Lifelong Event Detection with Knowledge Transfer

Pengfei Yu[1], **Heng Ji**[2], **Premkumar Natarajan**[2]
[1]University of Illinois Urbana-Champaign [2]Amazon Alexa AI
pengfei4@illinois.edu, {jihj,premknat}@amazon.com

Abstract

Traditional supervised Information Extraction (IE) methods can extract structured knowledge elements from unstructured data, but they are limited to a pre-defined target ontology. In reality, the ontology of interest may change over time, adding emergent new types or more fine-grained subtypes. We propose a new lifelong learning framework to address this challenge. We focus on lifelong event detection as an exemplar case and propose a new problem formulation that is also generalizable to other IE tasks. In event detection and more general IE tasks, rich correlations or semantic relatedness exist among hierarchical knowledge element types. In our proposed framework, knowledge is being transferred between learned old event types and new event types. Specifically, we update old knowledge with the mentions of new event types using a self-training loss. In addition, we aggregate the representations of old event types based on their similarities with new event types to initialize the representations of new event types. Experimental results show that our framework outperforms competitive baselines with a 5.1% absolute gain in the F1 score. Moreover, our proposed framework can boost the F1 score for over 30% absolute gain on some new long-tail rare event types with few training instances. Our knowledge transfer module improves the performance on both learned event types and new event types under the lifelong learning setting, showing that it helps consolidate old knowledge and improve novel knowledge acquisition. [1]

1 Introduction

The Information Extraction (IE) task aims to extract informative knowledge elements (e.g., entities, relations, and events) from natural language. In practice, we usually extract knowledge elements for a pre-defined ontology consisting of various types of knowledge elements of interest. In this setting, IE is often formulated as a classification problem over types in the ontology, with an additional `Not-Any` (`NA`) type to identify text spans that don't belong to any ontology types (negative instances). Ontology-based supervised IE methods such as (Lin et al., 2020) can produce more accurate and structured results with annotated training data than open-domain IE (Yates et al., 2007) while being limited to the ontology. However, the ontology of interest may change over time, adding emergent new types for various knowledge elements (e.g., change *bombing* to *disease outbreak*), or adding more fine-grained subtypes for some existing general types (e.g., break *justice* into *acquit*, *arrest*, *charge*, *convict*, *release*, *sentence*, and *trial*). In the past twenty-five years the IE community has been shifting from one old ontology to a new one once every five years based on the consumers' needs, under many shared tasks, including MUC (Grishman and Sundheim, 1996), ACE[2], TAC-KBP (Ji et al., 2011), DARPA AIDA[3] and DARPA KAIROS[4]. When we face a new ontology, we need to annotate a new training set for new types and retrain a new system on the new ontology while discarding the old established system for the old ontologies. In contrast, we propose a new and more economic paradigm, *Lifelong Event Detection*, which can combine old system and new resources in a never-ending continual learning fashion. We show an example of lifelong event detection in Figure 1.

Such a paradigm is commonly referred as *Lifelong Learning*, *Continual Learning* or *Incremental Learning* (Ring, 1995; Thrun, 1998). We formu-

[1]The source code is available at https://github.com/Perfec-Yu/Lifelong-ED.

[2]https://en.wikipedia.org/wiki/Automatic_content_extraction

[3]https://www.darpa.mil/program/active-interpretation-of-disparate-alternatives

[4]https://www.darpa.mil/program/knowledge-directed-artificial-intelligence-reasoning-over-schemas

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5278–5290
November 7–11, 2021.

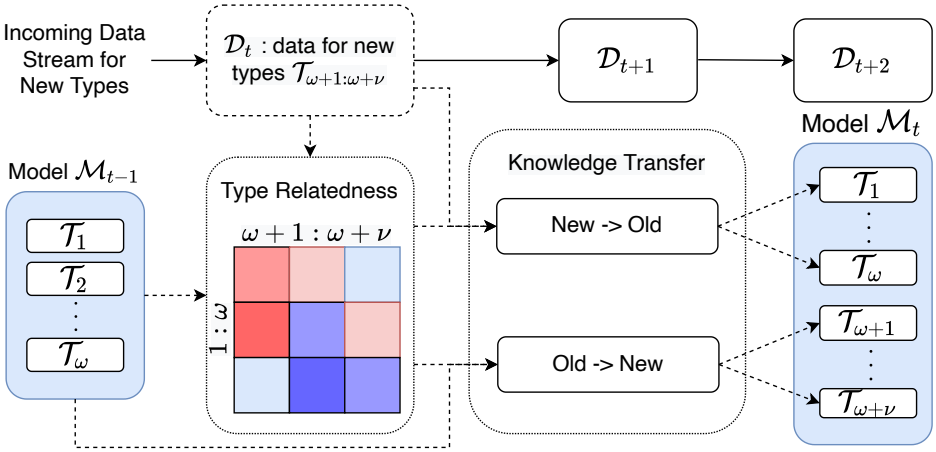

Figure 1: Illustration of lifelong event detection and our proposed knowledge transfer framework. The model continually learns from new data $\mathcal{D}_t$ containing mentions for new event types. After each training stage t, the model needs to detect mentions of all learned event types. We propose a novel knowledge transfer module to leverage rich connections between learned types and new types.

late lifelong event detection by modifying the commonly studied class incremental lifelong learning, where the model incrementally learns to classify more classes with only positive instances. We add a special `NA` type denoting negative instances that are not event triggers for any types. For example, *injured* in the sentence *"Bob is injured."* is a mention of an *injure* event, and *is* is a negative instance. Whenever training on new event types, the new training data includes instances of new types and negative instances. Lifelong event detection differs from class incremental learning, where the new data only contains instances for the new classes.

