Continual Few-shot Event Detection via Hierarchical Augmentation Networks

Chenlong Zhang[1,2]*, **Pengfei Cao**[1,2]*, **Yubo Chen**[1,2]†, **Kang Liu**[1,2,3]
Zhiqiang Zhang[4], **Mengshu Sun**[4], **Jun Zhao**[1,2]
[1]The Laboratory of Cognition and Decision Intelligence for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]Shanghai Artificial Intelligence Laboratory, Shanghai, China
[4]Ant Group, Hangzhou, China
zhangchenlong2023@ia.ac.cn
{pengfei.cao, yubo.chen, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Traditional continual event detection relies on abundant labeled data for training, which is often impractical to obtain in real-world applications. In this paper, we introduce continual few-shot event detection (CFED), a more commonly encountered scenario when a substantial number of labeled samples are not accessible. The CFED task is challenging as it involves memorizing previous event types and learning new event types with few-shot samples. To mitigate these challenges, we propose a memory-based framework: **H**ierarchical **A**ugmentation **Net**works (**HANet**). To memorize previous event types with limited memory, we incorporate *prototypical augmentation* into the memory set. For the issue of learning new event types in few-shot scenarios, we propose a *contrastive augmentation* module for token representations. Despite comparing with previous state-of-the-art methods, we also conduct comparisons with ChatGPT. Experiment results demonstrate that our method significantly outperforms all of these methods in multiple continual few-shot event detection tasks.

Keywords: Information Extraction, Continual Learning, Few-shot Learning

1. Introduction

Event Detection (ED) involves detecting event triggers and classifying the corresponding event types (Ahn, 2006) (e.g., in Figure 1, the words "married" and "left" trigger events "Marry" and "Transport", respectively.). It is an essential information extraction task that can be applied in various natural language processing applications. Conventional methods (Chen et al., 2015; Nguyen and Grishman, 2015) commonly model ED as a supervised task trained on fixed data with pre-defined event types. However, in real-world applications, new event types emerge continually.

Thus, **Continual Event Detection** (CED) has been proposed (Cao et al., 2020; Yu et al., 2021). The CED task assumes multiple ED tasks emerge continually, which requires ED models to learn new types while maintaining the capability of detecting previous types. The CED task is challenging due to the catastrophic forgetting problem (McCloskey and Cohen, 1989), where the model's performance on previous tasks declines significantly when learning new tasks. To mitigate such a dilemma, previous works have proved that memory-based methods (see Figure 1) are the most effective in solving CED task (Cao et al., 2020; Yu et al., 2021; Liu et al., 2022). These methods preserve prototypical samples as memory set to replay previous knowledge. Abundant representative features can effectively remind the model of previous types, achieving state-of-the-art performance.

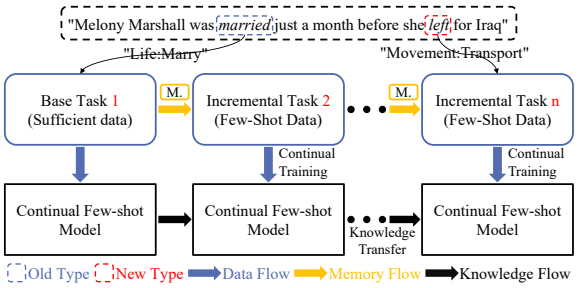

Figure 1: Memory-based framework for continual few-shot event detection. It preserves previous knowledge by maintaining a memory set "M." and transferring knowledge from previous models.

Even though these methods achieve remarkable performance, they all assume that the training samples in incremental tasks are sufficient. Actually, in practical applications, new events emerge successively, making it infeasible to obtain a sufficient number of high-quality samples for each emerging new event type. It is more commonplace to encounter incremental tasks with only a handful of annotated samples (e.g., 10, 5, or even 1) for

* These authors contribute equally to this work.
† Corresponding author.

LREC-COLING 2024, pages 3868–3880

each new type. Nonetheless, this circumstance has been overlooked by previous works.

To this end, we propose a new task: **Continual Few-shot Event Detection** (CFED), which aims to continually learn new event detection tasks with few-shot samples. For example, as shown in Figure 1, the first task (base task) denotes the regular ED task with abundant training samples (e.g., 100 samples are available for event type "Life: Marry"). Then, only a few samples are available for the emerging incremental tasks (e.g., there are only 5 labeled samples accessible for the new type "Movement: Transport").

Obviously, CFED introduces a more challenging yet realistic scenario as it requires *memorizing previous event types* and *learning new event types* with few-shot samples. We present the two challenges specifically as follows:

Memorizing previous event types with few-shot samples: In the CED task, memroy-based methods use a multitude of exemplars (e.g., 50) in memory set to effectively characterize the prototypical feature space, thus alleviating catastrophic forgetting. However, in the CFED task, only 10, 5, or 1 sample is available for training. In extreme scenarios, there is only one sample per type available to be stored in the memory set for further replay. Therefore, how to utilize rare stored samples to mitigate catastrophic forgetting remains challenging.

Learning new event types with few-shot samples: Supervised methods usually require a large number of annotated samples (Lai et al., 2020; Deng et al., 2020; Zhang et al., 2022a). When trained with limited samples, these methods often struggle to generalize well and suffer from overfitting. Current large language models (llms) (Brown et al., 2020; Touvron et al., 2023) have demonstrated promising capability to learn from few-shot samples with their in-context learning ability. However, these models are constrained by limited knowledge (e.g., ChatGPT's knowledge of world and events is limited after 2021). Though in-context learning is capable of temporarily empowering them with new event knowledge, it fails to truly inject this knowledge into the model(Moiseev et al., 2022). Therefore, We consider using a fine-tuned language model to solve the CFED task. How to effectively mitigate overfitting with few-shot samples for learning new event types is still a formidable challenge.

To address these problems, we propose a memory-based approach: **H**ierarchical **A**ugmentation **Net**work (**HANet**). When memorizing previous types, we devise *prototypical augmentation* to augment the prototypical feature space of exemplars in the memory, thus alleviating catastrophic forgetting. To address overfitting in learning new types, we design *contrastive augmentation* module to acquire valuable information from few-shot samples. Experimental results show that our method surpasses previous baselines significantly.

Our contributions can be summarized as follows:

(1) To the best of our knowledge, we are the first to propose continual few-shot event detection and construct benchmarks based on ACE and MAVEN.

(2) We propose a **H**ierarchical **A**ugmentation **Net**work (**HANet**), which leverage prototypical augmentation and contrastive augmentation to memorize previous event types and to learn new event types with few-shot samples.

(3) Experimental results demonstrate that our method significantly outperforms previous state-of-the-art methods in all CFED settings. Impressively, our method achieves 7.27% and 8.44% improvements on micro F1 in 4-way 5-shot MAVEN and 2-way 5-shot ACE settings. Moreover, experiments with ChatGPT show that our method achieves superior results. Our code and dataset are publicly available at https://github.com/chenlong-clock/CFED-HANet.

