Lifelong Event Detection with Embedding Space Separation and Compaction

Anonymous ACL submission

Abstract

To mitigate forgetting, existing lifelong event detection methods typically maintain a memory module and replay the stored memory data during the learning of a new task. However, the simple combination of memory data and new-task samples can still result in substantial forgetting of previously acquired knowledge, which may occur due to the potential overlap between the feature distribution of new data and the previously learned embedding space. Moreover, the model suffers from overfitting on the few memory samples rather than effectively remembering learned patterns. To address the challenges of forgetting and overfitting, we propose a novel method based on embedding space separation and compaction. Our method alleviates forgetting of previously learned tasks by forcing the feature distribution of new data away from the previous embedding space. It also mitigates overfitting by a memory calibration mechanism that encourages memory data to be close to its prototype to enhance intra-class compactness. In addition, the learnable parameters of the new task are initialized by drawing upon acquired knowledge from the previously learned task to facilitate forward knowledge transfer. With extensive experiments, we demonstrate that our method can significantly outperform previous state-of-the-art approaches.

1 Introduction

Event detection (ED) aims to detect the event type of trigger words in a given sentence, *e.g.,* extracting the event type *injure* from the trigger word *scalded* in text "*He was scalded by hot water*". Traditional ED methods typically consider a fixed pre-defined set of event types (Chen et al., 2015; Nguyen et al., 2016; Huang and Ji, 2020). However, as the environment and data distributions change in real scenarios, the model might face challenges in handling rapidly emerging new event types.

A more practical setting is lifelong event detection or LED (Cao et al., 2020), where the model learns event knowledge from a sequence of tasks with different sets of event types. In LED, the model is expected to retain and accumulate knowledge when learning new tasks, which is challenging due to *catastrophic forgetting* (McCloskey and Cohen, 1989) of previously acquired knowledge. Existing methods (Cao et al., 2020; Yu et al., 2021) for mitigating forgetting in LED typically maintain a memory that saves a few key samples of previous tasks, which are then combined with new data for training. Recently, Liu et al. (2022) introduce Episodic Memory Prompts (EMP) that leverages soft prompts to remember learned event types, achieving state-of-the-art performance on LED.

Despite its effectiveness, EMP has two key limitations. First, simply combining new data and memory samples for training can still result in forgetting as the feature distribution of new data might overlap with the previously learned embedding space (see Appendix A.2). Second, it may overfit on a few memory samples after frequent replays rather than effectively retaining learned patterns.

To address the above limitations of EMP, in this paper, we introduce a novel method based on Embedding space Separation and COmpaction (ESCO) for LED. In particular, we propose a margin-based loss that forces the feature distribution of new event types away from the learned embedding space to alleviate *forgetting*. Inspired by Han et al. (2020), we introduce a memory calibration mechanism to encourage memory data to be close to its prototype to avoid *overfitting* on the few memory samples. In addition, the learnable parameters of the new task are initialized using those of the previously learned task to facilitate *forward knowledge transfer*, which is as important for lifelong learning as preventing forgetting (Ke et al., 2020). The empirical results show that our method significantly outperforms previous state-of-the-art approaches. In summary, our main contributions are:

- We propose ESCO, a novel method based on embedding space separation and compaction to mitigate forgetting and overfitting in LED.
- With extensive experiments and analysis, we demonstrate the effectiveness of our method compared to existing ones. Our code base is available at `<redacted>`.

2 Problem Formulation

LED involves learning from a stream of event detection tasks $\mathbb{T} = (\mathcal{T}^1, \ldots, \mathcal{T}^n)$, where each task $\mathcal{T}^k$ has its own training set $\mathcal{D}^k_{\text{train}}$, validation set $\mathcal{D}^k_{\text{valid}}$, and test set $\mathcal{D}^k_{\text{test}}$. For every input text x^i in $\mathcal{D}^k$, it contains a set of target spans $\{\overline{x}^i_t\}$ and their corresponding labels y^i_t which belong to the event type set $\mathcal{C}^k$ of task $\mathcal{T}^k$. Note that the event type sets of different tasks are non-overlapping.

