Unify Named Entity Recognition Scenarios via Contrastive Real-Time Updating Prototype

Yanhe Liu[1], Peng Wang[1, 2*], Wenjun Ke[1, 2*], Guozheng Li[1], Xiye Chen[3], Jiteng Zhao[1], Ziyu Shang[1]

[1]School of Computer Science and Engineering, Southeast University, Nanjing, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
[3]Nanjing University of Finance & Economics, Nanjing, China
{liuyanhe, pwang, kewenjun, liguozheng, jiteng, ziyus1999}@seu.edu.cn, chemistrymaths2002@gmail.com

Abstract

Supervised named entity recognition (NER) aims to classify entity mentions into a fixed number of pre-defined types. However, in real-world scenarios, unknown entity types are continually involved. Naive fine-tuning will result in catastrophic forgetting on old entity types. Existing continual methods usually depend on knowledge distillation to alleviate forgetting, which are less effective on long task sequences. Moreover, most of them are specific to the class-incremental scenario and cannot adapt to the online scenario, which is more common in practice. In this paper, we propose a unified framework called Contrastive Real-time Updating Prototype (CRUP) that can handle different scenarios for NER. Specifically, we train a Gaussian projection model by a regularized contrastive objective. After training on each batch, we store the mean vectors of representations belong to new entity types as their prototypes. Meanwhile, we update existing prototypes belong to old types only based on representations of the current batch. The final prototypes will be used for the nearest class mean classification. In this way, CRUP can handle different scenarios through its batch-wise learning. Moreover, CRUP can alleviate forgetting in continual scenarios only with current data instead of old data. To comprehensively evaluate CRUP, we construct extensive benchmarks based on various datasets. Experimental results show that CRUP significantly outperforms baselines in continual scenarios and is also competitive in the supervised scenario.

Introduction

Traditional supervised named entity recognition (NER) aims to recognize entity mentions from the given text and classify them into the pre-defined types such as *Person*, *Location*, *Organization* and *MISC* (Li et al. 2020). However, in the real-world applications, unknown entity types are continually involved. Although the large language models have shown impressive performance, they still need some techniques to learn new knowledge. As shown in Figure 1, compared with the traditional supervised scenario, two continual scenarios are more common: (1) Class-incremental (CI) scenario (Li and Hoiem 2017; Wang et al. 2019) provides a sequence of supervised tasks to learn. Each task involves at least one new entity type and the model are evaluated on all learned types. (2) Online scenario (Lopez-Paz and Ranzato 2017; Prabhu, Torr, and Dokania 2020) is similar to the CI scenario but only one batch or even one sample is available. For continual scenarios, naive fine-tuning will result in dramatic performance drop on old tasks called catastrophic forgetting (McCloskey and Cohen 1989; French 1999).

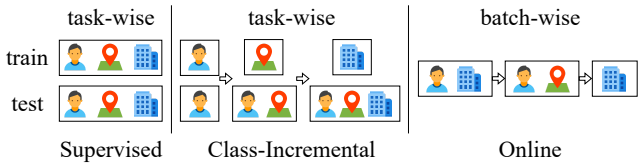


Figure 1: Three learning scenarios of NER. Each icon denotes an entity type.

To handle this problem, continual learning (Parisi et al. 2019; McCaffary 2021) has been introduced to NER in recent years (Monaikul et al. 2021; Wang et al. 2022; Xia et al. 2022). However, these methods usually depend on the knowledge distillation to alleviate forgetting, which are poor on long task sequences. Moreover, most of them are specific to the CI scenario and cannot adapt to the online scenario.

Recent studies show that memory-based methods are quite effective (Lopez-Paz and Ranzato 2017; Chaudhry et al. 2019; Han et al. 2020; Wang et al. 2019; de Masson D'Autume et al. 2019). These methods typically maintain a memory buffer to store some representative samples and replay them later. However, they may lead to overfitting on memorized samples and have to deal with the increment of the classes. Therefore, mean vectors of representations called prototypes are used to represent class distributions by some latest methods (De Lange and Tuytelaars 2021; Mai et al. 2021; Cui et al. 2021; Zhao et al. 2022). Instead of the Softmax linear classification, the nearest class mean classification is conducted between the prototypes and the representations, which is not affected by the increment of the classes. Nevertheless, prototype-based methods still memorize some data to update the prototypes after each task.

To address above issues, we propose a unified framework called Contrastive Real-time Updating Prototype (CRUP) for different named entity recognition scenarios. It mainly

*Corresponding author.

consists of a learning module and an updating module: (1) The learning module generates high-quality representations of samples. Before each train step, an augmented batch is fed into the frozen encoder to get out-dated representations. After that, the encoder with a Gaussian projection head is optimized by a contrastive objective with a designed regularization, which can generate more transferable representations to alleviate forgetting. (2) The updating module stores and updates prototypes. After training a batch, the updating module feeds the same batch into the optimized encoder again to get up-to-date representations. Then we initialize prototypes of new entity types with their mean vectors of representations, while update existing prototypes of old types only with the out-dated and up-to-date representations. The final prototypes will be used for the nearest class mean classification in testing. In this way, CRUP can handle different scenarios especially continual scenarios without dependency on old data. To comprehensively evaluate CRUP, we construct extensive benchmarks based on various datasets. Experimental results show that CRUP significantly outperforms strong baselines in continual scenarios and is also competitive in the supervised scenario.