2. Problem Definition

Continual few-shot event detection (CFED) aims to detect emerging events with few-shot samples. As shown in Figure 1, given tasks $\mathbb{T} = \{T_1, T_2, ..., T_n\}$, each task has individual training/validation/testing set $T_i = \{D_i^{train}, D_i^{dev}, D_i^{test}\}$. $D_i = \left\{(\mathbf{X}_i^j, \mathbf{Y}_i^j)\right\}_{j=1}^{m}$, where $\mathbf{X}$ and $\mathbf{Y}$ are samples and their corresponding labels, and m is the number of event types in each task. The first sub-task T_1 is the base task T_{base} that contains abundant training samples. The rest sub-tasks are defined as few-shot incremental tasks $T_{inc} = \{T_2, T_3, ..., T_n\}$, with only a few samples (e.g., 5 or 10) for each new event type. For any two tasks T_i and T_j, their types are non-overlapping: $T_i \cap T_j = \emptyset$. At time step t, for CFED task C_t, the training set is formulated as $C_t^{train} = D_t^{train}$ and the validation/testing set is $C_t^{test} = D_t^{test} \bigcup C_{t-1}^{test}$, indicating the CFED system is supposed to keep stable performance on all observed labels $L_t = \bigcup_{i=1}^{t} \{\mathbf{Y}_i^j\}_{j=1}^{m}$ with the currently available training samples in task T_t.

3. Methodology

The framework of our method is illustrated in Figure 2. It comprises a general *event detector*, a memory enhanced by *prototypical augmentation*, and a *contrastive augmentation* module. For input sentences, *event detector* performs trigger extraction. Then, the exemplars are augmented by *prototypical augmentation* to replay previous knowledge. Additionally, contrastive augmentation exploits infor-

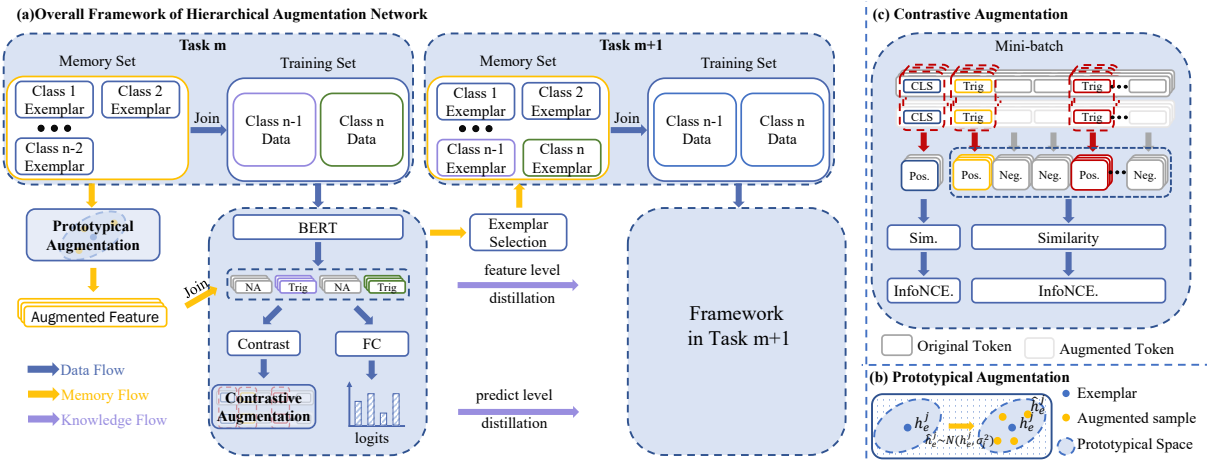

Figure 2: Our system consists of a general event detector, prototypical augmentation, and contrastive augmentation. When learning new tasks with an event detector, the model replays prior knowledge from the augmented feature. Then, contrastive augmentation maximizes the acquisition of knowledge from few-shot samples.

mation from each sample by applying an auxiliary contrastive loss. We provide a detailed introduction as follows.

3.1. Event Detector

The event detector is composed of a trigger extractor and a classifier. Following previous works (Cao et al., 2020; Liu et al., 2022), we implement a pre-trained 12-layer BERT (Devlin et al., 2019) model to encode sentences. Specifically, given a sentence $\mathbf{S} = \{\mathbf{x}_1, \mathbf{x}_2, ..., [\mathbf{e}_s, ..., \mathbf{e}_e], ..., \mathbf{x}_n\}$ containing event triggers $\mathbf{E} = [\mathbf{e}_s, ..., \mathbf{e}_e]$, the hidden representation is $\mathbf{H} \in \mathbb{R}^{n \times d}$. We get hidden states of a trigger $\mathbf{H_e}$ by concatenating their start and end representations. Then, $p(\mathbf{y}_i|\mathbf{h}_e)$ for event type $\mathbf{y}_i \in L_t$ at stage t is obtained by the following equation:

$$p(\mathbf{y}_i|\mathbf{h}_e) = \frac{\exp\left(\mathbf{W}_i^T \mathbf{h}_e + \mathbf{b}_i\right)}{\sum_{j=1}^{|L_t|} \exp\left(\mathbf{W}_j^T \mathbf{h}_e + \mathbf{b}_j\right)} \quad (1)$$

where $\mathbf{W}_i \in \mathbb{R}^{d \times |L_t|}$ is a linear projection for classification. The possible types are L_t. Then, we train the model with Cross Entropy Loss:

$$\mathcal{L}_{ce} = - \sum_{(X,Y) \in T_t} \mathbf{y} \log \mathbf{p} \quad (2)$$

where $\mathbf{y}$ is the ground-truth label for trigger $\mathbf{h}_e$, $\mathbf{p}$ is the label distribution calculated by Equation (1).

3.2. Prototypical Augmentation

We construct a memory set by selecting the most representative examples. Accordingly, we adopt a distance-based algorithm. Finally, prototypical augmentation is applied in the feature space.

3.2.1. Memory Construction

After task T_t, we combine a memory set M_t comprising exemplars of current types with previous memory M_{t-1}. Since only few samples are available for training in incremental tasks, the most extreme condition should be taken into account so that our method can be compatible with any real-world applications. Thus, we only select one exemplar $(\mathbf{x}_{e,t}^j, \mathbf{y}_{e,t}^j)$ for every category in T_t:

$$M_t = \begin{cases} \left\{(\mathbf{x}_{e,t}^j, \mathbf{y}_{e,t}^j)\right\}_{j=1}^{m}, & \text{if } t = 1 \\ \left\{(\mathbf{x}_{e,t}^j, \mathbf{y}_{e,t}^j)\right\}_{j=1}^{m} \bigcup M_{t-1}, & \text{if } t > 1 \end{cases} \quad (3)$$

The combined M_t is then treated as a part of the training set in the next task $T_{t+1} = T_{t+1} \bigcup M_t$. To select the most representative samples, we first create a prototype for each event type by averaging the encoded representations. Then we choose the closest sample measured by distance (e.g., L_2 *Distance* or *Cosine Distance*) as the exemplar.