Another challenge in lifelong event detection is the naturally imbalanced distribution of event types in natural language as shown in Figure 2. Existing methods (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2019; Hou et al., 2019) for class incremental learning usually study relatively balanced classification datasets, and previous attempts (Nguyen et al., 2016; Cao et al., 2020) on incremental learning of event detection ignore this problem by only experimenting on frequent types.

In light of the challenges above in lifelong event detection, we propose a knowledge transfer framework taking advantage of the rich connections between various types such as contextual and semantic similarity. For instance, mentions for both the `Trial` and the `Charge` events contain court entities and crime-related content frequently in context, and their respective typical triggers such as *try* and *charge* usually have similar word embeddings representing their semantics. In our proposed lifelong

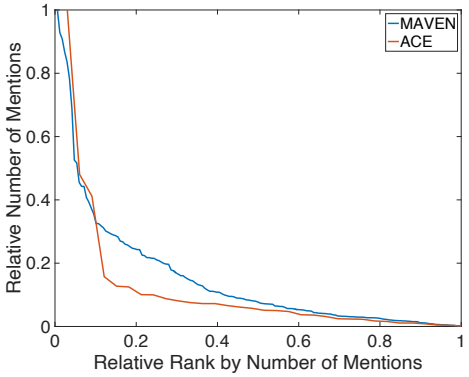

Figure 2: Long tail distribution of event types in both MAVEN and ACE. Y-axis is the number of training mentions for each event type divided by that of the most frequent event type, and X-axis is the rank of event types by number of mentions divided by the total number of event types in the ontology.

event detection framework, we measure the relatedness as a model's prediction of new event types' mentions on old event types, i.e. $P(c_{old}|m_{c_{new}})$ where c_{old} is an old event type and $m_{c_{new}}$ is the mention of a new type. The intuition is that an old event type's (e.g., `Trial`) identifier should predict higher score for a related new type such as `Charge` than an unrelated new type such as `Marry`. We then transfer knowledge between related event types in two directions. We use the representations of old event types to help the learning of new event types through knowledge-aware initialization, transferring knowledge from related learned event types to new event types. We also transfer knowledge from related new event types

to the old types by continually training the representations of old events with the data for new event types using a self-training loss. Our proposed framework can improve learning both old and new types, especially for acquiring new long-tail types. To summarize, our contributions are three-fold:

- We propose a new formulation for lifelong event detection;
- We study the unique challenge of lifelong event detection from heavily imbalanced type distribution;
- We propose a novel framework that can transfer knowledge between related types to benefit the learning of old and new event types, especially for new long-tail types. Our framework outperforms state-of-the-art methods with over 5.1% F1 score under our setting, and improves over 30% F1 on several rare types.

2 Approach

2.1 Problem Formulation

In event detection, given a text sequence $w_{1:L}$ and a target text span specified by the start and end offsets (s, e), we aim to classify the target span into a type in the ontology or label it as NA if it is not an event mention. This definition is generalizable to many other IE tasks by varying the number m of target spans. For instance, entity recognition follows the same setting as event detection where $m = 1$. Relation extraction takes $m = 2$ target entity spans to identify relations between them.

In lifelong event detection, the training phase is separated into a sequence of time stages, which we will denote by t. A stream of datasets $\{\mathcal{D}_t\}$ containing training instances in the above form is provided to the model sequentially according the current time stage t . Each dataset consists of training instances for a set of types $\mathcal{C}_t = \{c_t^1, c_t^2, \ldots, c_t^{n_t}\}$ and negative instances for NA. We will denote NA by c^ϕ below for equations. $\mathcal{C}_t \bigcap \mathcal{C}_{t'} = \emptyset$, meaning that the model continually learns new event types. At stage t, the model needs to detect events for the combined ontology of all seen types, i.e., $\mathcal{O}_t = \mathcal{C}_1 \bigcup \ldots \bigcup \mathcal{C}_t$. Throughout this paper, we don't include type NA when mentioning the term *ontology* unless specified. Compared with the traditional supervised learning setting, the main difference of lifelong learning is that the model is exposed to only training data $\mathcal{D}_t$ that covers a subset of ontology $\mathcal{O}_t$, while the traditional setting always train the model on the full training data $\bigcup_t \mathcal{D}_t$ on the full ontology $\bigcup_t \mathcal{O}_t$. We will refer to this setting as "joint training" on all event types in this paper. Since our definition of event detection can generalize to other IE tasks as shown above, this formulation can also generalize to various lifelong IE tasks.

2.2 Baseline Framework

We first introduce a simple baseline framework that applies experience replay and knowledge distillation to a span-based event detection model for lifelong event detection.

Span-based Event Detection Model. Similar to (Wadden et al., 2019) consider $x = (w_{1:L}, s, e)$ as a training instance consisting of the sentence $w_{1:L}$ and span offsets (s, e) as described in Section 2.1, and y is the corresponding event type label. We first use BERT (Devlin et al., 2019) to encode the text span x

$$\begin{aligned} \boldsymbol{w}_{1:L} &= \text{BERT}(w_{1:L}) \\ \boldsymbol{x} &= \text{InputMap}([\boldsymbol{w}_s; \boldsymbol{w}_e]), \end{aligned} \tag{1}$$

where InputMap is a two-layer feedforward neural network that maps BERT outputs to a lower-dimensional feature vector. For each type $c_i \in \mathcal{O}_t$ we also assign a unique type embedding $\boldsymbol{c}_i$, and the score for each type is computed via inner product

$$o_i = \boldsymbol{c}_i^\top \boldsymbol{x}. \tag{2}$$

Since the instances corresponding to NA are not semantically consistent, it could pose additional challenge to learn a type embedding for c^ϕ. Hence, we don't compute score for c^ϕ as above and instead always set $o^\phi = 0$. In this way, we essentially train the model to output negative scores for negative instances on all valid types. We use softmax to achieve output probability distribution,

$$\begin{aligned} p(c_i|x) &= \frac{e^{o_i}}{1 + \sum_{j=1}^{|\mathcal{O}_t|} e^{o_i}} \\ p(c^\phi|x) &= \frac{1}{1 + \sum_{j=1}^{|\mathcal{O}_t|} e^{o_i}} \end{aligned}. \tag{3}$$