After the training on $\mathcal{D}^k_{\text{train}}$, the model is expected to perform well on all the k tasks that it has learned and will be evaluated on the combined test set $\hat{\mathcal{D}}^k_{\text{test}} = \cup^k_{i=1}\mathcal{D}^i_{\text{test}}$ consisting of all known event types $\hat{\mathcal{C}}^k = \cup^k_{i=1}\mathcal{C}^i$. During the learning, a memory module $\mathcal{M}$ which stores a few key samples of previous tasks is maintained to overcome the forgetting problem.

3 Embedding Space Separation and Compaction

When learning a new task $\mathcal{T}^k$, following Liu et al. (2022), we first initialize a set of soft prompts $\mathcal{P}^k = \{p^k_1, ..., p^k_{|\mathcal{C}^k|}\}$ where $\mathcal{C}^k$ is the event type set of $\mathcal{T}^k$. The accumulated prompts $\mathcal{Q}^k = [\mathcal{P}^1, ..., \mathcal{P}^k]$ until $\mathcal{T}^k$ are then combined with the input text x^i to obtain the contextual representations using a frozen BERT (Devlin et al., 2019):

$$[\mathbf{x}^i, \mathbf{Q}^k] = \text{BERT}([x^i, \mathcal{Q}^k]) \qquad (1)$$

where $\mathbf{x}^i$ and $\mathbf{Q}^k$ are the representations of x^i and $\mathcal{Q}^k$, respectively. To facilitate *forward knowledge transfer*, we initialize soft prompts $\mathcal{P}^k$ of the new task using learned prompts $\mathcal{P}^{k-1}$ of the previous task. For the first task $\mathcal{T}^1$, we initialize each event type prompt p^1_i in $\mathcal{P}^1$ using its corresponding name.

To predict the event type of the span $\overline{x}^i_t$, we concatenate the representations corresponding to the start and end token and obtain the logits over all learned types through a feed-forward network (FFN) as well as a linear layer:

$$Z^i_t = \text{Linear}(\text{FFN}([\mathbf{x}^i_m, \mathbf{x}^i_n])) \qquad (2)$$

where $\overline{\mathbf{x}}^i_t = \text{FFN}([\mathbf{x}^i_m, \mathbf{x}^i_n])$ is the span representation, m and n denote the start and end index of the span, respectively. Following Liu et al. (2022), to entangle span representations with soft prompts, the probability distribution over all prompts is calculated as $Z_q = \text{FFN}(\mathbf{Q}^k) \cdot \overline{\mathbf{x}}^i_t$, where $\cdot$ is the inner product. Z_q is then combined with Z^i_t to optimize the cross entropy loss:

$$\mathcal{L}_{\text{new}} = - \sum_{(\overline{x}^i_t, y^i_t) \in \mathcal{D}^k_{\text{train}}} \text{CE}(Z^i_t + Z_q, y^i_t) \qquad (3)$$

After learning the previous task $\mathcal{T}^{k-1}$, we select the top-l most informative training examples for each event type in $\mathcal{C}^{k-1}$ using the herding algorithm (Welling, 2009), which are then saved in the memory module $\mathcal{M}$ for replay to mitigate forgetting. Similar as Eq. 3, the training objective for memory replay when learning $\mathcal{T}^k$ is:

$$\mathcal{L}_{\text{mem}} = - \sum_{(\overline{x}^i_t, y^i_t) \in \mathcal{M}} \text{CE}(Z^i_t + Z_q, y^i_t) \qquad (4)$$

However, the simple combination of $\mathcal{L}_{\text{new}}$ and $\mathcal{L}_{\text{mem}}$ can still result in substantial forgetting of acquired knowledge due to the potential overlap between the feature distribution of new event types and the previously learned embedding space (see Appendix A.2). To ensure that the new feature distribution is away from the learned embedding space, we design a *margin-based* loss, which decreases the similarity scores between new samples and prototypes (see Eq. 8 for the calculation of prototypes) of learned event types:

$$\mathcal{L}_{\text{sim}} = \sum_{(\overline{x}^i_t, y^i_t) \in \mathcal{D}^k_{\text{train}}} \sum_{\mathbf{e}_i \in \mathcal{E}^{k-1}} \max(0, g(\overline{\mathbf{x}}^i_t, \mathbf{e}_i) - m_1) \qquad (5)$$