3.2.2. Prototypical Augmentation

Since conventional memory preserves plenty of representative samples, these samples characterize the feature space of their types. However, in our settings, the memory is limited to 1 for each type. The exemplar can only be represented as a point in the feature space (see Figure 2 (b)). To tackle this, we reconstruct the feature space of the exemplar by prototypical augmentation.

We get the exemplar's representation $\mathbf{h}_e^j$ that belongs to class j. We assume the pseudo feature space follows Gaussian Distribution. In view that exemplars are normally considered the most representative sample, their representation is regarded

as the mean. The variance of the distribution is calculated in the exemplar selection process, where we calculate the mean squared deviation of all samples that belong to the same category:

$$\sigma_j^2 = \frac{1}{|\mathbf{H}_e^j|} \sum_{\mathbf{h}_i^j \in \mathbf{H}^j} (\mathbf{h}_i^j - \mu_j)^2 \tag{4}$$

where H_e^j are BERT representations that belong to event type $\mathbf{Y}_t^j$. According to Equation (3), the memory set M_{t-1} is reformulated as $M_{t-1} = \bigcup_{k=1}^{i-1} \left\{(\mathbf{x}_{e,k}^j, \mathbf{y}_{e,k}^j, {\sigma^2}_t^j)\right\}_{j=1}^{m}$. We define the mean squared deviation of all exemplars as the variants of Gaussian distribution. When replaying exemplars, given the representation of exemplar $\mathbf{h}_e^j$, we have $\mu_j = \mathbf{h}_e^j$. Then, we sample from the distribution to construct synthetic features multiple times:

$$\hat{\mathbf{H}}_e^j = \{\hat{\mathbf{h}}_{e,1}^j, \ldots, \hat{\mathbf{h}}_{e,n}^j\} \sim \mathcal{N}(\mu_j, \sigma_j^2) \tag{5}$$

These synthetic features can represent the feature space of their category (i.e., prototypical space). Then we replay the memory:

$$\mathcal{L}_{re} = -\sum^{\hat{\mathbf{H}}_e^j} \mathbf{y}_j \log \hat{\mathbf{p}}_j \tag{6}$$

where $\hat{\mathbf{p}}_j$ is obtained from $\mathbf{H}_e^j$ by Equation (1).

3.3. Contrastive Augmentation

Overfitting is likely to appear in T_{inc} when learning few-shot new event types. As shown in Figure 2(c), we propose contrastive augmentation (CA) to uncover the implicit inter-information in the token scale. Following Zhang et al. (2022b), we use multiple data augmentations (e.g., Dropout, Random Token Shuffle, and Random Token Replacement) to generate augmented tokens. These tokens are used to construct positive and negative pairs. Finally, we propose two contrastive losses to aggregate the information.

3.3.1. Contrastive Pairs Construction

We first construct positive pairs and negative pairs from batched data. Specifically, given a mini-batch $\mathcal{B} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, the original sentences are $(\mathbf{x}_i^1, \mathbf{y}_i^1)$ and the augmented sentences are $\{(\mathbf{x}_i^k, \mathbf{y}_i^k)\}_{k=2}^{m+1}$, where m is a hyperparameter, denoting the augmentation times. Thus, sentences that have the same origin can be described as $\mathcal{O} = \{(\mathbf{x}_i^k, \mathbf{y}_i^k)\}_{k=1}^{m+1}$. Based on these pairs, we perform contrastive learning in sentence representation and trigger representation.

3.3.2. Contrastive Sentence Representation Learning

As in BERT, the special [CLS] token generally conveys the sentence representation. Similar to Mou et al. (2022), we utilize contrastive sentence representation learning for $\mathbf{h}_{cls}$. Representations originating from the same sentence are regarded as positive pairs and those that originate from different sentences are regarded as positive pairs. We leverage InfoNCE loss (Oord et al., 2018):

$$\mathcal{L}_{cls} = \frac{1}{n-1} \sum_{i}^{|\mathcal{B}|} -\frac{1}{m} \sum_{j \neq k}^{|\mathcal{O}|} \log \frac{\exp(S(\mathbf{h}_{cls_i}^j, \mathbf{h}_{cls_i}^k)/\tau)}{\sum_{p \neq i}^{|\mathcal{B}|} \sum_{q}^{|\mathcal{O}|} \exp(S(\mathbf{h}_{cls_i}^j \mathbf{h}_{cls_p}^q)/\tau)} \tag{7}$$

where $S(\cdot)$ is the similarity function, and τ is a temperature parameter to smooth the distribution and control the similarity range by scaling the output.

3.3.3. Contrastive Trigger Representation Learning

Considering trigger representations, we propose to construct positive pairs when triggers within $\mathcal{B}$ belong to the same types, while they should form negative pairs when belonging to different types. The contrastive loss in trigger representation is:

$$\mathcal{L}_{trig} = \frac{1}{n-1} \sum_{i \neq l}^{|\mathcal{B}|} -\frac{1}{m} \sum_{j \neq k}^{|\mathcal{O}|} [\mathbf{y}_i^j = \mathbf{y}_l^k] \log \frac{\exp(S(\mathbf{h}_{e_i}^j, \mathbf{h}_{el}^k)/\tau)}{\sum_{p \neq i}^{|\mathcal{B}|} \sum_{q}^{|\mathcal{O}|} [\mathbf{y}_i^j \neq \mathbf{y}_p^q] \exp(S(\mathbf{h}_{e_i}^j, \mathbf{h}_{e_p}^q)/\tau)} \tag{8}$$

Algorithm 1 Training procedure

Require: Base task T_1, incremental task $\{T_2, ..., T_n\}$ and model's parameter θ

```
initialize θ_1 for base task T_1
update parameter θ_1 in task T_1 using loss function L_ce and L_cls
get memory set M_1 from T_1 and θ_1
for i = 2 to n do
    get a copy of the previous model's parameter θ_{i-1}
    freeze parameter θ_{i-1}
    get combined training set T_i = T_i ∪ M_{i-1}
    update parameter θ_i in task T_i using loss function L_ce, L_fd, L_pd, L_re, L_cls and L_trig
    get memory set M_i from T_i and θ_i
    update memory set M_i = M_i ∪ M_{i-1}
end for
```

3.4. Knowledge Distillation

Similar to Cao et al. (2020), we use Knowledge Distillation at feature-level and predict-level. At task T_t, we distill knowledge from T_{t-1} .