Then the cross-entropy loss is used to train the model on the current dataset:

$$\mathcal{L}_C = \sum_{(x,y)\in\mathcal{D}_t} -\log p(y|x). \tag{4}$$

Although some of the methods designed explicitly for event detection (Ji and Grishman, 2008; Chen et al., 2015; Feng et al., 2016; Liu et al., 2017; Yan et al., 2019; Tong et al., 2020) may have better detection performance, this model is more flexible in that (1) by taking event detection as label prediction for text spans, many existing lifelong learning methods for classification become applicable; (2) as described in Section 2.1, this architecture can handle a variety of IE tasks without significant modification.

Experience Replay. An exemplar set is kept and updated continually containing training instances for all learned types to remind the model when old training data is no longer available. We use $\mathcal{E}_t$ to denote the exemplar set for types in ontology $\mathcal{O}_t$. In lifelong learning literature (Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2019; Hou et al., 2019), most methods either allocate k slots for each type, or fix K slots in total and allocate evenly among all learned types where k or K is hyperparameter. Although the latter setting is considered economical in memory, the framework is limited to learn at most K types. Therefore we adopt the former as the more appropriate setting for lifelong learning where we may want to learn infinitely many types. We select exemplar instances of a type using herding algorithm (Welling, 2009) after the model is trained, following (Rebuffi et al., 2017) and most follow-up work. We don't need to keep instances for c^ϕ, since we assume negative instances are always available in the background text of positive mentions. At stage t, we augment the training dataset $\mathcal{D}_t$ with $\mathcal{E}_{t-1}$ in Equation (4).

Knowledge Distillation. Knowledge distillation (Hinton et al., 2015) is also widely used in lifelong learning (Rebuffi et al., 2017; Li and Hoiem, 2018; Castro et al., 2018; Wu et al., 2019; Hou et al., 2019). Before learning new types, we will keep a copy of the old model for ontology $\mathcal{O}_{t-1}$. During learning on each instance x, we compute p^{t-1} as old output probabilities on learned ontology $\mathcal{O}_{t-1}$ using Equation (3). Then we rescale the old and current predictions on $\mathcal{O}_{t-1}$ by a temperature parameter T,

$$\begin{aligned} \hat{p}^{t-1} &\sim \left(p^{t-1}\right)^{1/T} \\ \hat{p} &\sim p^{1/T} \end{aligned}, \tag{5}$$

We take $T = 2$ in our experiments to retain the old model's output distribution on all learned types instead of only the predicted labels (Hinton et al., 2015).

$$\mathcal{L}_D = - \sum_{\substack{(x,y)\in\mathcal{D}_t \bigcup \mathcal{E}_{t-1} \\ c\in\mathcal{O}_{t-1}\bigcup\{c^\phi\}}} \hat{p}^{t-1}(c|x) \log \hat{p}(c|x). \tag{6}$$

The final loss is a weighted sum of $\mathcal{L}_C$ and $\mathcal{L}_D$,

$$\mathcal{L} = \frac{|\mathcal{O}_{t-1}|}{|\mathcal{O}_t|}\mathcal{L}_D + (1 - \frac{|\mathcal{O}_{t-1}|}{|\mathcal{O}_t}|)\mathcal{L}_C. \tag{7}$$

2.3 Knowledge Transfer Between Learned Types and New Types

New to Old. Traditional lifelong learning methods focus on the catastrophic forgetting problem by retaining the old models' knowledge and don't effectively update the old knowledge. If a new type is related to some old types, some instances of the new type may share similarity with the old types. We utilize these instances to update learned knowledge by extending knowledge distillation loss that only retains old knowledge to a new self-training loss similar to (Xie et al., 2016) that trains the model with a soft pseudo-label predicted by the old model. For each instance $x \in \mathcal{D}_t$, we first compute a distribution over old types as the pseudo-label $q \sim \left(p^{t-1}\right)^{1/\tau}$ with a temperature factor $\tau = 0.5 < 1$ to sharpen the distribution. Then we train the model with the following loss,

$$\mathcal{L}_S = - \sum_{\substack{(x,y)\in\mathcal{D}_t \\ c\in\mathcal{O}_{t-1}\bigcup\{c^\phi\}}} q^{t-1}(c|x) \log p(c|x). \tag{8}$$

Note that different from knowledge distillation in Equation (6), the current model output is not scaled by the temperature τ, which facilitates the model to update its knowledge on learned types. Then we substitute $\mathcal{L}_D$ in Equation (7) with $\lambda\mathcal{L}_S + (1 - \lambda)\mathcal{L}_D$, where λ is a weighting hyperparameter.

Old to New. We transfer old knowledge to new types by initializing the type embeddings for new types based on learned types. When a new type has sufficient training instances, random initialization is usually good enough. We adopt a knowledge-aware random initialization $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, d^2\boldsymbol{I}/\dim(\boldsymbol{r}))$ for frequent new types, with $d = \frac{\sum_{c\in\mathcal{O}_{t-1}} \|\boldsymbol{c}\|_2}{|\mathcal{O}_{t-1}|}$ being the average norm of existing type embeddings. Our intuition is that the norm of learned features also contains knowledge about the feature space, and it can be leveraged to benefit learning of new types.