where $\mathcal{E}^{k-1}$ is the prototype set of previous $k-1$ tasks, $g(,)$ is the similarity function (cosine similarity) and m_1 is the margin for $\mathcal{L}_{\text{sim}}$. Note that $\mathcal{L}_{\text{sim}}$ is different from metric learning or contrastive learning (Qin and Joty, 2022) which typically considers both positive and negative pairs. $\mathcal{L}_{\text{sim}}$ only includes negative pairs while ignoring positive ones as our goal in designing $\mathcal{L}_{\text{sim}}$ is to *separate* the new feature distribution and the learned embedding space.

As the size of memory $\mathcal{M}$ is typically small, the model is prone to overfit on the few memory samples after frequent replays, making learned distributions distorted. To effectively recover from distorted learned distributions, we introduce a *memory calibration* mechanism inspired by Han et al. (2020). Specifically, for each memory sample in $\mathcal{M}$, we encourage it to be close to its corresponding prototype to improve the intra-class compactness of learned distributions. More formally,

$$\mathcal{L}_{\text{cal}} = -\sum_{(\overline{x}_t^i, y_t^i) \in \mathcal{M}} \log \frac{\exp g(\overline{\mathbf{x}}_t^i, \mathbf{e}_l)}{\sum_{j=1}^{|\mathcal{E}^{k-1}|} \exp g(\overline{\mathbf{x}}_t^i, \mathbf{e}_j)} \tag{6}$$

where $\mathbf{e}_l$ is the prototype of y_t^i. The *total* loss for learning on $\mathcal{T}^k$ is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{new}} + \lambda_1 \mathcal{L}_{\text{sim}} + \lambda_2 (\mathcal{L}_{\text{mem}} + \mathcal{L}_{\text{cal}}) \tag{7}$$

where λ_1 and λ_2 are loss weights.

After learning $\mathcal{T}^k$ and selecting memory data for $\mathcal{T}^k$, we use the memory $\mathcal{M}$ to calculate prototypes of all learned event types in $\mathcal{C}^k$. Specifically, for each event type e_j in $\mathcal{C}^k$, we obtain its prototype $\mathbf{e}_j$ by averaging the span representations of all samples labeled as e_j in $\mathcal{M}$ as follows:

$$\mathbf{e}_j = \frac{1}{|\mathcal{M}_{e_j}|} \sum_{(\overline{x}_t^i, y_t^i) \in \mathcal{M}_{e_j}} \overline{\mathbf{x}}_t^i \tag{8}$$

where $\mathcal{M}_{e_j} = \{(\overline{x}_t^i, y_t^i) | (\overline{x}_t^i, y_t^i) \in \mathcal{M}, y_t^i = e_j\}$.

4 Experiment

4.1 Experimental Setup

We conduct experiments on two representative event detection datasets in our work: ACE05 (Doddington et al., 2004) and MAVEN (Wang et al., 2020). Following Liu et al. (2022), we divide each dataset into a sequence of 5 tasks with non-overlapping event type sets to form a *class-incremental* setting. After learning $\mathcal{T}^k$, the model is evaluated on the combined test set $\hat{\mathcal{D}}_{\text{test}}^k = \cup_{i=1}^k \mathcal{D}_{\text{test}}^i$ of all seen tasks. As the task order might influence the model performance, we run experiments for each dataset 5 times with different task order permutations and report the average results. More details of the experimental setup are in Appendix A.3.

4.2 Methods Compared

We compare our approach with the following methods: (1) **Fine-tuning** tunes the model only on new data without memory; (2) **BiC** (Wu et al., 2019) introduces a bias correction layer to improve lifelong learning performance; (3) **KCN** (Cao et al., 2020) designs prototype enhanced retrospection and hierarchical distillation to alleviate semantic ambiguity and class imbalance; (4) **KT** (Yu et al., 2021) proposes to transfer knowledge between related types; (5) **EMP** (Liu et al., 2022) leverages type-specific soft prompts to remember learned event knowledge; and (6) **Multi-task learning (MTL)** simultaneously trains the model on all data, serving as the *upper bound* in LED.