Feature-level Distillation. We get previously and currently normalized representations $\tilde{\mathbf{h}}$ and $\mathbf{h}$

at the last layer's hidden states. We measure the similarity by function $S(\cdot)$ (*Cosine Similarity*). The feature-level distillation loss is:

$$\mathcal{L}_{fd} = \sum_{(X,Y)\in T_t} 1 - S(\tilde{\mathbf{h}}, \mathbf{h}) \quad (9)$$

Predict-level Distillation. As is demonstrated in Hinton et al. (2015), given trigger representations h_e, we obtain probability distribution:

$$p(\mathbf{y}_i|\mathbf{h}_e) = \frac{\exp(\mathbf{W}_i^T\mathbf{h}_e + \mathbf{b}_i)/\tau_d}{\sum_{j\in L_{t-1}} \exp(\mathbf{W}_j^T\mathbf{h}_e + \mathbf{b}_j)/\tau_d} \quad (10)$$

where τ_d is the temperature to control the smoothness of the distribution target. We compute previous and current probability distribution $\tilde{\mathbf{p}}$ and $\mathbf{p}$ on previous label set L_{t-1}. The training objective is:

$$\mathcal{L}_{pd} = -\sum_{(X,Y)\in T_t} \tilde{\mathbf{p}} \log \mathbf{p} \quad (11)$$

3.5. Training

We present detailed training procedures in Algorithm 1. In view that $\mathcal{L}_{ce}$ is the primary training objective and $\mathcal{L}_{cls}$ plays an auxiliary role to help exploit sentence information, we enable $\mathcal{L}_{ce}$ and $\mathcal{L}_{cls}$ in T_{base}. In T_{inc}, we incorporate the distillation losses ($\mathcal{L}_{fd}$ and $\mathcal{L}_{pd}$) and the exemplar replay loss ($\mathcal{L}_{re}$) as they rely on previous knowledge for training. We exclusively enable $\mathcal{L}_{trig}$ in T_{inc} due to its superior effectiveness in few-shot learning. Each loss function is weighted by a factor λ_i, where $i \in \{ce, re, cls, trig, fd, pd\}$.

4. Experiments

4.1. Continual Few-shot Event Detection Benchmarks

We construct our benchmarks based on two publicly available datasets:

MAVEN (Wang et al., 2020): The original MAVEN dataset contains 168 event types, which is a massive general domain event detection dataset. Regarding the training/validation/testing split, similar to Yu et al. (2021), the test set is built upon the initial development set. We randomly select samples in the original training set to collect another development set. For incremental task split, we select the most frequent types to construct CFED tasks. Accordingly, we randomly sample 100 instances for each type in the base task, and 5 or 10 instances for each type in the incremental task.

ACE 2005 (Walker et al., 2006): The ACE 2005 dataset consists of 33 event types. The training/validation/testing split is formed by previously mentioned works (Yang and Mitchell, 2016; Nguyen et al., 2016). We execute the identical operation on the incremental task split as we do on the MAVEN dataset to construct CFED tasks.

Our experiments contain 5 sub-tasks. We define the task containing m event types for each sub-task and k training samples for each type as m-way k-$shot$ CFED task. We select 10 and 20 most frequent types to conduct 2-way 5-shot, 2-way 10-shot, 4-way 5-shot and 4-way 10-shot tasks. We randomly sample 100 instances for each type in T_{base}, 5 and 10 instances for each type in T_{inc}.

4.2. Evaluation Metrics

Following Cao et al. (2020), we use ***micro F1*** score to evaluate the performance under each stage. For stage C_i^{test} we calculate $F1_i$ on all observed event types, as is defined in section 2. ***Micro F1*** score enables a comprehensive evaluation of the prediction results for all categories. We define $\bar{F1}_{micro} = \sum_{i=1}^{n} F1_i$ as the metric for overall performance on CFED.

4.3. Baseline Systems

Fine-tune. We fine-tune BERT continually on every sub-task. Typically, this option is the lower boundary in Continual Learning.

Combined Retrain. We retrain the model by combining all training samples of currently known types every time a new task arrives. It is usually regarded as the upperbound.

EWC (Kirkpatrick et al., 2017), which is an regularization-based method. It applies a regularization term to restrict updates for parameters that are important for previous task.

LwF (Li and Hoiem, 2017), which contains a distillation module to match the probability of previous models to maintain previous knowledge.

ICaRL (Rebuffi et al., 2017), which is a memory-based method. Besides, they utilize a representation learning method.

KCN (Cao et al., 2020), which is a popular continual event detection method following the memory replay-knowledge distillation paradigm.

KT (Yu et al., 2021). It generally follows the memory-based paradigm with a novel initialization method to transfer knowledge.

EMP (Liu et al., 2022). Besides memory replay, it introduces prompt learning of each event type to load previous types' knowledge.

4.4. Implementation Details

All baselines are implemented in the same settings as follows. BERT model is the open-sourced 110M bert-base-uncased from HuggingFace[1]. The number of training iterations is 30, the batch size is 4,

[1] https://huggingface.co/bert-base-uncased

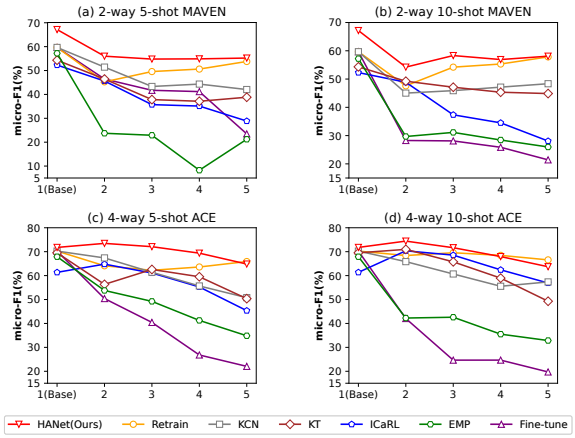

Figure 3: $F1_{micro}$ performance of every sub-task on 2-way MAVEN and 4-way ACE.

AdamW(Loshchilov and Hutter, 2019) is used as the optimizer, the learning rate is set to 2e-5, and the weight decay is set to 1e-4. The memory capacity is 1 for each type.All computations are performed on the NVIDIA GeForce RTX 3090 (24GB) platform with 5 different random seeds. More detailed implementations can be seen in the open-sourced code repository.

4.5. Main Results

We conduct each experiment 5 times and report the $\mathbf{means} \pm \mathbf{std}$. on MAVEN and ACE benchmarks in comparison with previously mentioned baselines. We report results in Table 1, and Table 2 and Figure 3. From the results, we can observe that:

(1) Compared with previous baselines, our approach significantly outperforms them across all sub-tasks. On 4-way 5-shot MAVEN and 2-way 5-shot ACE, our model obtains improvements of 7.27% and 8.44% on $\bar{F}1_{micro}$ when compared with previous state-of-the-art methods. Our approach even exceeds the strong retrain baseline with improvements of 5.94% and 5.56% on $\bar{F}1_{micro}$, which strongly proves the effectiveness of our approach.