For long-tail new types with less training data, it may be difficult to train from random initialization and transferring more knowledge from old types can be helpful. To find related old types, we first collect another exemplar set with h instances for each new type before training on them. Suppose $x_{1:h}$ are instances for a new type, the current model's output $p(\cdot|x_i)$ in Equation (3) is considered as relatedness measure. The type embeddings of learned types are aggregated accordingly

$$\boldsymbol{\omega} = \frac{1}{h}\sum_{i=1}^{h}\sum_{c\in\mathcal{O}_{t-1}} p(c|x_i)\boldsymbol{c}. \tag{9}$$

The new type is not necessarily a combination of existing knowledge. We represent the new knowledge by aggregating encoded exemplar instances

$$\boldsymbol{\nu} = \frac{1}{h}\sum_{i=1}^{h} p(c^{\phi}|x_i)\frac{d\boldsymbol{x}_i}{\|\boldsymbol{x}_i\|_2}, \tag{10}$$

where $\boldsymbol{x}_{1:h}$ are computed using Equation (1) and rescaled with the average norm of existing type embeddings d. We weight each exemplar instance by $p(c^{\phi}|x_i)$, indicating how unrelated it is to learned types. The new type initialization is $\boldsymbol{\mu} = \boldsymbol{\omega} + \boldsymbol{\nu}$.

We shift between these two cases with a gate function $g_{\alpha,\beta}(N) = \alpha\exp(-\beta N)$ where N is the number of the new type's training instances and α, β are positive hyperparameters. The new type embedding $\boldsymbol{z}$ is computed as

$$\boldsymbol{z} = g_{\alpha,\beta}(N)\boldsymbol{\mu} + (1 - g_{\alpha,\beta}(N))\,\boldsymbol{r}. \tag{11}$$

3 Experiments

3.1 Datasets and Incremental Tasks

We use two datasets and create incremental tasks on them to evaluate lifelong learning. We include detailed statistics of data splits in the Appendix.

ACE 2005 English (Walker et al., 2006): ACE 2005 English corpus contains 33 event types. We find that in the split used by previous work (Lin et al., 2020) several event types are missing in the development set and the test set. Hence, we re-split the data for better coverage of event types.

MAVEN (Wang et al., 2020): This is a general domain event detection dataset with 168 event types. MAVEN covers a wide range of diverse event types compared with ACE 2005 English dataset, and this diversity makes it a better benchmark for evaluating the correlation and knowledge transfer between old and new event types. Since the original test annotations are not publicly available, we use the original development set as a test dataset and randomly collect another development set from the training data.

Incremental Tasks We construct incremental tasks for our formulation of lifelong event detection in Section 2.1. We partition the ontology into 5 subsets. Then these subsets are given to the model in a fixed order as $\mathcal{D}_{1:5}$. At stage t, the model needs to perform event detection for types in seen subsets $\mathcal{D}_{1:t}$, resulting in 5 incremental event detection tasks with expanding ontology. We denote them by *Task* 1-5. Although we may also need to tackle ontologies from multiple datasets, we simulate this situation by partitioning ontology from a single dataset to avoid the implicitly overlapping types in existing benchmark datasets. In our experiments, we sample one random partition of types in the ontology. We then sample 5 random permutations of orders given to the model. We report the average performance on 5 random permutations. We include the details of the sampled partitions and order permutations in the Appendix.

3.2 Experiment Settings

We experiment with two settings.

Oracle Negative: we provide "oracle" negative instances, including all negative instances in the original datasets and all unlearned types. We exclude instances for learned types for the training of new types. This setting simulates the situation of adding the new type's annotations into the existing dataset for the old types. Our lifelong detection framework needs only added annotations to update the detection model.

Silver Negative: we provide negative instances, including the negative instances in the original datasets, instances of unlearned types, and also instances for already learned types. This setting simulates annotating new types in a different corpus from the existing dataset for the old types. This setting is practical when new types come from another domain, or when we want to add new documents to train the model. We include more details of this setting in the Appendix. We only experiment with this setting using the larger MAVEN dataset since we need sufficient training instances for each type. We hold out some of them as negative instances for other data subsets.

Evaluation We use the F1 score to evaluate the model's performance for each task. In traditional

lifelong learning evaluation for Task i, the model is only tested on instances for learned types. While in event detection, we constantly evaluate the model on the entire test set while taking the mentions of unlearned types as negative instances.

We include hyper-parameters and training details in the Appendix.

3.3 Methods in Comparison

We consider the following methods for comparison.
Finetune: The model is simply finetuned on $\mathcal{D}_t$ at time step t.
KD+R: The baseline framework introduced in Section 2.2 that combines knowledge distillation with replay.
KD+R+K (Ours): The baseline framework and proposed knowledge transfer strategy introduced in Section 2.3.
Joint: We also report a joint training performance over all types using the same event detection module in 2.2 as upperbound for the final task.

We also adapt class incremental learning methods **iCaRL** (Rebuffi et al., 2017), **EEIL** (Castro et al., 2018), **BIC** (Wu et al., 2019) to our tasks based on our best knowledge on the papers and official code if released. **KCN*** (Cao et al., 2020) studies a different formulation of lifelong event detection. We include a more detailed comparison with their formulation in Section 4. We are able to adapt their main methods for our formulation. We give brief descriptions on these methods and adaptation details in the Appendix.

3.4 Results on *Oracle Negative*

The main results for the *Oracle Negative* setting are summarized in Table 1. All methods but iCaRL have the same performance on Task 1, since these lifelong learning strategies are not applicable when training on the first subsets of types. However, iCaRL uses a different exemplar-based scoring method. Existing methods with balancing strategy suffer from the problem of long-tail distribution on event types. Compared to our simpler KD+R, all of iCaRL, EEIL, and BIC balance the training data of new types and the exemplars of old types for classification but achieve less competitive results on our task. For iCaRL, we found significantly worse results in the beginning tasks while improving performance for later tasks. The reason is that iCaRL is a feature-based method and learns better representation with more training data of various types. It also explains why it has even worse performance on ACE 2005 with much fewer event mentions. EEIL and BIC also show comparable or even slightly inferior performance over KD+R without balancing, indicating that naïve balancing may degrade performance when the original distribution is long-tailed.

