Knowledge-Based Systems 254 (2022) 109601

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

HEFT: A History-Enhanced Feature Transfer framework for incremental event detection

Kaiwen Wei [a,b,c,d], Zequn Zhang [a,b], Li Jin [a,b,*], Zhi Guo [a,b], Shuchao Li [a,b], Weihong Wang [e], Jianwei Lv [a,b,c,d]

[a] Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China
[b] Key Laboratory of Network Information System Technology (NIST), Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China
[c] University of Chinese Academy of Sciences, Beijing 100190, China
[d] School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China
[e] College of Engineering and Computer Science, the Australian National University, Australia

ARTICLE INFO

Article history:
Received 31 October 2021
Received in revised form 15 July 2022
Accepted 3 August 2022
Available online 11 August 2022

Keywords:
Incremental event detection
History-Enhanced Feature Transfer
Knowledge distillation
Catastrophic forgetting

ABSTRACT

Incremental event detection is a challenging subfield of event detection, which could be applied in various complex situations, especially for the real-time streaming data scenario where new event classes are continuously emerging. To handle the catastrophic forgetting problem, knowledge distillation has been proved to be a promising mechanism since it requires no extra tools or data storage. However, conventional knowledge distillation-based methods are not effective due to two challenging issues: (a) there is an inherent conflict between reserving history information from old tasks and adjusting to new tasks with shared model output features. (b) the historical category information is not fully utilized, making the model prone to forgetting knowledge on the observed tasks. In this work, we present a History-Enhanced Feature Transfer (HEFT) framework to solve the two challenges mentioned above. For the first challenge, we employ a feature transfer module to separate reserving and adjusting sub-functions by reconstructing the old task features. In this manner, the reconstructed features are responsible for preserving historical knowledge, and the feature extractor with followed classifier only requires to focus on new task classification. For the second issue, a history-enhanced question answering model is introduced to reinforce the ability to memorize historical information, which leverages the trained categories as clues to guide the reasoning process. The experimental results show HEFT outperforms the state-of-the-art model by 6.2% and 9.1% of the whole F1 score on ACE 2005 and TAC KBP 2017 benchmarks, respectively. The ablation study and case study also demonstrate that HEFT could overcome catastrophic forgetting of old class event triggers without heavily relying on historically preserved samples.

1. Introduction

Conventional event detection (ED) systems commonly follow an idealized assumption that all data during training and testing must obey identical independent distribution (i.i.d.). However, this assumption is not applicable in real situations because new event classes are rapidly emerging and different classes typically have diverse distributions. Moreover, it is infeasible to memorize all observed data and then re-trained the model from scratch since this method is time-consuming and storage-wasting. To solve such challenging issues, a naive but straightforward approach is directly fine-tuning on new classes. Nevertheless, fine-tuning on new tasks is likely to make the well-trained model forget the knowledge of previous tasks, known as *catastrophic forgetting* (CF). As shown in Fig. 1, an ED system could initially classify the categories of *Meet* triggers correctly. But after fine-tuning on *Elect* event, the output feature space of *Meet* class has been adjusted and mixed up with *Elect* class. As a result, the updated model mistakenly classifies *Meet* event data as *Elect*.

Incremental event detection is proposed to appropriately alleviate the CF problem in ED. It seeks to utilize incremental learning methods to have promising results on new trigger classification tasks while maintaining the performance of all previous tasks. Knowledge distillation [1], which owns the advantage of

* Corresponding author at: Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China.
E-mail address: jinlimails@gmail.com (L. Jin).

https://doi.org/10.1016/j.knosys.2022.109601
0950-7051/

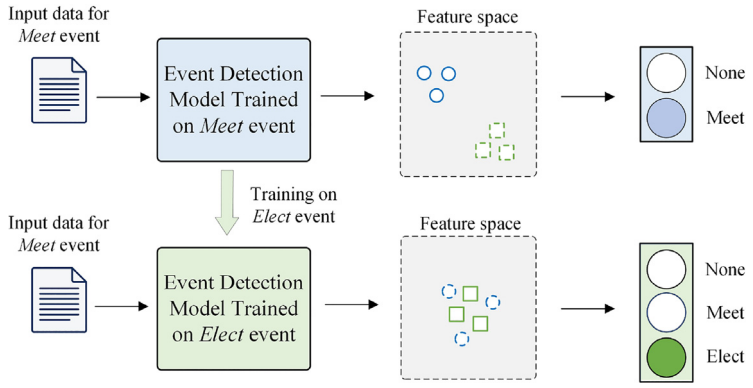


Fig. 1. Illustration of catastrophic forgetting. In the feature space plots, points in various colours and shapes are samples of different categories, and samples in dashed lines indicate that those samples are not available at the current task. The old model (on the top) is trained on the *Meet* event, and could classify the related triggers correctly. After training on the *Elect* event and feeding the same *Meet* event data to the new model (on the bottom), the output features are adjusted. The confusing sample points of the two tasks make the new model hard to classify.