(2) KCN and KT achieve relatively good performance. As we limit the memory capacity to only one sample for each type to replay, they can learn little knowledge from memory replay, which strongly demonstrates the importance of characterizing prototypical feature space.

(3) When compared with methods optimized for continual event detection, traditional methods: EWC, LwF, and ICaRL perform poorly. The giant gap between the lower bound and HANet illustrates that CFED is a challenging task.

4.6. Ablation Study

We conduct ablation study to validate the effectiveness of each component. We choose 2-way MAVEN for the ablation study in Table 3. The "Replay*" denotes removing memory replay. As prototypical augmentation is based on memory set, $\mathcal{L}_{re}$ is also set to 0 in "Replay*". The distillation losses $\mathcal{L}_{fd}$ and $\mathcal{L}_{pd}$ are removed in "w/o Distill". $\mathcal{L}_{re}$ and $\mathcal{L}_{cls}$ and $\mathcal{L}_{trig}$ are removed in settings "w/o PA" and "w/o CA", respectively. Here are the conclusions:

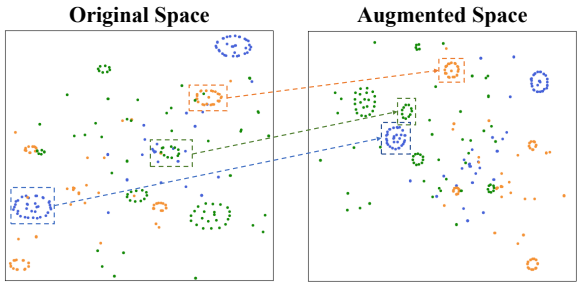

Figure 4: Embedding space visualization via t-SNE on original and prototypical augmented feature in task T_2. Points within the same color indicate identical event types. As we can see, after prototypical augmentation, the intra-class distances become closer for each type. Besides, some hard samples (pointed in the squared region) initially proximate to the centers of other classes in the original space become easier to classify after prototypical augmentation, showcasing the effectiveness of prototypical augmentation.

(1) **Effectiveness of Prototypical Augmentation.** Compared with removing prototypical augmentation (PA), PA boosts the performance by an average of 2.09% and 1.57%. Meanwhile, with the task proceeding, the model can gain more improvements, demonstrating that PA plays an increasingly vital effect in alleviating catastrophic forgetting. We also plot t-SNE visualization in Figure 4 to show how PA contributes to memorizing previous event types.

(2) **Effectiveness of Contrastive Augmentation.** In comparison with removing contrastive augmentation, our approach delivers improvements of 5.04% and 4.27% on $\bar{F}1_{micro}$, which indicates that contrastive augmentation is beneficial in mitigating overfitting in few-shot incremental tasks. Although we focus more on on T_{inc}, the model can greatly benefit from the auxiliary objectives in T_{base}.

(3) **Effectiveness of Prototypical Augmentation and Contrastive Augmentation.** When removing prototypical augmentation and contrastive augmentation, the $\bar{F}1_{micro}$ faces a sharp decline of 9.46% and 9.72%, implying the synergistic effect of the two modules to address the CFED problem.

Method	4-way 5-shot						4-way 10-shot					
	1	2	3	4	5	$\bar{F1}_{micro}$	1	2	3	4	5	$\bar{F1}_{micro}$
Fine-tune	40.43±2.34	33.17±3.55	17.5±2.07	19.72±0.92	21.01±0.87	26.36±1.3	40.43±2.34	38.18±2.83	20.46±1.11	20.35±2.19	23.57±1.01	28.6±0.92
Retrain	40.43±2.34	42.1±1.13	39.61±1.12	43.03±1.56	**47.43±0.67**	42.52±0.7	40.43±2.34	44.27±1.36	44.76±1.37	**48.28±1.43**	**53.66±0.97**	46.28±0.95
EWC	40.43±2.34	34.29±1.41	17.4±1.5	18.61±2.52	20.43±1.67	26.23±1.39	40.43±2.34	36.42±3.34	19.69±0.93	20.02±1.14	23.72±1.19	28.06±1.01
LwF	40.43±2.34	37.27±4.9	26.69±4.07	24.7±1.47	30.54±1.43	31.93±2.05	40.43±2.34	41.09±2.8	31.89±0.57	30.57±1.09	34.43±2.08	35.68±0.69
ICaRL	35.82±4.76	37.16±4.85	33.74±2.85	35.54±2.37	35.98±2.48	35.65±2.93	35.82±4.76	42.43±4.48	37.45±1.58	40.11±0.9	41.04±1.17	39.37±2.05
KCN	40.43±2.35	48.38±1.66	41.99±2.01	41.32±1.53	40.29±1.51	42.48±1.49	40.43±2.35	51.15±1.19	45.22±1.22	44.31±0.69	44.47±1.51	45.12±1.09
KT	41.04±1.59	40.19±2.17	35.21±1.34	32.69±0.78	33.77±0.58	36.58±1.06	41.04±1.59	44.39±0.91	40±1.3	39.42±0.33	37.87±0.95	40.54±0.58
EMP	40.17±1.34	30.95±0.75	31.21±1.32	22.9±2.09	22.25±1.43	29.5±0.76	40.17±1.34	32.33±0.69	32.95±1.11	26.68±1.5	28.16±1.89	32.06±0.8
HANet(Ours)	**41.91±3.76**	**51.39±1.55**	**43.21±3.19**	**43.53±4.21**	43.89±5.65	**44.79±2.33**	**41.91±3.76**	**53.17±1.27**	**46.71±2.51**	46.36±3.64	48.12±5.49	**47.25±2.23**

Table 1: $F1_{micro}$ of every sub-task and $\bar{F1}_{micro}$ across all sub-tasks on 4-way MAVEN benchmark.