Most event detection evaluation focuses on micro F1 score that is averaged over instances. Due to the long-tail distribution of event types, micro F1 score is usually dominated by frequent event types. To further study the improvement for old types and new types, we consider the per-type F1 scores (a.k.a macro F1) of our proposed KD+R+K and baseline framework KD+R. In Figure 3, we show the curves of macro F1 scores on learned and new types. We notice that our framework improves performance on both learned types and new types at all stages. Comparing the curves for rare new types with less than 120 training mentions, KD+R+K and KD+R, our framework significantly improves the performance for rare new types. We also observe that the performance gain is larger in later stages because the model has seen more event types and accumulated more knowledge.

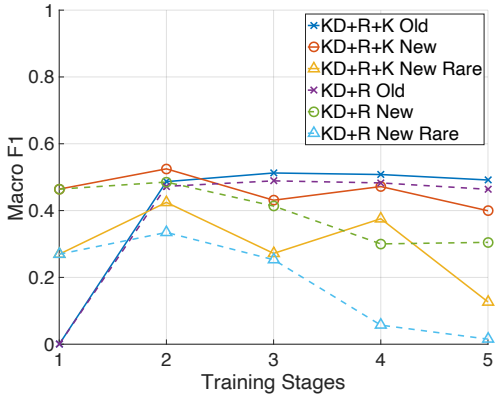

Figure 3: Performance on old types, new types, and rare new types with less than 120 training instances, which reflects the model's ability to keep learned knowledge, to acquire new types, and to acquire new long-tail types respectively. Solid lines refer to our knowledge transfer framework (KD+R+K) and dotted lines refer to the baseline framework without knowledge transfer (KD+R). The figure is plotted for one random permutation of MAVEN.

3.5 Results on *Silver Negative*

We show results in Table 2 for *Silver Negative* setting. Compared with the *Oracle Negative* setting, the first task's performance is worse with reduced

Task	MAVEN 1	2	3	4	5	ACE 2005 1	2	3	4	5
Finetune	63.16	40.30	33.00	23.86	22.34	60.21	43.38	38.01	21.98	24.57
iCaRL	18.08	27.03	30.78	31.26	29.77	4.05	5.41	7.25	6.94	8.94
EEIL	63.16	48.17	44.17	40.35	37.75	60.21	50.01	51.47	46.27	45.20
BIC	63.16	55.51	53.96	50.13	49.07	60.21	57.69	60.18	56.99	53.81
KCN*	63.16	55.73	53.69	48.86	47.44	60.21	58.08	61.43	58.45	55.77
KD+R	63.16	55.50	54.12	50.31	49.22	60.21	57.69	59.47	57.46	53.97
KD+R+K (Ours)	63.16	**58.04**	**57.47**	**54.94**	**54.32**	60.21	**59.56**	**61.53**	**58.68**	**57.57**
Joint (Upperbound)	/	/	/	/	63.46	/	/	/	/	66.23

Table 1: Classification F1 scores (%) on incremental tasks in *Oracle Negative* setting. Later tasks have larger ontologies with new types. KCN* refers to adapted version of original KCN (Cao et al., 2020). We also provide additonal ablation results on two-way knowledge transfer in Appendix.

Task	MAVEN 1	2	3	4	5
Finetune	54.60	38.71	33.54	24.78	23.71
iCaRL	18.12	24.81	29.80	32.49	33.55
EEIL	54.60	47.13	42.56	37.33	34.13
BIC	54.60	53.21	54.84	52.58	52.06
KCN*	54.60	53.63	55.08	53.00	52.33
KD+R	54.60	53.22	54.76	52.59	52.15
+K (Ours)	54.60	**54.82**	**57.05**	**56.16**	**56.76**
Joint	/	/	/	/	63.46

Table 2: Classification F1 scores (%) on incremental tasks in *Silver Negative* setting.

training data because we hold some of them as silver negative instances for later training. However, there is a significantly smaller performance drop as training proceeds for BIC, KCN*, KD+R, and KD+R+K. This result indicates that most lifelong learning methods can effectively avoid catastrophic forgetting when the mentions of old types exist in the context of new types, even if we don't provide annotations for them. Furthermore, our proposed methods are more effective, showing improved performance on some later tasks since our knowledge transfer module explicitly utilizes related new instances to update learned knowledge. However, the significant gap compared with joint training performance indicates that improving old knowledge with new related training data instead of retaining learned knowledge from old training data is an important research direction.

3.6 Case Study on Knowledge Transfer

In Table 3 we show some examples of test instances with new long-tail event types. The baseline framework fails to identify these instances and labels them as `NA`. Our proposed method makes the correct predictions leveraging knowledge from related old types. For each of these new types, we examine the weights over old types $p(c|x_i)$ in Equation (9) and show the most related old type with the highest weight. The semantic relatedness between these types can help the identification.

Furthermore, we compare the precision and recall of two rare event types in Table 4. We can observe that our proposed method improves the recall significantly, although it has a slightly lower precision. The extremely low recall of the baseline method is due to the lack of training instances, which causes the model to identify only the most similar mentions. In contrast, our proposed framework can identify more diverse instances with knowledge transfer, although such knowledge from related types may also bring some false positives that slightly undermine the precision.