In summary, our main contributions are three-fold:

- We try to unify different scenarios for NER. To the best of our knowledge, we are among the first to address this problem.
- We propose a novel framework called contrastive real-time updating prototype (CRUP). It enhances the representations through contrastive learning. Prototypes instead of samples are stored and updated at real time to alleviate forgetting.
- We introduce some latest and difficult datasets to evaluate CRUP. Experimental results demonstrate that CRUP outperforms strong baselines.

Related Work

Continual Learning

Existing continual learning methods can be roughly divided into three categories: (1) Regularization-based methods (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Li and Hoiem 2017) prevent the important parameters from significant changes, (2) Architecture-based methods (Chen, Goodfellow, and Shlens 2016; Fernando et al. 2017) expand the model architecture to accommodate new knowledge, and (3) Memory-based methods (Lopez-Paz and Ranzato 2017; Chaudhry et al. 2019; de Masson D'Autume et al. 2019) memorize some old data to preserve old knowledge. Recent continual NER methods mainly leverage knowledge distillation to alleviate forgetting. (Monaikul et al. 2021) distills knowledge and expands the student model to learn new entity types (Monaikul et al. 2021). (Wang et al. 2022) augments training data with synthetic data and distills knowledge with both synthetic data and current data (Wang et al. 2022). (Xia et al. 2022) proposes a two-stage framework to distill both old and new knowledge to an enhanced student model (Xia et al. 2022). However, these methods are specific to the CI scenario and cannot perform well on long task sequences. Recent methods in image classification and relation extraction mainly rely on additional memory modules to recall old knowledge (Ye and Bors 2022; Pourcel, Vu, and French 2022; Han et al. 2020; Wang et al. 2019). However, these methods cause overfitting on memorized samples and have to deal with the increment of the classes. Instead of simple replay, prototype-based methods use memory to update prototypes for the nearest class mean classification (De Lange and Tuytelaars 2021; Zhao et al. 2022). Although such methods obtain competitive performance, they ignore data access restrictions in real-world applications. In this paper, we focus on how to unify both continual and supervised scenarios for NER only based on the current data.

Contrastive Learning

Contrastive learning aims to make similar samples closer to each other, while dissimilar samples should be farther away from each other in the embedding space (Jaiswal et al. 2020). The state-of-the-art methods in computer vision show that downstream tasks can benefit much from contrastive learning (Chen et al. 2020a,b). Therefore, recent continual learning methods also try to utilize contrastive learning to enhance continual learning. SCR (Mai et al. 2021) incorporates contrastive learning and replay to address continual image classification. CRL (Zhao et al. 2022) addresses continual relation extraction in a similar way. However, these methods are only capable of the CI scenario and require much additional time to update prototypes. In this paper, we also leverage contrastive learning to enhance continual learning but update prototypes at real time.

Methodology

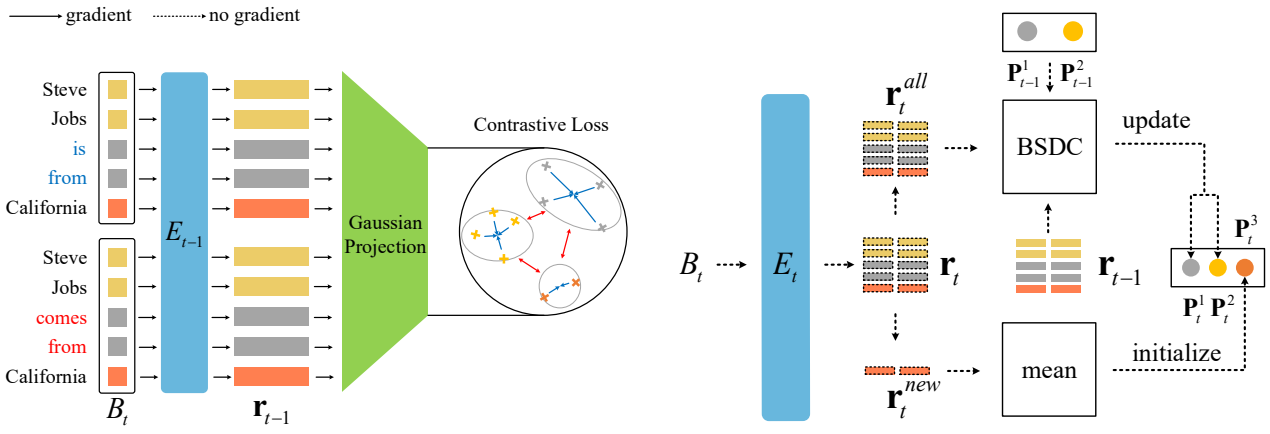
Problem Formulation

Given a sequence of NER tasks $\mathcal{T} = \{T_1, T_2, \dots\}$, each task T_i has its own dataset $\mathcal{D}_i = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^{N}$ and entity type set $\mathcal{Y}_i$. $\mathbf{x}_k$ is a token sequence $\{x_1, x_2, \dots\}$ and has a corresponding label sequence $\mathbf{y}_k = \{y_1, y_2, \dots\}, y \in \mathcal{Y}_i$. Descriptions of different learning scenarios are as follows:

- **Supervised scenario** provides only one task for the model, i.e. $|\mathcal{T}| = 1$, which is a typical supervised task.
- **Class-Incremental (CI)** scenario (Monaikul et al. 2021; Xia et al. 2022) requires the model to sequentially learn a sequence of supervised tasks, i.e. $|\mathcal{T}| > 1$. However, each task involves only one new entity type.
- **Online scenario** (Prabhu, Torr, and Dokania 2020; Ye and Bors 2022) provides several tasks while $\mathcal{D}_i$ is only one batch and the model should be both tested and trained on it. Each task may involve multiple and learned types.