	MAVEN					ACE05				
Task index	1	2	3	4	5	1	2	3	4	5
Fine-tuning	63.51	39.99	33.36	23.83	22.69	58.30	43.96	38.02	21.53	25.71
BiC	63.51	46.69	39.15	31.69	30.47	58.30	45.73	43.28	35.70	30.80
KCN	63.51	51.17	46.80	38.72	38.58	58.30	54.71	52.88	44.93	41.10
KT	63.51	52.36	47.24	39.51	39.34	58.30	55.41	53.95	45.00	42.62
EMP	**67.50**	59.67	58.03	54.80	54.39	**58.35**	50.03	54.91	47.78	47.19
ESCO	**67.50**	**61.37**	**60.65**	**57.43**	**57.35**	**58.35**	**57.42**	**57.63**	**53.64**	**55.20**
MTL	—	—	—	—	68.42	—	—	—	—	67.22

Table 1: F1 score (%) of different methods at every time step on two datasets. 'MTL' stands for 'multi-task learning'. ESCO is significantly better than EMP with p-value < 0.05 (paired t-test). We report results with variance and detailed results for different task orders in Appendix A.4 and Appendix A.5, respectively. In addition, the comparison of computational resources between EMP and ESCO is shown in Appendix A.6.

4.3 Main Results

We report the F1 score of different methods at each time step in Table 1. From the results, we can observe that ESCO significantly outperforms previous baselines on both datasets, demonstrating its superiority. Simply fine-tuning the model on new data without memory replay results in poor performance due to severe forgetting of learned knowledge. Although BiC, KCN and KT could alleviate forgetting to some extent, there is still a large performance drop after learning all tasks. EMP achieves better performance because the type-specific soft prompts help retain previously acquired knowledge. However, it does not necessarily ensure large distances among feature distributions of different event types, and easily overfits on the memory samples. Our proposed ESCO outperforms EMP by a large margin through embedding space separation and compaction. To verify its effectiveness, we visualize the embedding spaces of EMP and ESCO on ACE05 in Fig. 1. Specifically, we randomly select 6 event types from different learning stages and visualize their test data using t-SNE (Van der Maaten and Hinton, 2008). The comparison demonstrates that ESCO could achieve larger inter-class distances

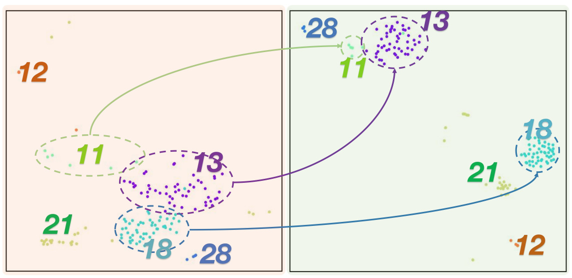

Figure 1: Comparison between the embedding spaces of EMP (left) and ESCO (right). Colors represent different event types with numbers being the event indexes. Compared with EMP, ESCO shows larger inter-class distances, *e.g.,* the distance between 13 and 18, and better intra-class compactness (circled regions).

and better intra-class compactness in the embedding space.

4.4 Ablation Study

To analyze the contribution of different components of ESCO, we conduct several ablations. Specifically, we investigate three variants of ESCO: (*a*) without the margin-based loss (*w.o.* margin), (*b*) removing the memory calibration mechanism (*w.o.* calibration), and (*c*) without forward knowledge transfer (FKT) (*w.o.* FKT). The results of different ablations after learning all tasks are reported in Table 2. We can see that all components contribute to overall performance. The margin-based loss yields about **1.25**% performance boost as it can bring feature distribution of new data away from the learned embedding space. The memory calibration mechanism improves the F1 score by **1.47**%, which demonstrates the necessity of improving intra-class compactness of learned distributions. The adoption of forward knowledge transfer leads to **0.80**% improvement, indicating that it can indeed transfer useful learned knowledge to facilitate the learning of new tasks.

4.5 Further Analysis

Quantify Knowledge Transfer. Following Lopez-Paz and Ranzato (2017), we report the backward transfer (BWT) and forward transfer (FWT) of EMP and ESCO after learning all tasks in Table 3. From the comparison, we can observe that ESCO outperforms EMP by a large margin in terms of BWT and FWT on both datasets, demonstrating its effectiveness.