4 Related Work

4.1 Event Detection

In this work we mainly use event detection as an exemplar task. Event detection under the traditional setting has been widely studied (Ji and Grishman, 2008; Chen et al., 2015; Feng et al., 2016; Liu et al., 2017, 2018, 2019; Lu et al., 2019; Ding et al., 2019; Yan et al., 2019; Tong et al., 2020). Other methods on joint information extraction (Li et al., 2013; Wadden et al., 2019; Lin et al., 2020) also include event detection as a subtask. There are also a few attempts to apply lifelong learning to information extraction. Nguyen et al. (2016) studies the problem of adding one new event type into the existing model. However they only focus on frequent types and single-stage incremental learning. Wang et al. (2019) study lifelong relation extraction while mainly focusing on the relation classification scenario without paying special attention to the `NA`

New Instance	New Type	Related Old Type
In 1378 Manuel II Palaiologos **promised** to hand over the city of Philadelphia to the Turks in return for the Ottoman sultan's aid in a disastrous Byzantine civil war.	Commitment	Statement
Graham's family filed a **lawsuit** against the city of New York, and the lawsuit was settled for $ 3. 9 million in 2015.	Legality	Judgment_communicationn
The French and British commanders in the pocket decided to make for Le Havre and Fortune detached Arkforce, the equivalent of two brigades, to **guard** the routes back to the port.	Patrolling	Perception_active

Table 3: Examples of test instances with long-tail event types for which the proposed methods with knowledge transfer detects successfully while the baseline framework predicts as NA. The related old types are the most related types based on $p(c|x_i)$ in Equation (9).

Type: Commitment	P	R	F
KD+R	**100.00**	3.45	6.67
KD+R+K	66.67	**34.48**	**45.45**
Type: Cost	P	R	F
KD+R	**100.00**	5.26	10.00
KD+R+K	81.82	**47.37**	**60.00**

Table 4: Precision and Recall (%) on two MAVEN event types with fewer than 120 instances that KD+R has non-zero scores. With knowledge transfer recall is significantly improved. KD+R overfits on training instances, resulting in perfect precision but low recall.

type. (Cao et al., 2020) focuses on lifelong event detection. However, they bypass the long-tailed distribution problem by focusing on frequent event types, while we consider severe data imbalance as the challenge of lifelong IE. Besides, based on their released code, the framework is designed for single token trigger and learning one type at a time, and thus it is limited to extend to more general settings. In contrast, our formulation of lifelong IE can be applied to more general cases for event detection and also other IE tasks. Besides, our framework outperforms the adapted version of their methods under our formulation.

4.2 Lifelong Learning

Our formulation is mainly developed based on class incremental learning. The *Learning without Forgetting* method (Li and Hoiem, 2018) uses knowledge distillation to avoid catastrophic forgetting. ICaRL (Rebuffi et al., 2017) combines knowledge distillation with experience replay to learn feature representation for each class. EEIL (Castro et al., 2018) adopts an end-to-end training method by adding another finetuning stage on the balanced exemplar set. BIC (Wu et al., 2019) prevents overfitting during balanced finetuning by training only two additional parameters on a small balanced validation exemplar set to correct the bias towards new classes. Hou et al. (2019) add additional constraints into the loss to further reduce forgetting. Other work uses bi-level optimization to select better exemplar instances (Liu et al., 2020; Borsos et al., 2020). Most of these methods conduct experiments on image classification. Our formulation and proposed framework consider the unique challenge of the long-tail distribution in event detection and information extraction and take advantage of rich correlations among ontology types.

In addition to class incremental learning, regularization-based methods (Kirkpatrick et al., 2016; Jung et al., 2020; Ahn et al., 2019; Golkar et al., 2019; Serrà et al., 2018) are widely used for the situation where the model needs to learn a sequence of "disjoint" tasks. Unlike class incremental learning, the model continually learns classification on an entirely new ontology at each stage instead of combining old ontology with several new classes.

5 Conclusions and Future Work

In this paper, we formulate the Lifelong Event Detection problem, and propose a novel framework that transfers knowledge among related event types and between old and new types to tackle the unique challenges brought by the imbalanced distribution of event types. This framework benefits learning of both old types and new types, and improves performance on long-tail types. It is worth mentioning that although we use event detection as an exemplar

task, our proposed framework is applicable to other information extraction tasks. We will extend and empirically study our approach to them in the future. Moreover, in lifelong learning the new types' can come from a completely different domain, for which the context distribution (such as text style and genre) changes significantly from seen training instances. We leave explicit exploration and evaluation of lifelong learning for specific domains as future work. Moreover, although the knowledge transfer can improve the performance on long-tail types significantly, there is still a gap between rare types and frequent types. We can combine our framework with other efficient learning methods such as zero-shot learning, few-shot learning, and weakly-supervised learning to learn these types better.

References

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. 2019. Uncertainty-based continual learning with adaptive regularization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 4394–4404.

Zalán Borsos, Mojmír Mutný, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming. *CoRR*, abs/2006.03875.

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.

Francisco M. Castro, Manuel J. Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. 2018. End-to-end incremental learning. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XII*, volume 11216 of *Lecture Notes in Computer Science*, pages 241–257. Springer.

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.

Ning Ding, Ziran Li, Zhiyuan Liu, Haitao Zheng, and Zibo Lin. 2019. Event detection with trigger-aware lattice neural network. 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)*, pages 347–356, Hong Kong, China. Association for Computational Linguistics.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. 2016. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 66–71, Berlin, Germany. Association for Computational Linguistics.

Siavash Golkar, Michael Kagan, and Kyunghyun Cho. 2019. Continual learning via neural pruning. *CoRR*, abs/1903.04476.

R. Grishman and B. Sundheim. 1996. Message understanding conference- 6: A brief history. In *COLING*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. 2019. Learning a unified classifier incrementally via rebalancing. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 831–839. Computer Vision Foundation / IEEE.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. An overview of the tac2011 knowledge base population track. In *Proc. Text Analysis Conference (TAC2011)*.

Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. 2020. Continual learning with node-importance based adaptive group sparse regularization. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 33, NeurIPS 2020, December 2020*, volume 33.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Zhizhong Li and Derek Hoiem. 2018. Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2019. Neural cross-lingual event detection with minimal parallel resources. 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)*, pages 738–748, Hong Kong, China. Association for Computational Linguistics.