Framework Overview

As shown in Figure 2, CRUP mainly consists of a learning module and an updating module. Before each train step, the learning module feeds an augmented batch into the frozen encoder to get out-dated representations. After that, the encoder with a Gaussian projection head is optimized by a contrastive objective with a designed regularization. Then the



(a) The learning module. (b) The updating module.

Figure 2: Overview of CRUP framework. At each train step t: (a) The learning module first feeds an augmented batch B_t into the frozen encoder E_{t-1} and the representations $\mathbf{r}_{t-1}$ are temporarily saved. After that, E_{t-1} is optimized by the supervised contrastive loss to E_t. (b) Then the updating module feeds B_t into E_t again to get up-to-date representations $\mathbf{r}_t$. For each new entity type involved by B_t, we initialize its prototype with the mean of its up-to-date representations. For each old entity type, we use $\mathbf{r}_t$ and $\mathbf{r}_{t-1}$ to update its prototype.

updating module feeds the same batch into the optimized encoder to get up-to-date representations to initialize or update prototypes. The final prototypes will be used for the nearest class mean classification in testing.

Learning Module

As shown in Figure 2 (a), the learning module trains an encoder with the supervised contrastive loss (SCL) (Khosla et al. 2020). Following (Monaikul et al. 2021; Xia et al. 2022), we use the pre-trained BERT (Devlin et al. 2019) as the encoder to get the token representations. Specifically, at the time step t, a batch with b tokens is firstly augmented to a new batch B_t with $2b$ tokens. Then B_t is fed into the frozen encoder E_{t-1} to get representations $\mathbf{r}_{t-1} \in \mathbb{R}^{d_E}$, where d_E is the representation dimension. Note that B_t and $\mathbf{r}_{t-1}$ are temporarily saved for updating prototypes later. Next, $\mathbf{r}_{t-1}$ is projected into an embedding space by the Gaussian projection head (Das et al. 2022). It consists of two networks, f_μ and f_Σ, which generate the Gaussian distribution parameters of $\mathbf{r}$:

$$\mu = f_\mu(\mathbf{r}), \Sigma = \text{Diag}(\text{ELU}(f_\Sigma(\mathbf{r})) + (1+\epsilon)), \quad (1)$$

where $\mu \in \mathbb{R}^{b\times d_p}$, $\mathbf{\Sigma} \in \mathbb{R}^{b\times d_p\times d_p}$ are the mean and diagonal covariance of Gaussian embedding respectively, d_p is the embedding dimension, ELU is an exponential linear unit, and $\epsilon \approx 1e-14$ is for numerical stability.

Given Gaussian embeddings $\mathcal{N}_i(\boldsymbol{\mu}_i, \mathbf{\Sigma}_i), \mathcal{N}_j(\boldsymbol{\mu}_j, \mathbf{\Sigma}_j)$ of two representations $\mathbf{r}^i, \mathbf{r}^j$, the similarity between two distributions is measured by the KL-divergence calculated as

$$\begin{aligned} KL(\mathbf{r}_i|\mathbf{r}_j) = &\frac{1}{2}(\text{trace}(\mathbf{\Sigma}_j^{-1}\mathbf{\Sigma}_i) + \\ &(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i)^{\mathrm{T}}\mathbf{\Sigma}_j^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_i) + \\ &\log\frac{|\mathbf{\Sigma}_j|}{|\mathbf{\Sigma}_i|} - d_p). \end{aligned} \quad (2)$$

Both directions are considered since the KL-divergence is not symmetric:

$$d(\mathbf{r}_i, \mathbf{r}_j) = \frac{1}{2}\left(KL(\mathbf{r}_i|\mathbf{r}_j) + KL(\mathbf{r}_j|\mathbf{r}_i)\right). \quad (3)$$

The loss of each representation $\mathbf{r}_i$ in B_t can be calculated by

$$\ell_i = -\log\frac{\exp\left(-d(\mathbf{r}_i, \mathbf{r}_p)/\tau\right)}{\sum_{a\in A(i)}\exp\left(-d(\mathbf{r}_i, \mathbf{r}_a)/\tau\right)}. \quad (4)$$

The sum loss of all $\mathbf{r}_i$ in B_t is

$$\mathcal{L}^{\text{con}} = \sum_{i\in I}\frac{1}{|P(i)|}\sum_{p\in P(i)}\ell_i, \quad (5)$$

where $i \in I \equiv \{1, \ldots, 2b\}$ is the index of token within B_t and $A(i) \equiv I \setminus \{i\}$. The token indexed by i is called the *anchor*. $P(i) \equiv \{p \in A(i) : y_p = y_i\}$ is the set of indices of all the other tokens with the same label as token i called the *positives*. $\tau \in \mathbb{R}^+$ is a scalar temperature parameter which plays a role in controlling the strength of penalties on hard negative samples (Wang and Liu 2021). Although existing prototypes are outdated after the model is optimized, they can well represent the impressions about old entity types in the hidden space. Therefore, we introduce a designed online regularization in the CI scenario to alleviate forgetting. For each entity prototype $\mathbf{P}_k \in \mathcal{P}$, we push the entity embeddings in B_t from $\mathbf{P}_k$ and draw the non-entity embeddings in B_t to $\mathbf{P}_0$ by minimizing:

$$R_k = \sum_{m\in M(k)} -\log\frac{\exp\left(-d(\mathbf{P}_k, \mathbf{r}_m^k)/\tau\right)}{\sum_{i\in I}\exp\left(-d(\mathbf{P}_k, \mathbf{r}_i)/\tau\right)}, \quad (6)$$

where $M(k) \equiv \{m \in I : y_m = k\}$ is the set of indices of tokens labeled with k in B_t. The final regularization is the

sum of all R_k:

$$R = \sum_{P_k \in \mathcal{P}} \frac{1}{|M(k)|} R_k. \tag{7}$$

Finally, the optimization objective for the CI scenario is a weighted sum of $\mathcal{L}^{\text{con}}$ and R:

$$\mathcal{L} = (1-\alpha) \cdot \mathcal{L}^{\text{con}} + \alpha \cdot R. \tag{8}$$