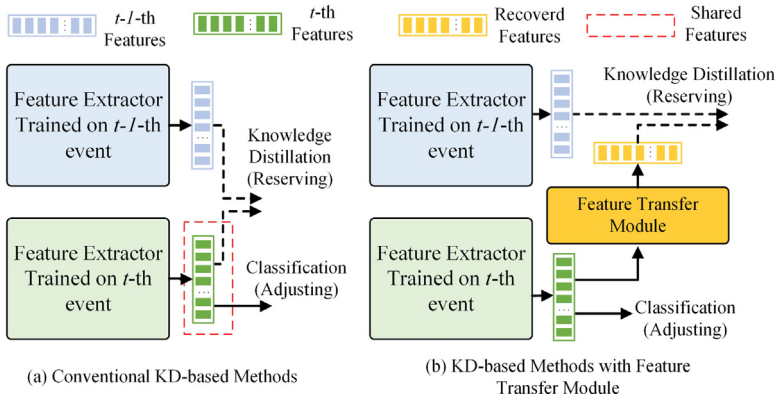


Fig. 2. Illustration of reserving/adjusting conflict issue in conventional knowledge distillation-based methods: (a) conventional knowledge distillation-based methods leverage shared features to simultaneously reserve the knowledge of $t-1$th event and adjust on classification at tth event. (b) feature transfer module maps the features from tth event to $t-1$th event. The recovered features are leveraged for old class trigger classification, and the features from the feature extractor at tth event are used for new class classification.

requiring no extra tools or data storage during training, has become a promising solution [2–4]. The underlying idea of knowledge distillation-based methods is transferring historical knowledge from a network model trained on the old task (teacher) to a new network (student) when learning a new task. Despite achieving great performance, they are facing the following two crucial issues: (1) there is a conflict that conventional distillation-based methods require both to keep the model parameters from the old tasks and to adjust parameters on the new ones with shared features, which we call **reserving/adjusting conflict** issue. (2) the underutilized historical category information makes the model prone to forgetting knowledge on observed tasks, which we call **historical category information utilization** issue. We will elaborate on the two issues.

Reserving/adjusting conflict: traditional knowledge distillation-based approaches expect the student model satisfies two goals: (1) adequately *reserving* historical information. (2) *adjusting* well to new tasks. However, it is impractical for one feature extractor to achieve the aforementioned two goals with shared output features simultaneously. As illustrated in Fig. 2(a), when training at the tth event, the feature extractor cannot fulfil reserving (remaining output features unchanged) and adjusting (changing output features) simultaneously. As a result, a certain level of knowledge would be lost, and the student model cannot achieve ideal performance, i.e., a conflict exists between reserving and adjusting sub-functions with shared output features.

Historical category information utilization: existing methods try to preserve historical information by storing historical sample information or keeping the distribution of historical models. However, historical category information is not fully utilized in previous research. For one thing, trained category information provides more valid knowledge about historical classes and contributes to alleviating catastrophic forgetting. For another, the observed categories could assist in predicting the classes having similar output feature distributions. For instance, it is confusing to classify the event class *die* and *injure* since they have relatively close semantics. Nevertheless, if the event class *die* has been observed before, the distribution of two classes can be pulled apart during the model training process, which is conducive to trigger classification.

To solve the above issues in incremental event detection, we propose a novel history-enhanced feature transfer framework, named HEFT. For the reserving/adjusting conflict issue, we find it ill-suited to meet reserving and adjusting with shared output features simultaneously. Therefore, a feature transfer module (FTM) is proposed to split reserving and adjusting by generating recovered output features of old tasks. As shown in Fig. 2(b), FTM is responsible for precisely mapping the new task features to the previous one: the recovered features hope to be more similar to the output features of the previous task, thus preserving history information. Accordingly, the feature extractor and the followed classifier only need to focus on adjusting to the new task. Noted that [5] also proposed an alignment model which assumes the feature spaces do not distort much in different tasks and work as a regulation mechanism. Without such a hypothesis, FTM

achieves arbitrary feature transformation between tasks, which is more adaptable for sophisticated situations.

To solve the historical category information utilization issue and fully utilize the pre-trained language models, we extend the Machine Reading Comprehension (MRC) model as a question answering (QA) process [6,7,7,8] with trained categories to strengthen the ability to preserve historical category information. Concretely, we leverage a pre-trained language model (e.g., BERT [9]) to build an MRC-based QA system, and add the historical category information to the question content. During training, since there are multi-head and multi-layer attention operations in the pre-trained language model, the question containing historical information and the raw context could thoroughly interact with each other. In this manner, historical category information is integrated in an implicit way, and contributes to alleviating the catastrophic forgetting problem.