In addition, we show the effect of memory size, results of different memory sample selection methods, the comparison with the contrastive loss in Qin and Joty (2022), and a case study of the model output in Appendix A.7 ∼ A.10, respectively.

Method	MAVEN	ACE05	Average
ESCO	**57.35**	**55.20**	**56.28**
w.o. margin	55.92	54.13	55.03
w.o. calibration	55.76	53.85	54.81
w.o. FKT	56.58	54.38	55.48

Table 2: F1 score (%) of different ablations after learning all tasks: (i) without the margin-based loss, (ii) without the memory calibration mechanism, and (iii) without forward knowledge transfer. All components improve the performance of our method.

Dataset	MAVEN		ACE05	
	BWT	FWT	BWT	FWT
EMP	-10.4	-2.9	-16.8	0.5
ESCO	**-6.3**	**-1.2**	**-7.5**	**4.1**

Table 3: Backward transfer (BWT) and forward transfer (FWT) of EMP and ESCO after learning all tasks on MAVEN and ACE05.

5 Related Work

Lifelong event detection (LED) aims to continually learn from a sequence of event detection tasks with different sets of event types. Cao et al. (2020) propose KCN which addresses the semantic ambiguity and data imbalance problems in LED by prototype enhanced retrospection and hierarchical distillation. KT (Yu et al., 2021) encourages bi-directional knowledge transfer between old and new event types. Liu et al. (2022) introduce EMP to retain previously learned task-specific event knowledge through soft prompts. In contrast to previous works, we innovate on the methodology by imposing further constraints in the embedding space to mitigate forgetting and overfitting.

6 Conclusion

In this work, we have introduced embedding space separation and compaction (ESCO) for lifelong event detection (LED). ESCO imposes novel feature constraints in the embedding space to alleviate forgetting and overfitting problems. It initializes the learnable parameters for the new task by inheriting those from the previously learned task to facilitate forward knowledge transfer. With extensive experiments and analysis, we have demonstrated that ESCO significantly outperforms previous methods. For future work, we are interested in exploring ESCO in a meta-learning paradigm for LED.

References

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.

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.

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 Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Xu Han, Yi Dai, Tianyu Gao, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2020. Continual relation learning via episodic memory activation and reconsolidation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6429–6440, Online. Association for Computational Linguistics.

Lifu Huang and Heng Ji. 2020. Semi-supervised new event type induction and event detection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 718–724, Online. Association for Computational Linguistics.

Zixuan Ke, Bing Liu, and Xingchang Huang. 2020. Continual learning of a mixed sequence of similar and dissimilar tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 18493–18504. Curran Associates, Inc.

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.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30.

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.

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.

Chengwei Qin and Shafiq Joty. 2022. Continual few-shot relation learning via embedding space regularization and data augmentation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2776–2789, Dublin, Ireland. Association for Computational Linguistics.

Qing Sun, Fan Lyu, Fanhua Shang, Wei Feng, and Liang Wan. 2022. Exploring example influence in continual learning. In *Advances in Neural Information Processing Systems*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Xiaozhi Wang, Ziqi Wang, Xu Han, Wangyi Jiang, Rong Han, Zhiyuan Liu, Juanzi Li, Peng Li, Yankai Lin, and Jie Zhou. 2020. MAVEN: A Massive General Domain Event Detection Dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1652–1671, Online. Association for Computational Linguistics.

Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1121–1128.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382.

Method	Avg Time (hour)
EMP	8.2
ESCO	8.4

Table 4: The comparison of the average running time (Avg Time) between EMP and ESCO.

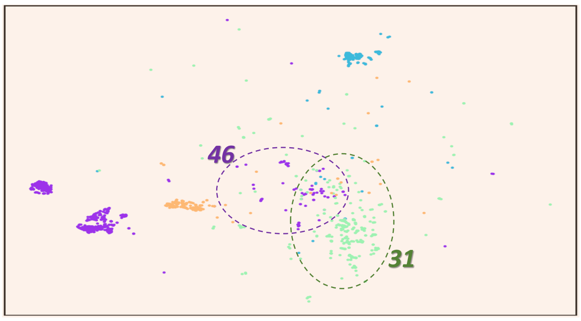

Figure 2: Overlap between feature distributions of event types at different learning stages, *e.g.,* circled regions. Colors represent different event types with numbers being the event indexes.