Method	2-way 5-shot						2-way 10-shot					
	1	2	3	4	5	$\bar{F1}_{micro}$	1	2	3	4	5	$\bar{F1}_{micro}$
Fine-tune	60.86±2.96	52.09±9.59	46.37±10	26.64±6.98	23.15±4.66	41.82±3.56	60.86±2.96	48.17±9.8	49.55±2.91	23.29±8.2	24.66±3.23	41.31±3.31
Retrain	60.86±2.96	62.45±4.27	52.21±7.83	52.2±4.68	**58.36±6.09**	57.22±4.48	60.86±2.96	63.39±2.87	63.75±2.67	**61.23±2.08**	**64.25±3.13**	**62.7±1.3**
EWC	60.86±2.96	49.3±8.93	45.41±10.43	27.14±11.24	22.36±3.9	41.02±4.85	60.86±2.96	47.58±10.11	51.15±3.05	23.82±7.67	21.79±3.1	41.04±2.78
LwF	60.86±2.96	47.31±10.4	38.91±12.89	23.31±13.46	28.4±2.83	39.76±6.85	60.86±2.96	46.98±8.32	50.77±3.35	33.48±2.7	29.69±2.91	44.36±2.2
ICaRL	50.85±6.51	52.21±2.72	37.39±6.78	31.33±6.31	28.85±5.04	40.13±4.1	50.85±6.51	52.06±2.66	42.45±6.48	32.89±4.96	34.7±3.93	42.59±2.8
KCN	60.86±2.96	56.38±5.03	47.56±10.41	38.62±9.47	37.05±7.11	48.09±6.41	60.86±2.96	59.41±6.74	57.39±6.19	46.48±6.1	44.3±5.43	53.69±4.42
KT	53.16±2.25	42.55±2.33	33.93±2.97	38.48±8.66	31.27±9.34	39.88±3.84	53.16±2.25	59.12±1.78	50.02±5.13	49.02±5.34	28.54±2.95	47.97±2.67
EMP	54.78±1.49	40.49±1.9	24.32±3.37	27.15±8.46	22.53±6.02	33.85±2.96	54.78±1.49	37.28±7.37	19.6±4.96	34.69±4.76	24.19±6.62	34.11±3.48
HANet(Ours)	**61.16±2.29**	**63.07±3.09**	**57.5±5.98**	**53.21±4.64**	54.31±3.21	**57.85±2.91**	**61.16±2.29**	**66.84±2.88**	**64.68±3.77**	58.02±6.58	54.37±5.94	61.02±3.46

Table 2: $F1_{micro}$ of every sub-task and $\bar{F1}_{micro}$ across all sub-tasks on 2-way ACE benchmark.

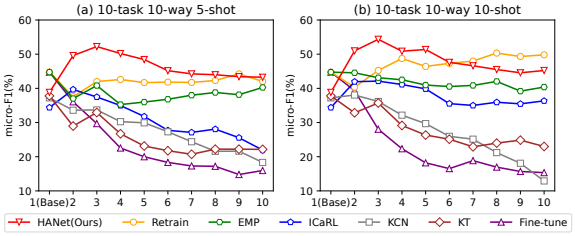

Figure 5: $F1_{micro}$ performance of each sub-task in Larger MAVEN benchmark.

4.7. Effect of Augmentation Method in Contrastive Augmentation

Different augmentation methods affect contrastive augmentation. We evaluate "Dropout", "Shuffle", and "Random Token Replacement" ("RTR"). As mentioned in Gao et al. (2021), "Dropout" means making a forward pass with dropout modules. "Shuffle" randomly shuffle the sentence. "RTR" refers to randomly replacing non-trigger tokens with other tokens. From Table 5, we can draw the following conclusion: In most cases, "Shuffle" is the most effective method. "Dropout" performs worse than the others, however, it still outperforms "w/o CA".

4.8. Evaluation in Extreme Scenarios

To validate the effeciveness of our method in various CFED applications, we conduct experiments to investigate on extreme conditions with more incremental tasks and fewer shot numbers. . **Larger CFED Task.** We exploit MAVEN benchmark to select 100 most frequent types to conduct 10-task 10-way task. From the results in Figure 5, we conclude that existing methods can not generalize well to larger CFED, meanwhile, HANet still maintains the best performance, showcasing strong continual learning ability in more practical situations.

Continual Fewer-shot Event Detection Task. To explore the minimum samples from which models can learn to maintain good performance, we perform 2-way 1-shot and 2-way 2-shot experimental settings. According to Table 6 and 7, our method outperforms other baselines, proving the ability to better utilize few-shot samples in severe conditions when dealing with CFED tasks.

4.9. Capability of LLM in Solving Continual Few-shot Event Detection

Recently, there have been growing discussions (Chen et al., 2023; Wang et al., 2023) about the capabilities of Large Language Models (LLMs) on IE tasks. Though these LLMs demonstrate promising abilities to learn from few-shot samples, their performance on continual few-shot event detection is to be discussed. In this section, we aim to evaluate the capability of ChatGPT in CFED settings. We conduct comparisons with gpt-3.5-turbo[2].

Following Event Extraction Trigger instructions by Wang et al. (2023) to perform in-context learning in gpt-3.5-turbo (Ouyang et al., 2022), we use few-shot samples as instructions selected from the training set. The original training set in T_1 contains 100 samples, we randomly select 1 or 2 samples every time a new test sample arrives. Specifically, at stage C_t, we conduct evaluations in C_t^{test} by providing few-shot samples of each type in C_t^{train}. Detailed instructions and cases of gpt-3.5-turbo are shown in Appendix A.

From the results illustrated in Table 4, we can observe that, compared with gpt-3.5-turbo failed to perform well on continual few-shot event detection tasks. Our method outperforms gpt-3.5-turbo significantly.

[2] https://api.openai.com/v1/chat/completions

Method	2-way 5-shot						2-way 10-shot					
	1	2	3	4	5	$\bar{F}1_{micro}$	1	2	3	4	5	$\bar{F}1_{micro}$
HANet(Ours)	**67.16**	**56.01**	**54.80**	**54.89**	**55.22**	**57.62**	**67.16**	**54.22**	**58.31**	**56.90**	**58.09**	**58.94**
w/o Replay*	**67.16**	51.02	44.15	38.76	36.78	47.57	**67.16**	48.13	48.14	41.07	40.01	48.90
w/o Distill	**67.16**	46.83	42.77	37.17	42.90	47.37	**67.16**	45.45	44.07	44.90	47.77	49.87
w/o PA	**67.16**	54.28	53.01	50.98	52.21	55.53	**67.16**	52.94	57.47	53.91	55.38	57.37
w/o CA	59.67	54.45	49.14	50.08	49.57	52.58	59.67	53.31	53.75	53.16	53.46	54.67
w/o PA and CA	59.67	51.43	43.32	44.32	42.04	48.16	59.67	45.03	45.90	47.14	48.35	49.22

Table 3: We perform ablation studies, comparing $F1_{micro}$ by removing each component at a time.

Benchmark	Method	2-way 1-shot						2-way 2-shot					
		1	2	3	4	5	$\bar{F}1_{micro}$	1	2	3	4	5	$\bar{F}1_{micro}$
MAVEN	HANet(Ours)	**67.16**	45.54	38.28	**42.39**	**40.40**	**46.75**	**67.16**	55.87	**50.35**	**51.63**	**51.39**	**55.28**
	gpt-3.5-turbo	54.22	**55.25**	**41.60**	37.88	33.31	44.45	57.00	**58.51**	43.64	40.39	36.56	47.22
ACE	HANet(Ours)	**60.99**	**51.93**	**41.67**	41.54	**35.84**	**46.40**	**60.99**	**58.38**	39.48	41.76	**44.60**	**49.04**
	gpt-3.5-turbo	42.20	50.29	40.51	**43.46**	35.21	42.33	56.36	49.72	**45.16**	**44.44**	42.96	47.73

Table 4: Comparison with gpt-3.5-turbo on MAVEN and ACE benchmark.