Updating Module

As shown in Figure 2 (b), the updating module maintains the prototypes after a step of training. Firstly, B_t is fed to the frozen optimized encoder E_t to get up-to-date representations $\mathbf{r}_t$. For each new entity type involved by B_t, its prototype is initialized with the mean:

$$\mathbf{P}_t^{\text{new}_i} = \frac{1}{N} \sum \mathbf{r}_t^{\text{new}_i}, \tag{9}$$

where N is the number of tokens annotated by i-th new entity type in B_t. Then $P_t^{\text{new}_i}$ is saved for classification. However, existing prototypes $\{P_{t-1}^k\}$ are out-dated because the hidden space is changed. Although memorizing some old data to update prototypes may be effective, it ignores data access restrictions and require additional time. Moreover, B_t typically involves few data about old entity types especially in unbalanced data distribution.

To handle this problem, we propose a batch-wise real-time updating way inspired by semantic drift compensation (SDC) (Yu et al. 2020) called batch-wise SDC (BSDC). As shown in Figure 3 (a) and Figure 3 (b), given out-dated prototypes $\sum_k P_{t-1}^k$s and representations $\sum_i \mathbf{r}_{t-1}^i$ at time step t in the hidden space, thee semantic drift of a prototype P^k is defined as:

$$\Delta_{t-1\to t}^k = \mathbf{P}_t^k - \mathbf{P}_{t-1}^k, \tag{10}$$

where P_t^k denotes the true prototype after learning the current batch B_t.

Our target is to approximate every $\Delta_{t-1\to t}^k$ only with the current data. As shown in Figure 3 (c), the semantic drift of a token in B is:

$$\delta_{t-1\to t}^i = \mathbf{r}_t^i - \mathbf{r}_{t-1}^i. \tag{11}$$

Then the sparse vector field of these drifts is used to approximate $\Delta_{t-1\to t}^k$ as shown in Figure 3 (d):

$$\hat{\Delta}_{t-1\to t}^k = \frac{\sum_i w^i \delta_{t-1\to t}^i}{\sum_i w^i} \tag{12}$$

with

$$w_i = e^{-\frac{\|\mathbf{r}_{t-1}^i - P_{t-1}^k\|^2}{2\sigma^2}}, \tag{13}$$

where σ is the standard deviation of euclidean distances between $\{\mathbf{r}_{t-1}^i\}$ and P_{t-1}. Finally, the prototype is updated as following:

$$\hat{\mathbf{P}}_t^k = \hat{\Delta}_{t-1\to t}^k + \mathbf{P}_{t-1}^k. \tag{14}$$

Therefore, existing prototypes are updated at real time without dependency on old data.

Nearest Class Mean Classification

In the testing phase, we discard f_μ and f_Σ and use the nearest class mean classification to predict the label of a token. Specifically, we compare the representation $\mathbf{r}$ of a token with existing prototypes and predict its label by

$$\hat{y} = \underset{y \in \sum_{i=1}^n \mathcal{Y}_i}{\arg\min} \|\mathbf{r} - \mathbf{P}_y\|_2^2, \tag{15}$$

where $\mathbf{P}_y$ is the prototype of entity labeled by y.

Experiments

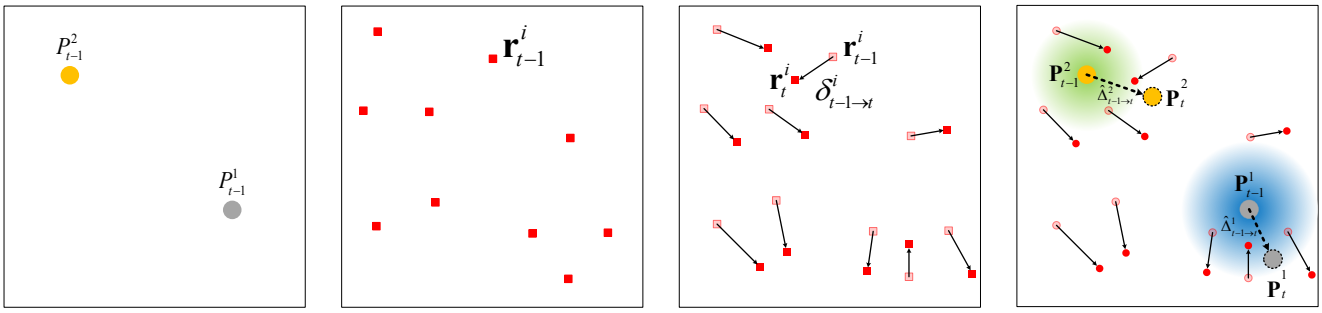
Datasets

Following (Monaikul et al. 2021; Xia et al. 2022), we conduct experiments on CoNLL-03 (Tjong Kim Sang and De Meulder 2003) and OntoNotes-5.0 (Weischedel et al. 2013). Moreover, we introduce two annotation versions of Few-NERD (Ding et al. 2021) denoted by FewNER-8 and FewNER-66 respectively, and StackOverflowNER (Tabassum et al. 2020; Payan et al. 2021) denoted by SO. Since SO is quite imbalanced, we eliminate entity types with less than 200 sentences with 19 entity types left. Following (Monaikul et al. 2021; Xia et al. 2022), we conduct experiments on the orders from the Latin square of each dataset in CI and online scenarios. We use the IO annotation schema, where we annotates entity tokens with some **I** label and other tokens with **O** label. Details of setups and metrics are as follows.