Shaobo Liu, Rui Cheng, Xiaoming Yu, and Xueqi Cheng. 2018. Exploiting contextual information via dynamic memory network for event detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1030–1035, Brussels, Belgium. Association for 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.

Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. 2020. Mnemonics training: Multi-class incremental learning without forgetting. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 12242–12251. IEEE.

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.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 158–165, Berlin, Germany. Association for Computational Linguistics.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5533–5542. IEEE Computer Society.

Mark B. Ring. 1995. *Continual learning in reinforcement environments*. Ph.D. thesis, University of Texas at Austin, TX, USA.

Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4555–4564. PMLR.

Sebastian Thrun. 1998. Lifelong learning algorithms. In Sebastian Thrun and Lorien Y. Pratt, editors, *Learning to Learn*, pages 181–209. Springer.

Meihan Tong, Bin Xu, Shuai Wang, Yixin Cao, Lei Hou, Juanzi Li, and Jun Xie. 2020. Improving event detection via open-domain trigger knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5887–5897, Online. Association for Computational Linguistics.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. 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)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus LDC2006T06. Web Download. Philadelphia: Linguistic Data Consortium.

Hong Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 796–806, Minneapolis, Minnesota. Association for Computational Linguistics.

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. *CoRR*, abs/2004.13590.

Max Welling. 2009. Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1121–1128. ACM.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 374–382. Computer Vision Foundation / IEEE.

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 478–487. JMLR.org.

Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. 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)*, pages 5766–5770, Hong Kong, China. Association for Computational Linguistics.

Alexander Yates, Michele Banko, Matthew Broadhead, Michael Cafarella, Oren Etzioni, and Stephen Soderland. 2007. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 25–26, Rochester, New York, USA. Association for Computational Linguistics.

A Dataset Details

A.1 Collection of New Splits

On MAVEN (Wang et al., 2020), we take the original development set as test set, and collect another development set from the original training data. The collection process is as follows: we first randomly sample 413 documents from the original 2913 training documents as development documents. Then we manually check the missing types. For each missing type, we sample one more document from the remaining training documents that mentions it. Then we end up with a development set that covers all event types.

On ACE (Walker et al., 2006), we develope new splits from the one used by Lin et al. (2020). We do the similar process as above to add documents to both development and test set. Since `Justice:Pardon` only has 2 event mentions in the entire ACE 2005 data, we don't include this type `Justice:Pardon` in both development and test set. Besides, since original development set misses much more types than original test set, the modified development set contains more documents than test set. We therefore use the modified test set as development set, and modified development set as the test set to make the test set more diverse.

We show detail statistics of the new splits in Table 5. We use the provided negative instances for MAVEN (see original paper (Wang et al., 2020) for more details), and collect negative instances as all unlabeled consecutive text spans of at most 3 tokens for ACE 2005.

		#Document	#Mention	#Negative
MAVEN	Train	2,498	66,812	277,839
	Dev	415	11,181	46,001
	Test	710	18,904	79,699
ACE	Train	501	4,088	717,302
	Dev	41	433	54,544
	Test	55	790	93,566

Table 5: Statistics of used MAVEN/ACE training, development and test sets.

A.2 Collection of Type Partitions and Permutations

Traditional lifelong learning methods usually make each subset contain same number of types. However, since the data distribution is heavily long-tailed, we construct each subset to contain approximately same number of the instances. On ACE 2005 `Conflict:Attack` takes up around 30% of total mentions, therefore the resulting partition is not perfectly balanced. We sample one random partition of types in the ontology for MAVEN and ACE 2005 respectively, and further sampler 5 ran-

Data Subset	MAVEN		ACE 2005	
	#Type	#Mention	#Type	#Mention
1	33	12,783	9	584
2	30	12,259	6	840
3	39	14,268	5	1,335
4	35	13,209	5	717
5	31	14,293	8	612

Table 6: Distribution of types and instances in subsets.

dom permutations of the order given to the model. Hence we have 5 sets of incremental tasks for each dataset. The sampling process is as follows: for each dataset we first randomly shuffle the list of all event types, and initialize five empty sets. Then we traverse over the shuffled list, and each time when we pick up an event type, we will put it into one of the five sets types in which have the fewest total training instances. After all event types are visited, the construction of partitioning subsets is finished.

We show basic statistics of partitions in Table 6. We provide the partition of types, and order permutations used in the code.

A.3 Details on *Silver Negative* Setting

We also introduce how we prepare $\mathcal{D}_t$ in *Silver Negative* setting. We first divide the entire MAVEN training splits into 169 subsets, each representing mentions for an event type including `NA`. For `NA` subset, we evenly devide it into 5 subsets and put them into $\mathcal{D}_{1:5}$ respectively. For each valid event types, we divide its subsets into 5 subsets, with a training subset containing 90% of training instances, and 4 silver negative subsets evenly split the rest of instances. The training subset is put into the $\mathcal{D}_i$ that it is supposed to be learned as new type. The rest 4 subsets serve as negative instances with label `NA` in other data subsets. This setting is to simulate the situation that all event types are in similar domains, and frequently co-occurs with each other. Therefore in annotated data for a subset of event types, event mentions of other types in the context will become negative instances.

B Hyperparameters and Training Details.