Way-num	Method	MAVEN		ACE	
		5-shot	10-shot	5-shot	10-shot
2way	w/o CA	52.58	54.67	48.27	60.45
	Dropout	54.68	56.32	53.06	61.87
	Shuffle	**57.62**	**58.94**	55.10	**63.98**
	RTR	54.60	56.57	**55.53**	63.27
	Retrain	51.78	54.93	49.54	60.69
4way	w/o CA	45.68	48.95	64.66	68.70
	Dropout	44.36	47.45	67.41	68.58
	Shuffle	**48.47**	**49.91**	**70.31**	**69.90**
	RTR	46.18	47.96	67.93	68.11
	Retrain	42.53	46.59	65.21	68.65

Table 5: $\bar{F}1_{micro}$ of different augmentation methods on MAVEN and ACE benchmarks. We also list the "w/o CA" and Retrain method for comparison.

5. Related Work

5.1. Traditional Event Detection

Impressive progress has been made in research related to traditional event detection by neural network-based methods (Chen et al., 2015; Nguyen and Grishman, 2015; Liu et al., 2017; Chen et al., 2018; Lu et al., 2019). These approaches greatly improved the performance on the ideal ED task. Nevertheless, they face considerable catastrophic forgetting and few-shot overfitting when handling continual event types with few samples, which seriously restricts their real-world applications.

5.2. Continual Event Detection

The major challenge of Continual ED is to learn emerging tasks while avoiding forgetting previous tasks (McCloskey and Cohen, 1989; Ring, 1994; Thrun and Mitchell, 1995; Thrun, 1998). Cao et al. (2020) construct a replay-distillation method to preserve knowledge from memory set and previous models. Besides replay and distillation, Yu et al. (2021) utilize an initialization method to transfer knowledge. Liu et al. (2022) adopt prompt learning for preserving previous knowledge. Although these works perform well on Continual ED, their abilities are limited with few-shot samples.

5.3. Few-shot Event Detection

Few-shot event detection aims to learn great representations with insufficient samples. Lai et al. (2020) propose two matching losses to provide cluster signals for few-shot learning. Deng et al. (2020) introduce a prototypical network with dynamic memory. Zhang et al. (2022a) design a hybrid contrastive learning approach. Zhao et al. (2022) align event types to FrameNet to obtain more instances for prototype calculation. Since these methods only concentrate on few-shot tasks with fixed types, they dismiss the continual situation.

6. Conclusions

In this paper, we focus on a more realistic yet challenging scenario of continual few-shot event detection, where the system is required to detect and classify events on continually emerging new types with limited labeled data. We propose a **H**ierarchical **A**ugmentation **Net**work (**HANet**). To alleviate catastrophic forgetting in memorizing previous event types, we incorporate prototypical augmentation to preserve previous knowledge with limited exemplars. We also devise a contrastive augmentation module to tackle with overfitting when learning new event types. This module leverages valuable token information from limited samples in incremental tasks. We conduct a series of experiments to show that our model perform well on continual few-shot event detection tasks, achieving state-of-the-art performance compared with previous baselines and ChatGPT.

Method	2-way 1-shot						2-way 2-shot					
	1	2	3	4	5	$\bar{F}1_{micro}$	1	2	3	4	5	$\bar{F}1_{micro}$
Fine-tune	59.67	26.81	28.34	22.96	18.79	31.31	59.67	56.17	41.67	33.13	22.81	42.69
Retrain	59.67	42.34	33.33	29.04	28.25	38.53	59.67	44.68	37.73	38.70	40.98	44.35
EWC	59.67	35.95	28.22	15.79	16.17	31.16	59.67	55.68	47.96	36.10	26.92	45.27
LwF	59.67	5.28	24.63	27.11	30.82	29.50	59.67	36.72	34.07	28.94	28.71	37.62
ICaRL	52.29	36.71	34.18	31.06	25.77	36.00	52.29	41.38	34.44	33.47	29.19	38.15
KCN	59.67	39.10	**43.19**	41.97	38.18	44.42	59.67	54.40	**50.67**	49.98	47.58	52.46
KT	54.32	5.94	5.78	3.70	3.61	14.67	54.32	35.22	32.71	27.47	28.23	35.59
EMP	57.21	4.95	5.53	5.42	5.29	15.68	57.21	18.28	6.84	7.06	8.43	19.56
HANet(Ours)	**67.16**	**45.54**	38.28	**42.39**	**40.40**	**46.75**	**67.16**	**55.87**	50.35	**51.63**	**51.39**	**55.28**

Table 6: 2-way Continual Fewer-shot Event Detection Task in MAVEN benchmark.

Method	2-way 1-shot						2-way 2-shot					
	1	2	3	4	5	$\bar{F}1_{micro}$	1	2	3	4	5	$\bar{F}1_{micro}$
Fine-tune	57.75	52.97	26.47	15.87	3.50	31.31	57.75	49.38	26.01	22.52	29.71	37.07
Retrain	57.75	43.16	30.16	31.69	28.36	38.22	57.75	48.91	33.86	36.97	35.01	42.50
EWC	57.75	45.09	25.37	16.18	6.51	30.18	57.75	50.60	23.87	13.90	25.46	34.32
LwF	57.75	37.50	18.31	7.97	6.37	25.58	57.75	44.00	16.72	16.29	29.45	32.84
ICaRL	54.68	45.96	27.08	25.29	22.34	35.07	54.68	43.81	32.89	33.12	28.49	38.60
KCN	57.75	54.13	40.71	**43.97**	26.52	44.61	57.75	51.37	36.83	34.66	40.40	44.20
KT	51.90	1.47	1.36	1.14	1.51	11.48	51.90	40.19	24.03	24.20	20.81	32.22
EMP	56.10	1.77	3.59	3.59	3.70	13.75	56.10	34.11	16.57	3.62	15.09	25.10
HANet(Ours)	**60.99**	**51.93**	**41.67**	41.54	**35.84**	**46.40**	**60.99**	**58.38**	**39.48**	**41.76**	**44.60**	**49.04**

Table 7: 2-way Continual Fewer-shot Event Detection Task in ACE benchmark.

7. Limitations

Though performing well on the CFED task, there are still some limitations to be mentioned: (1) Our method focuses on a fixed emerging number of event types and the shot number of each few-shot task is unchanging, which is still ideal in real-world scenarios. (2) Though we propose space augmentation for prototypes in memory, the approach still requires extra storage space, which limits its application in some extreme scenarios. (3) Since our method performs well for event detection, it has the potential to explore the possibility of extending our approach to other IE applications (e.g., Relation Extraction and Named Entity Recognition). We leave this as future work.

8. Acknowledgements

This work is supported by the National Key Research and Development Program of China (No. 2022ZD0160503), and the National Natural Science Foundation of China (No. 62176257). This work is also supported by the Youth Innovation Promotion Association CAS, and Yunnan Provincial Major Science and Technology Special Plan Projects (No.202202AD080004).