- Supervised: We split the data for training, validating and testing with a ratio of $7:1:2$. We use the precision, recall and F1-score to measure the performance.
- Class-Incremental: We divide the supervised datasets into disjoint subsets to construct supervised task sequences. We mask other entity types in the train set of each task to involve one entity type. Then we mask all unknown entity types in the test set of each task. At each step k, we compute the F1-scores on all test sets until T_k denoted by AF1^k, i.e. $\text{AF1}^k = \frac{1}{k}\sum_{i=1}^k \text{F1}_i$.
- Online: We use all samples of each dataset to generate online data streams. We mask all types except one in the first 20 batches. Then we unmask a new type every 20 batches and compute the F1-score every 10 batches.

Baselines

We compare our framework with various methods in different scenarios as follows: **Bert-Tagger** (Ding et al. 2021) conducts softmax classification on representations from the encoder. **AddNER** (Monaikul et al. 2021) adds an output layer for each new task and then distills knowledge. **ExtendNER** (Monaikul et al. 2021) also distills knowledge but extends the dimension of the output layer. **L&R** (Xia et al. 2022) distills twice to obtain an enhanced student model on each task. **GDumb** (Prabhu, Torr, and Dokania 2020) maintains a balanced memory sampled from training data and use it to retrain a model for testing. **CoPE** (De Lange and Tuytelaars 2021) updates prototypes based on a balanced memory. **ODDL** (Ye and Bors 2022) estimates the discrepancy between the current memory and the already accumulated knowledge. **DSDM** (Pourcel, Vu, and French 2022) evolves



(a) Out-dated prototypes. (b) Current representations. (c) The drift vector field. (d) Updated prototypes.

Figure 3: Illustration of BSDC.

dynamically and continually modeling the distribution of any non-stationary data stream. **ChatGPT-3.5** is a popular large language model and we ask it to conduct CNER. Specifically, we first tell it to behave as an entity recognizer. Then we explain each entity type and feed it all train samples to predict labels in a designed format. Here is an output example:

> *Detective* [**O**] *Earl* [**PER**] *Feugill* [**PER**] , [**O**] *camouflaged* [**O**] *as* [**O**] *a* [**O**] *shaggy* [**O**] *green* [**O**] *bush* [**O**] , [**O**] *ordered* [**O**] *them* [**O**] *to* [**O**] *freeze* [**O**] . [**O**]

Implementation Details

For all methods, we use BERT (Devlin et al. 2019) as the encoder implemented by `bert-base-uncased` in the huggingface transformers library (Wolf et al. 2020). We set the batch size as 32 and the max length of token sequence as 64. f_μ and f_Σ are implemented as multi-layer networks with the output dimension of 64. We compare different augmentation methods (Wei and Zou 2019) and find no difference in performance. Therefore, we augment one sample with two views through the random deletion and the random swap. All models are trained for 20 epochs on each supervised task with early stopping (patience=3). We set the learning rate as 5e-5, the max size of replay memory of GDumb and CoPE as 1000, the temperature parameter τ as 0.1 and the regularization weight α as 0.2. The weights of CE loss and distillation loss are balanced for distillation-based methods. We set the momentum parameter α of CoPE to 0.1. Our experiments are accelerated by GeForce RTX 3090.

Results and Discussion

Main Results

CI Scenario The top half of Table 1 shows the results on each task of CoNLL-03 and OntoNotes-5.0, which provides a short task sequence (T1∼T4, T1∼T6). We find that distillation-based methods (AddNER, ExtendNER and L&R) perform well in the early tasks, while they suffer from degradation on latter tasks. It is probably caused by the error accumulation during distillation among the tasks. Meanwhile, the performance of GDumb keeps stable on the overall sequence but is not good on each single task. The reason is that GDumb is similar to the multi-task learning, while it only maintains limited subsets from seen tasks instead of all seen datasets. CoPE, ODDL and DSDM also achieve competitive performance, since a training batch for CoPE is composed by a batch from training data and a batch from a balanced replay memory. Then it uses a weighted sum of the old prototype and the mean of representations to approximate the true center of mass in the hidden space. However, CoPE fixes weights of the sum and does not make use of the representations of other entity types. ODDL and DSDM are both designed to handle the online scenario, which cannot handle the CI scenario especially in later tasks. ChatGPT achieves the second best performance on most steps as expected. CRUP outperforms the above baselines on almost every single step and its overall performance is more stable.