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.

A Appendix

A.1 Limitations

Although effective, ESCO also has some limitations. For example, ESCO mainly focuses on the setting where each task has enough training data. We leave how to explore lifelong event detection in few-shot settings as future work.

A.2 Overlap of Feature Distributions

Fig. 2 shows an example of the overlap between feature distributions of event types at different learning stages.

A.3 Implementation Details

All methods are implemented with PyTorch/Transformers library (Wolf et al., 2020). For hyperparameters, we mainly follow the settings in Liu et al. (2022) to have a fair comparison. We adopt -0.1 for the margin value m_1 so that the similarity score between a new sample and the prototype of a previously learned event type could be optimized to a negative number, *i.e.,* large inter-class distance. We set the weight λ_1 to 0.1 so that its corresponding loss $\mathcal{L}_{\text{sim}}$ has roughly the same order of magnitude as other losses. The loss weight λ_2 is set to $\frac{s}{k+s}$, where k is the number of target spans in the current batch and s is equal to 50 following Liu et al. (2022). For each task, we train the model for 20 epochs with early stopping.

For the state-of-the-art EMP (Liu et al., 2022), we reproduce the results using its open-source code and the same environment. For our method, we use the same environment and shared hyperparameters as EMP. For other baselines, we reuse the results in Liu et al. (2022). There are mainly two reasons: (*a*) They perform much worse than EMP, *i.e.,* they are not primary comparison approaches in our work; and (*b*) EMP reports different baseline results from Yu et al. (2021), indicating different settings. However, EMP does not provide details on how to obtain baseline results. As we use the same setting as EMP, we decide to reuse its results for other baselines.

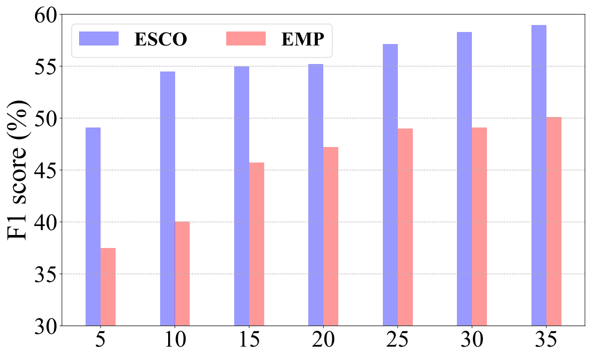

Figure 3: The performance of ESCO and EMP with different memory sizes.

A.4 Results with Variance

We show results with variance for EMP and ESCO in Table 5.

A.5 Detailed Results for Different Task Orders

Table 6 reports detailed results for different task orders.

A.6 Comparison of Computational Resources

We present the average running time of EMP and ESCO in Table 4. The comparison demonstrates that ESCO can outperform EMP by a large margin with a negligible increase in computational resources.

A.7 Effect of Memory Size

Following Liu et al. (2022), we select 20 samples as memory data for each event type. To inves-

	MAVEN					ACE05				
Task index	1	2	3	4	5	1	2	3	4	5
EMP	**67.50**$_{\pm 3.54}$	59.67$_{\pm 2.74}$	58.03$_{\pm 1.44}$	54.80$_{\pm 0.95}$	54.39$_{\pm 0.82}$	**58.35**$_{\pm 6.92}$	50.03$_{\pm 18.18}$	54.91$_{\pm 9.19}$	47.78$_{\pm 2.57}$	47.19$_{\pm 8.53}$
ESCO	**67.50**$_{\pm 3.54}$	**61.37**$_{\pm 2.92}$	**60.65**$_{\pm 1.85}$	**57.43**$_{\pm 0.81}$	**57.35**$_{\pm 0.66}$	**58.35**$_{\pm 6.92}$	**57.42**$_{\pm 13.56}$	**57.63**$_{\pm 6.26}$	**53.64**$_{\pm 4.41}$	**55.20**$_{\pm 4.16}$

Table 5: F1 score (%) and variance of EMP and ESCO at every time step on two datasets.