9. Bibliographical References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part III*, page 144–161, Berlin, Heidelberg. Springer-Verlag.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Pengfei Cao, Yubo Chen, Jun Zhao, and Taifeng Wang. 2020. Incremental event detection via

knowledge consolidation networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 707–717, Online. Association for Computational Linguistics.

Tianqi Chen, Ian J. Goodfellow, and Jonathon Shlens. 2016. Net2net: Accelerating learning via knowledge transfer. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Xuanting Chen, Junjie Ye, Can Zu, Nuo Xu, Rui Zheng, Minlong Peng, Jie Zhou, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. How robust is gpt-3.5 to predecessors? a comprehensive study on language understanding tasks. *arXiv preprint arXiv:2303.00293*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Yubo Chen, Hang Yang, Kang Liu, Jun Zhao, and Yantao Jia. 2018. Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1267–1276, Brussels, Belgium. Association for Computational Linguistics.

Shumin Deng, Ningyu Zhang, Jiaojian Kang, Yichi Zhang, Wei Zhang, and Huajun Chen. 2020. Meta-learning with dynamic-memory-based prototypical network for few-shot event detection. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 151–159.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie M Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *Lrec*, volume 2, pages 837–840. Lisbon.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

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

Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. 2019. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zixuan Ke, Bing Liu, Hu Xu, and Lei Shu. 2021. CLASSIC: Continual and contrastive learning of aspect sentiment classification tasks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6871–6883, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Viet Dac Lai, Thien Huu Nguyen, and Franck Dernoncourt. 2020. Extensively matching for few-shot learning event detection. In *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events*, pages 38–45, Online. Association for Computational Linguistics.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Minqian Liu, Shiyu Chang, and Lifu Huang. 2022. Incremental prompting: Episodic memory prompt for lifelong event detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2157–2165, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2019. Distilling discrimination and generalization knowledge for event detection via delta-representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4366–4376, Florence, Italy. Association for Computational Linguistics.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. SKILL: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States. Association for Computational Linguistics.

Yutao Mou, Keqing He, Yanan Wu, Zhiyuan Zeng, Hong Xu, Huixing Jiang, Wei Wu, and Weiran Xu. 2022. Disentangled knowledge transfer for OOD intent discovery with unified contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 46–53, Dublin, Ireland. Association for Computational Linguistics.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

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

Mark Bishop Ring. 1994. *Continual Learning in Reinforcement Environments*. Ph.D. thesis, University of Texas at Austin, USA. UMI Order No. GAX95-06083.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Sebastian Thrun. 1998. Lifelong learning algorithms. *Learning to learn*, 8:181–209.

Sebastian Thrun and Tom M Mitchell. 1995. Lifelong robot learning. *Robotics and autonomous systems*, 15(1-2):25–46.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Pengfei Yu, Heng Ji, and Prem Natarajan. 2021. Lifelong event detection with knowledge transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5278–5290, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruihan Zhang, Wei Wei, Xian-Ling Mao, Rui Fang, and Dangyang Chen. 2022a. HCL-TAT: A hybrid contrastive learning method for few-shot event detection with task-adaptive threshold. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1808–1819, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuwei Zhang, Haode Zhang, Li-Ming Zhan, Xiao-Ming Wu, and Albert Lam. 2022b. New intent discovery with pre-training and contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 256–269, Dublin, Ireland. Association for Computational Linguistics.

Kailin Zhao, Xiaolong Jin, Long Bai, Jiafeng Guo, and Xueqi Cheng. 2022. Knowledge-enhanced self-supervised prototypical network for few-shot event detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6266–6275, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

10. Language Resource References

Walker, Christopher and Strassel, Stephanie and Medero, Julie and Maeda, Kazuaki. 2006. *ACE 2005 Multilingual Training Corpus* . Linguistic Data Consortium (LDC), ISLRN 458-031-085-383-4. PID https://catalog.ldc.upenn.edu/LDC2006T06.

Wang, Xiaozhi and Wang, Ziqi and Han, Xu and Jiang, Wangyi and Han, Rong and Liu, Zhiyuan and Li, Juanzi and Li, Peng and Lin, Yankai and Zhou, Jie. 2020. *MAVEN: A Massive General Domain Event Detection Dataset*. Association for Computational Linguistics. PID https://aclanthology.org/2020.emnlp-main.129.

Appendix A. Instructions for large language models

In this section, we show gpt-3.5-turbo's instructions and cases for the continual few-shot event detection task in Figure 6 and Figure 7. When learning new event types, we simply append new options and examples for these types as in-context learning prompts.

Please tell me event type and its trigger word from given type options and few-shot examples. Output format is "type: trigger". Option: {type1}, {type2}
Examples:
Text: {Example for type 1}
Answer: {Ground truth for Text 1}
Text: {Example for type 2}
Answer: {Ground truth for Text 2}
Input:
Text: {Test sample}
Answer:

ChatGPT: \\ ChatGPT Response

Figure 6: Instructions of gpt-3.5-turbo for CFED

Please tell me event type and its trigger word from given type options and few-shot examples. Output format is "type: trigger". Option: Killing, Attack, Social_event, Catastrophe
Examples:
Text: throughout the island, damage totaled $ 2 million (1987 usd) and 10 people were killed.
Answer: Killing: killed
Text: Text: on 19 december 2013, both of the attackers were found guilty of rigby's murder.
Answer: Attack: attackers
Text: the show was divided into seven segments with the last one being the encore.
Answer: Social_event: show
Text: around 12, 500 red prisoners of war died of malnutrition and disease in camps.
Answer: Catastrophe: disease
Input:
Text: mexican casualties are unknown.
Answer:

Ground Truth: Killing: casualties

ChatGPT: Catastrophe: unknown

Please tell me event type and its trigger word from given type options and few-shot examples. Output format is "type: trigger". Option: Killing, Attack, Social_event, Catastrophe, Process_start, Motion
Examples:
Text: prime minister sandor wekerle resigned and former prime minister istvan tisza was murdered
Answer: Killing: murdered
Text: a united states report declared arif qasmani to be involved in the attack.
Answer: Attack: attack
Text: the show was divided into seven segments with the last one being the encore.
Answer: Social_event
Text: around 12, 500 red prisoners of war died of malnutrition and disease in camps.
Answer:Catastrophe: disease
Text: bombing of the trail system had begun on 14 december 1964 with the advent of operation barrel roll.
Answer:Process_start: begun;Attack: bombing
Text: when the danes left their camp he attacked, while the remaining rebels moved over the river.
Answer: Attack: attacked;Motion: moved
Input:
Text: the temperature in dallas that day reached 104 degrees fahrenheit, prompting many in the very crowded ground area to pass out and be lifted overhead to the indoor areas of the stadium where water fountains were at.
Answer:

Ground Truth: Motion: lifted

ChatGPT: Social_event: stadium

Figure 7: Cases of gpt-3.5-turbo for CFED