Recent continual NER methods are only evaluated on a few tasks from balanced datasets like CoNLL-03 and OntoNotes-5.0. Therefore, we use FewNER-8, FewNER-66 and SO to further evaluate continual NER performance on the long and imbalanced task sequences. Especially, SO involves many entity types from the programming domain with a unbalanced distribution. Due to the space limitation, we report the sampled results from the whole CI task sequences of these five datasets in the bottom half of Table 1. Consistent with observations on two classical datasets, performances of distillation-based methods drop faster than other baselines. GDumb is effective and stable on early tasks (∼T20) of FewNER-66. However, as more tasks come, each entity type only have few samples, which leads to sharp performance drop in the later tasks. CoPE, ODDL and DSDM are affected for the same reason. This problem is more obvious on SO since it has much less samples for each task than FewNER-66. ChatGPT even loses much performance on SO, which demonstrates that it has only learn general knowledge and still need to enable the continual learning capability. Despite the performance degradation on early tasks of FewNER-66, CRUP still performs more stably over the sequence and outperforms all other baselines on SO.

Online Scenario Most of existing continual NER methods including AddNER, ExtendNER and L&R focus on the CI scenario while ignore the online scenario. Therefore, we conduct experiments in the online scenario to evaluate on-

Method	CoNLL-03				OntoNotes-5.0					
	T1	T2	T3	T4	T1	T2	T3	T4	T5	T6
AddNER	92.1	73.8	58.0	43.6	88.3	71.7	58.2	45.6	44.6	36.5
ExtendNER	92.2	69.8	57.3	38.2	88.2	69.8	58.1	43.0	38.8	34.8
L&R	92.2	82.7	74.3	61.2	88.3	81.4	73.3	63.9	51.2	38.3
GDumb	82.3	78.6	75.9	71.4	83.1	72.3	67.5	62.1	58.1	54.3
CoPE	92.4	83.5	76.3	62.6	87.4	81.1	71.4	62.5	56.5	53.3
ODDL	92.3	85.2	75.3	67.1	88.1	81.5	73.8	64.4	56.3	53.6
DSDM	92.4	84.6	76.1	70.7	88.0	81.6	72.9	64.6	58.2	52.1
ChatGPT	88.1	84.2	80.1	**75.6**	86.5	82.1	74.7	67.8	61.5	57.3
CRUP	**92.4**	**87.6**	**80.3**	73.2	**88.3**	**82.2**	**75.5**	**68.1**	**62.3**	**58.4**
Method	FewNER-8				FewNER-66			SO		
	T2	T4	T6	T8	T20	T40	T60	T6	T12	T18
AddNER	69.7	48.3	30.1	19.5	3.61	2.19	1.09	1.60	1.14	0.81
ExtendNER	65.3	47.4	29.3	18.7	4.52	2.26	1.36	1.43	1.10	0.30
L&R	73.6	56.4	45.3	34.8	19.2	5.24	2.35	15.1	8.22	4.15
GDumb	64.2	58.6	55.0	49.7	**36.4**	10.4	1.42	1.93	0.53	0.63
CoPE	75.2	56.3	45.7	36.9	19.7	13.6	7.31	16.1	11.7	8.93
ODDL	76.1	57.7	48.0	43.9	19.6	14.1	7.57	17.3	12.0	7.29
DSDM	76.1	58.9	49.1	43.8	20.8	14.2	8.18	16.8	12.1	9.34
ChatGPT	76.0	65.4	56.4	48.6	36.3	18.5	11.0	7.55	4.76	1.85
CRUP	**78.1**	**66.3**	**58.2**	**51.2**	27.1	**18.7**	**11.1**	**21.6**	**14.1**	**9.73**

Table 1: CI results on CoNLL-03, OntoNotes-5.0, FewNER-8, FewNER-66 and SO.

Method	FewNER-66			SO		
	P	R	F1	P	R	F1
Bert-Tagger	53.21	55.80	54.48	55.56	44.78	49.59
ChatGPT	55.79	56.69	56.24	46.12	41.16	43.50
CRUP	**60.72**	**57.21**	**58.91**	**59.63**	**54.70**	**57.05**

Table 2: Supervised results on FewNER and SO.

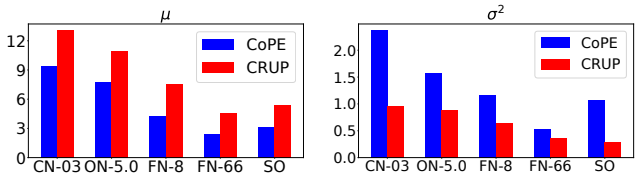


Figure 4: The mean μ ($\times 0.01$) and variance σ^2 ($\times 0.01$) of distances among the normalized prototypes. The method with higher μ and lower σ^2 is better.

line learning abilities of the models. Figure 5 reports the results on every 10 batches of the first 200 batches. Note that a sample in this scenario may involve multiple entity types, making it difficulty to swap samples to maintain balanced memory buffers for GDumb and CoPE. Therefore, we first keep all seen samples until the memory is full, then randomly swap out a sample and make sure that all seen entity types are involved in the memory. We find that GDumb suffers from instability since it is more sensitive to memory balance. ODDL and DSDM also rely on their stored data, whose quality is affected by the unstable data distribution. ChatGPT is immune to the shifts of data distribution, while it cannot continually improve its performance. Overall, benefited from real-time updating prototypes, CRUP is more stable and keeps optimizing over the data stream.

Supervised Scenario Although CRUP is mainly designed for continual NER scenarios, the real-time learning way can be also applied in the supervised scenario with the mini-batch gradient descent. Therefore, we also evaluate our method in the supervised scenario. As shown in Table 2, CRUP also achieves impressive performance. This is consistent with what we have observed in online experiments because the online training can be viewed as the one-epoch supervised training.