Task Order	MAVEN					ACE05				
	1	2	3	4	5	1	2	3	4	5
1	**70.80**	62.19	60.00	56.11	54.95	**62.58**	65.60	**67.20**	45.06	39.07
	70.80	**64.73**	**62.22**	**57.26**	**57.67**	**62.58**	**68.35**	66.02	**54.13**	**49.33**
2	**66.06**	56.01	56.16	54.07	54.91	**49.17**	50.99	57.78	51.76	42.49
	66.06	**58.36**	**58.16**	**57.71**	**57.40**	**49.17**	**52.13**	**60.73**	**57.85**	**54.76**
3	**70.80**	59.91	57.43	55.53	55.02	**62.58**	59.43	50.08	46.27	55.12
	70.80	**62.70**	**60.03**	**58.27**	**57.57**	**62.58**	**61.03**	**51.96**	**47.95**	**59.97**
4	**67.42**	62.35	58.74	54.10	53.19	**64.68**	55.15	56.87	47.30	41.65
	67.42	**62.81**	**62.72**	**56.12**	**56.20**	**64.68**	**69.12**	**58.44**	**57.78**	**53.67**
5	**62.39**	57.90	57.81	54.20	53.88	**52.74**	18.99	42.62	48.53	57.64
	62.39	**58.22**	**60.09**	**57.78**	**57.88**	**52.74**	**36.49**	**51.01**	**50.46**	**58.29**

Table 6: F1 score (%) of 5 runs with different task orders on two datasets. For every order, the upper row shows the performance of EMP and the lower row is the result of ESCO.

	EMP	ESCO
Herding algorithm	54.4	**57.1**
Example influence	53.5	**56.4**

Table 7: F1 score (%) of EMP and ESCO with different memory sample selection approaches.

	ESCO	ESCO$_{\text{con}}$
F1 score (%)	**57.1**	56.6

Table 8: Performance comparison between ESCO and ESCO$_{\text{con}}$.

tigate whether different memory sizes influence the performance gain of ESCO, we conduct controlled experiments on ACE05 with memory size $\{5, 10, 15, 25, 30, 35\}$. The performance comparison between ESCO and EMP is shown in Fig. 3. We can observe that ESCO consistently outperforms EMP by a large margin with different memory sizes, demonstrating its robustness.

A.8 Different Memory Sample Selection Approaches

Following EMP (Liu et al., 2022), we use the herding algorithm (Welling, 2009) to select memory samples. To validate whether different memory sample selection approaches influence the performance gain of ESCO, we replace the herding algorithm of EMP and ESCO with *example influence* (Sun et al., 2022) for memory selection. We randomly select three sequences for experiments and report the performance comparison between EMP and ESCO in Table 7. We can see that: (*a*) ESCO consistently outperforms EMP in different cases, demonstrating its effectiveness; and (*b*) herding algorithm performs better than example influence, justifying our choice.

A.9 Comparison with the Contrastive Loss

As mentioned in §3, our designed margin-based loss $\mathcal{L}_{\text{sim}}$ is different from the contrastive loss as $\mathcal{L}_{\text{sim}}$ only includes negative pairs while ignoring positive ones. To further demonstrate its superiority, we replace it with the contrastive loss in (Qin and Joty, 2022), namely ESCO$_{\text{con}}$. We use the same sequences as Appendix A.8 for experiments and report the results of ESCO and ESCO$_{\text{con}}$ in Table 8, which verify the effectiveness of $\mathcal{L}_{\text{sim}}$.

A.10 Case Study

We select MAVEN as a representative task and show several example outputs in Table 9. Compared with EMP, ESCO is able to retain more precise and fine-grained event knowledge, *e.g.,* ESCO can successfully detect the event type *Escaping*

Fighting continued until 9 May, when the Red Army entered the nearly liberated city.	
Label	*Arriving*
EMP	*Becoming_a_member*
ESCO	*Arriving*
In Japan, hundreds of people evacuated from mudslide-prone areas.	
Label	*Escaping*
EMP	*Removing*
ESCO	*Escaping*

Table 9: Output examples of EMP and ESCO. We color target spans in blue, correct outputs in green, and wrong outputs in red.

from the target span *evacuated* while EMP is confused by another semantically similar type *Removing*.