the 66 fine-grained entity types of Few-NERD and show their names and corresponding coarse-grained entity categories in Table 5.

coarse-grained category	fine-grained types
location (LOC)	GPE park island mountain bodiesofwater road/railway/highway/transit other
person (PER)	actor athlete scholar director soldier politician artist/author other
organization (ORG)	company religion education sportsteam sportsleague politicalparty showorganization media/newspaper government/governmentagency other
other (OTH)	god law astronomything award biologything chemicalthing currency disease educationaldegree language livingthing medical
product (PROD)	airplane car food game ship software train weapon other
build (BUID)	airport hospital hotel library restaurant sportsfacility theater other
art (ART)	broadcastprogram film music painting writtenart other
event (EVET)	attack/battle/war/militaryconflict disaster election protest sports other

Table 5: The fine-grained entity types of Few-NERD.

6 OntoNotes Permutations (→)						
1:	ORG	PER	GPE	DATE	CARD	NORP
2:	DATE	NORP	PER	CARD	ORG	GPE
3:	GPE	CARD	ORG	NORP	DATE	PER
4:	NORP	ORG	DATE	PER	GPE	CARD
5:	CARD	GPE	NORP	ORG	PER	DATE
6:	PER	DATE	CARD	GPE	NORP	ORG

4 Few-NERD Permutations (→)								
1:	LOC	PER	ORG	OTH	PROD	BUID	ART	EVET
2:	ORG	PROD	ART	EVET	OTH	PER	LOC	BUID
3:	PROD	EVET	OTH	PER	ART	LOC	BUID	ORG
4:	BUID	OTH	PROD	PER	ORG	LOC	ART	EVET

Table 6: Different learning orders of CL tasks on OntoNotes and Few-NERD.

B Learning Orders of CL tasks

We follow the recent works(Zhang and Chen, 2023; Chen and He, 2023) to select 6 permutations of tasks for OntoNotes and 4 permutations for Few-NERD respectively, aiming to eliminate the impact of learning order on model performance. The details are listed in Table 6.

C Hyper-parameter Settings

Towards model components, we apply the pre-trained seq2seq model BART-large to continuously generate entity triplet sequence. We only use BART embeddings with the dimension of 1024. During confidence-based pseudo labeling, we remove entity triplets predicted by old model with lower confidence than type-specific thresholds defined in formula 8. Considering the diverse learning difficulties of different entity types and the trade-off between quantity and quality of training samples, we set the threshold hyper-parameter δ as Table 7. To effectively balance the KD and CE loss, the weights are set to 1(α) and 0.5(β). In model training, we choose the AdamW optimizer with the learning rate 5e-5 and set warmup ratio to 0.1 for both datasets. The epoch number of each CL task is 10 and the batch size is set to 8. We conduct each experiment on GeForce RTX 3090 GPU.

dataset	involved entity type	threshold δ
OntoNotes	ORG	0.8
	PER	0.7
	GPE	0.8
	DATE	0.6
	CARD	0.5
	NORP	0.5
Few-NERD	LOC	0.6
	PER	0.6
	ORG	0.5
	OTH	0.5
	PROD	0.3
	BUID	0.4
	ART	0.4
	EVET	0.3

Table 7: Type-specific threshold settings for two datasets. We test a range of thresholds on the validation set and select the optimal one.