Quality of Prototypes

The above results especially in the online scenario show that CRUP achieves excellent performances in different scenarios. We argue that a crucial reason is that CRUP can produce well-separated hidden space as well as high-quality prototypes. To compare the quality of prototypes, we calculate the mean μ and variance σ^2 of distances among normalized prototypes generated by CoPE and CRUP. Higher μ and lower σ^2 demonstrate that the prototypes are evenly distributed in the hidden space. As shown in Figure 4, CRUP separates prototypes better than CoPE. CoPE uses a similar idea as contrastive learning that encourages t he inter-class variance and reduces the intra-class variance. However, it assumes

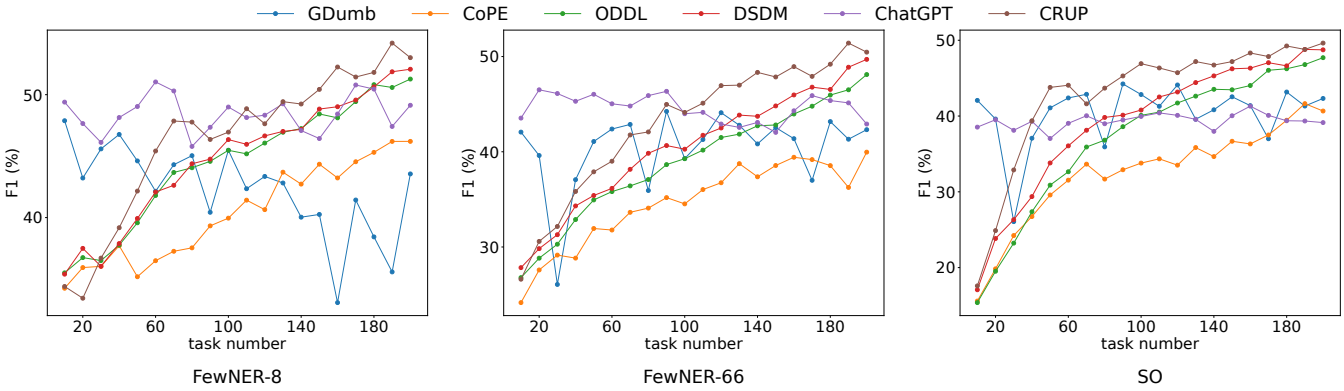


Figure 5: Online results on FewNER-8, FewNER-66 and SO.

Method	FewNER-8				FewNER-66		
	T2	T4	T6	T8	T20	T40	T60
Replace-CE	21.5	8.00	10.7	4.39	10.2	1.66	3.75
Mean-CE	44.6	23.8	12.6	8.00	10.7	2.00	1.01
BSDC-CE	75.4	61.9	53.7	44.8	3.61	1.93	1.36
BSDC-Con	76.2	65.5	57.3	49.6	26.2	16.8	8.23
CRUP-EWC	76.5	65.6	55.4	48.2	24.9	15.7	7.05
CRUP-SI	76.3	65.9	55.1	48.7	25.0	15.9	7.44
Point	16.9	4.37	1.41	0.85	2.10	0.57	0.26
CRUP	**78.1**	**66.3**	**58.2**	**51.2**	**27.1**	**18.7**	**9.10**

Table 3: CI results of different CRUP variants.

samples are independent, which is reasonable in CV tasks but not appropriate for NER, since there exists contextual dependencies between entity tokens within the same sample. Moreover, contrastive methods in CV field mostly map representations to certain point embeddings, which cannot reflect uncertainty about the concept of a sample (Vilnis and McCallum 2015). We find that this projection is less effective in our experiments. Instead, the Gaussian projection can model dependencies and uncertainty by mapping a representation into a continuous area and using KL-divergence to measure the similarity. We further verify our analysis through the following ablation study.

Effects of CRUP Components

We also introduce some variants of CRUP as follows to demonstrate the effect of each component: **Replace-CE** replaces the old prototype with the mean vectors of up-to-date representations. **Mean-CE** uses the mean vector between the old prototype and up-to-date representations as new prototypes. **BSDC-CE** uses BSDC to update prototypes. **BSDC-Con** uses BSDC and the contrastive objective instead of the cross entropy loss without the designed regularization. **CRUP-EWC** and **CRUP-SI** use EWC and SI as the regularization respectively. **Point** maps hidden representations to point embeddings instead of Gaussian embeddings. **CRUP** is our approach that uses BSDC, the regularized contrastive objective and the Gaussian projection.

As shown in Table 3, we can make following observations: (1) Two heuristic methods, Replace-CE and Mean-CE, also obtain good performance and sometimes are even better than AddNER and ExtendNER. Moreover, we find that BSDC-CE further improves performance on early tasks of FewNER, while it does not work well on later steps of the CI benchmarks. We argue that it is because BSDC compensates semantic drifts of old prototypes in the CI scenario only based on the sample drifts of the current entity type and the non-entity type. Therefore, the non-entity drifts contribute more to compensation of prototypes because old entity types are viewed as non-entity type in the current task. This error is accumulated as more tasks come. However, BSDC-CE works well on SO, which indicates that it can fast adapt to unbalanced and even few-shot scenarios. Moreover, the introduction of contrastive learning improves BSDC performance in most cases. The sample drifts of entity and non-entity types are more fairly considered to compensate the old prototypes in the well-separated hidden space generated by contrastive learning. Finally, compared with BSDC-Con, CRUP further improves the performance in the CI scenario due to the effect of our designed regularization. (2) Other regularization techniques like EWC and SI only focus on the change of parameters and improve little even hurt performance. By contrast, our regularization considers the changes of prototypes, which are more representative than parameters, and thus performs much better. (3) The point projection underperforms the Gaussian projection with a considerable gap, which verifies our analysis on the quality of prototypes.