B.1 Hyperparameters

All equation references in this section refer to the main paper. We use BERT-large-cased (Devlin et al., 2019) as the BERT encoder and fix it during training, and concatenated span offsets is mapped to dimension of 512 in Equation (1). Number of exemplar instances for each type is 20. For knowledge transfer, λ is used to balance knowledge distillation loss and new-to-old type knowledge transfer loss. We assume that there will be little new knowledge for old types from a new type and old types if the old model confidently predicts its instance as `NA` with high probability. Therefore, we will use $\lambda = 0.5$ if the probability of `NA` is less than 0.9, otherwise we set $\lambda = 0$, meaning that only knowledge distillation is used. α, β specifies the gating function in Equation (11). We use $\alpha = 0.5$ and $\beta = 0.05$. $h = 20$ is the number of pre-fetch exemplar instances for initialization of new types' embeddings. To avoid irrelavant old knowlege in initialization in Equation (9), we only aggregate most probable old types with total probability over 0.9 and with renormalization on remaining old types.

B.2 Training Details

We apply an additional filtering on ACE 2005 negative instances during training to reduce training time. We only include all single-token span, or multi-token span that overlaps with any event mentions as negative instance. This filtering will reserve only one third of all negative instances in Table 5, although it slightly degrades the upper-bound joint training performance by around 1% F1. Better filtering techniques may be developed to trade off between training time and performance.

During training, we use AdamW (Loshchilov and Hutter, 2019) to train the model with learning rate $1e-4$ and weight decay $1e-2$. We use batch size of 128 to sample instances from $\mathcal{D}_t$. We append instance from exemplar set as additional training data to each training batch. For each training stage t on $\mathcal{D}_t$, we run for a maximum of 20 epochs. Early stopping is adopted if performance on development set doesn't increase for consecutive 5 epochs. It is worth mentioning some class incremental methods run experiments with fixed epochs for each training stage, which is only reasonable when task of learning new sets of types are equally difficult. However, due to long-tail distribution and variation of number of types in our formulation of lifelong event detection, a fixed number of epochs may result in sub-optimal training for different type subsets.

B.3 Training Environments

We use pytorch for implementation. All models are trained on Nvidia V100.

Task	MAVEN 1	2	3	4	5	ACE 2005 1	2	3	4	5
Finetune	63.57	40.85	31.97	24.37	22.79	64.91	42.81	38.89	20.56	26.60
iCaRL	17.76	27.78	30.89	30.99	30.26	4.94	6.40	8.01	7.96	9.51
EEIL	63.57	48.52	44.04	40.47	37.85	eeil64.91	56.06	51.33	48.40	50.16
BIC	63.57	56.23	53.79	50.46	49.19	64.91	64.63	63.69	61.09	58.45
KCN*	63.57	56.02	52.97	48.74	47.57	64.91	65.51	65.24	62.56	60.61
KD+R	63.57	56.24	54.13	50.35	49.40	64.91	64.63	64.22	59.56	58.45
KD+R+K (Ours)	63.57	58.67	57.45	55.26	54.41	64.91	64.39	65.18	61.98	61.62

Table 7: Classification F1 scores on development set in *Oracle Negative* setting.

B.4 Performance on Development Set

We compare results for *Oracle Negative* and *Silver Negative* on development set in Table 7 and Table 8.

Task	MAVEN 1	2	3	4	5
Finetune	54.27	39.22	32.50	25.41	24.22
iCaRL	18.05	24.83	29.41	32.23	33.54
EEIL	54.27	46.92	42.43	37.55	34.54
BIC	54.27	53.27	54.31	52.41	52.14
KCN*	54.27	53.59	54.67	52.84	52.61
KD+R	54.27	53.27	54.39	52.36	52.25
+K(Ours)	54.27	54.82	56.59	56.01	56.87
KD+R	54.60	53.22	54.76	52.59	52.15
+K (Ours)	54.60	**54.82**	**57.05**	**56.16**	**56.76**
Joint	/	/	/	/	63.46

Table 8: Classification F1 scores on incremental tasks of development set in *Silver Negative* setting.

B.5 Details of baselines

iCaRL (Rebuffi et al., 2017). In original iCaRL, a classification network is trained using knowledge distillation and experience replay, but only the representation network before the last classification layer is used to score the instances using distance to the mean encoded features of each types' exemplar instances. We refer readers to original paper for more details. In our adaption, event type embeddings are discarded and encoded $\boldsymbol{x}$ in Equation (1) is considered as output of the representation network. And since we always have `NA` in every stage, we use mean encoded features of all negative instances in current training stage to represent it.

EEIL (Castro et al., 2018). In our adaptation, an additional balanced finetuning training on $\mathcal{E}_t$ is performed in addition to our baseline framework, after training on $\mathcal{D}_t$ and updating exemplar set to $\mathcal{E}_t$.

BIC (Wu et al., 2019). In our adaptation, an additional bias correction training on a balanced development exemplar set is performed in addition to our baseline framework, after training on $\mathcal{D}_t$. An affine transformation of scores on new types is learned to correct the bias towards new types. Besides, for the knowledge distillation loss after stage 2, the old scores are also corrected by the stored old correction parameters. Another adaptation is on the collection of development exemplar set. In original paper, since they don't use development set in their benchmarks, development exemplar set is collected from a portion of reserved training instances. We directly collect them from our development set. In this way their model is essentially strengthened to include more data for training.

KCN* (Cao et al., 2020) We re-implemented their hierarchical distillation for feature $\boldsymbol{x}$ in Equation (1) and outputs in Equation (3) of the main paper, while substituting their exemplar set construction with more widely used herding algorithm (Welling, 2009).

B.6 Ablation Studies

We perform ablation studies in Table 9 on the two knowledge transfer components, the new-to-old knowledge transfer and the old-to-new knowledge transfer, under the oracle negative settings on MAVEN dataset. We only show results on task 2-4 since the performance for the first task are the same for all methods.

Task	2	3	4	5
full	58.04	57.47	54.94	54.32
– old-to-new	58.10	57.70	54.76	53.80
– new-to-old	55.50	54.12	50.31	49.22

Table 9: Ablation results on two-way knowledge transfer on MAVEN dataset. The last row is the same as ıthe baseline KD+R method without knowledge transfer.