Conclusion

In this paper, we propose a framework called CRUP to unify common scenarios for NER. It introduces contrastive learning to enhance representations. The prototypes instead of samples are stored to alleviate forgetting and updated at real time only with the current data. Experimental results demonstrate the superiority of CRUP on various scenarios. However, CRUP involves high complexity during optimization. We will explore some more efficient substitutes of current learning objective in the future work.

Acknowledgements

We thank the anonymous reviewers for their insightful comments. This work was supported by National Science Foundation of China (Grant Nos.62376057) and the Start-up Research Fund of Southeast University (RF1028623234). All opinions are of the authors and do not reflect the view of sponsors.

References

Chaudhry, A.; Ranzato, M.; Rohrbach, M.; and Elhoseiny, M. 2019. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*.

Chen, T.; Goodfellow, I.; and Shlens, J. 2016. Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*.

Chen, X.; Fan, H.; Girshick, R.; and He, K. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Cui, L.; Yang, D.; Yu, J.; Hu, C.; Cheng, J.; Yi, J.; and Xiao, Y. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. 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)*, 232–243.

Das, S. S. S.; Katiyar, A.; Passonneau, R.; and Zhang, R. 2022. CONTaiNER: Few-Shot Named Entity Recognition via Contrastive Learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

De Lange, M.; and Tuytelaars, T. 2021. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

de Masson D'Autume, C.; Ruder, S.; Kong, L.; and Yogatama, D. 2019. Episodic memory in lifelong language learning. In *Conference and Workshop on Neural Information Processing Systems*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Ding, N.; Xu, G.; Chen, Y.; Wang, X.; Han, X.; Xie, P.; Zheng, H.; and Liu, Z. 2021. Few-NERD: A Few-shot Named Entity Recognition Dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Fernando, C.; Banarse, D.; Blundell, C.; Zwols, Y.; Ha, D.; Rusu, A. A.; Pritzel, A.; and Wierstra, D. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*.

Han, X.; Dai, Y.; Gao, T.; Lin, Y.; Liu, Z.; Li, P.; Sun, M.; and Zhou, J. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jaiswal, A.; Babu, A. R.; Zadeh, M. Z.; Banerjee, D.; and Makedon, F. 2020. A survey on contrastive self-supervised learning. *Technologies*.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. In *Conference and Workshop on Neural Information Processing Systems*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*.

Li, J.; Sun, A.; Han, J.; and Li, C. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. In *Conference and Workshop on Neural Information Processing Systems*.

Mai, Z.; Li, R.; Kim, H.; and Sanner, S. 2021. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

McCaffary, D. 2021. Towards continual task learning in artificial neural networks: current approaches and insights from neuroscience. *arXiv preprint arXiv:2112.14146*.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*.

Monaikul, N.; Castellucci, G.; Filice, S.; and Rokhlenko, O. 2021. Continual learning for named entity recognition. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter, S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*.

Payan, J.; Merhav, Y.; Xie, H.; Krishna, S.; Ramakrishna, A.; Sridhar, M.; and Gupta, R. 2021. Towards Realistic Single-Task Continuous Learning Research for NER. In *Findings of the Association for Computational Linguistics: EMNLP 2021*.

Pourcel, J.; Vu, N.-S.; and French, R. M. 2022. Online Task-free Continual Learning with Dynamic Sparse Distributed Memory. In *European Conference on Computer Vision*.

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*.

Tabassum, J.; Maddela, M.; Xu, W.; and Ritter, A. 2020. Code and Named Entity Recognition in StackOverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Tjong Kim Sang, E. F.; and De Meulder, F. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL*.

Vilnis, L.; and McCallum, A. 2015. Word representations via gaussian embedding. In *International Conference on Learning Representations*.

Wang, F.; and Liu, H. 2021. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Wang, H.; Xiong, W.; Yu, M.; Guo, X.; Chang, S.; and Wang, W. Y. 2019. Sentence Embedding Alignment for Lifelong Relation Extraction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.

Wang, R.; Yu, T.; Zhao, H.; Kim, S.; Mitra, S.; Zhang, R.; and Henao, R. 2022. Few-Shot Class-Incremental Learning for Named Entity Recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Wei, J.; and Zou, K. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Weischedel, R.; Palmer, M.; Marcus, M.; Hovy, E.; Pradhan, S.; Ramshaw, L.; Xue, N.; Taylor, A.; Kaufman, J.; Franchini, M.; El-Bachouti, M.; Belvin, R.; and Houston, A. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadel- phia, PA, 23*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; and et al. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

Xia, Y.; Wang, Q.; Lyu, Y.; Zhu, Y.; Wu, W.; Li, S.; and Dai, D. 2022. Learn and Review: Enhancing Continual Named Entity Recognition via Reviewing Synthetic Samples. In *Findings of the Association for Computational Linguistics*.

Ye, F.; and Bors, A. G. 2022. Task-free continual learning via online discrepancy distance learning. *Advances in Neural Information Processing Systems*.

Yu, L.; Twardowski, B.; Liu, X.; Herranz, L.; Wang, K.; Cheng, Y.; Jui, S.; and Weijer, J. v. d. 2020. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*.

Zhao, K.; Xu, H.; Yang, J.; and Gao, K. 2022. Consistent Representation Learning for Continual Relation Extraction. In *Findings of the Association for Computational Linguistics*.