We conduct experiments on the widely used ACE 2005 and TAC KBP 2017 benchmarks. The experimental results illustrate that our model achieves state-of-the-art performance compared to other models and possesses a superior ability to overcome catastrophic forgetting problems in incremental event detection scenarios. The contributions to this work can be summarized as follows:

- We introduce a History-Enhanced Feature Transfer (**HEFT**) framework in incremental event detection. The proposed model adequately utilizes knowledge distillation mechanism and could effectively alleviate the catastrophic forgetting problem.
- A Feature Transfer Module (**FTM**) is introduced to overcome the reserving/adjusting conflict with shared features in knowledge distillation-based approaches. By reconstructing old task features, it allows reserving and adjusting subfunctions to work independently and be fully functional.
- A History-Enhanced Question Answering (**HE_QA**) mechanism is introduced to adequately utilize history category information. It leverages observed categories as clues to utilize the pre-trained language models and implicitly incorporates history information.
- Experimental results show that HEFT outperforms existing methods significantly, and achieves 6.2% and 9.1% improvements in whole F1 score on the ACE 2005 and TAC KBP 2017 benchmarks, respectively.

We first discuss related work in the next section. We then describe our model architecture in Section 3. In Section 4, we present experimental results and performance analysis. Finally, We summarize the paper in Section 5.

2. Related work

2.1. Event detection

In recent years, neural network layers have been leveraged by deep learning methods to capture deep semantic information automatically. [10] proposed Dynamic Multi-pooling Convolutional Neural Network (DMCNN) based on CNN. [11] introduced an RNN-based method for the event detection task. Later, [12] used a graph convolutional neural network (GCN) over dependency trees to capture syntactic contextual representations of each node. [13] made the model concentrate more on entities by building a supervised attention mechanism for trigger detection. [14–16] also employed distance supervision or data augmentation strategy to promote the ED model.

Recently, pre-trained language models (e.g., BERT [9] and EMLo [17]) have made considerable improvements in many natural language processing (NLP) tasks since they incorporate universal language representations from a large amount of unlabelled data. [18] applied BERT as the feature extractor directly for the event detection task, achieving great performance without designing task-specific architectures or using external resources. Besides, there is a trend to model event detection tasks as question answering (QA) tasks, which utilize manually designed question templates to extract the corresponding answers from the original context. [6–8] formulated event detection as a machine reading comprehension (MRC) task and achieved state-of-the-art results, where the extraction of trigger information is treated as a QA process. Although promising performance can be achieved on fixed datasets, in the real world, new event classes are rapidly emerging, and the distributions of different classes are usually diverse. Simply fine-tuning the model on new classes is likely to result in the catastrophic forgetting problem. Moreover, due to issues such as huge computation costs, storage budget, or training time restrictions, storing up all training data and then re-train the model from scratch is inappropriate. Therefore, some researchers proposed incremental event detection task [3] by introducing incremental learning methods to alleviate the catastrophic forgetting problem.

2.2. Incremental learning methods

The majority of existing continual learning approaches could be split into the following four categories: regularization-based methods, replay-based methods, architecture-based methods, and knowledge distillation-based methods. Regularization-based methods calculate the importance of parameters and then add certain regularization items in loss functions to penalize the forgetting of the previous knowledge. Elastic Weight Consolidation (EWC) [19] and its variants [20–24] protected important parameters for old tasks from being changed. [5] alleviated the distortion of output features by appending simple linear transformations with L2 normalization. These methods usually do not need to save any previous data and only train on each task once, but they have limited abilities to overcome catastrophic forgetting. Replay-based methods refer to storing a part of previously learned data or training another models to create pseudo data of old tasks. [25] proposed iCaRL model, which stores several samples for each category closest to the average category vector. Similarly, [5] introduced an episodic memory replay (EMR) method by randomly reserving examples. Replay-based methods have been proven to be the most promising for NLP tasks [3,5,26–28]. To benefit from replay-based methods, our method HEFT also stores a small portion of samples. Architecture-based methods attempt to prevent forgetting by adding more modules for new tasks. Typically, previous task parameters are kept fixed [29] or masked out [30,31]. Afterwards, [32] and the improved versions [33,34] proposed progressive neural networks (PNN) by having a specific model for each task and connecting it to the old ones. There are two crucial issues in architecture-based methods: firstly, if the extended model is designed to be large, it will significantly increase the size of the model. Secondly, the noise brought by complex additional modules makes the whole model hard to converge. Those methods increase the storage space of the model, as does our proposed HEFT, but HEFT adds only a tiny amount of parameters for each additional task. Knowledge distillation-based methods are inspired by the knowledge distillation mechanism, which was firstly proposed by [35]. Those methods leveraged the model trained from old tasks as the teacher model and treated the model scheduled to be trained on the new task as a student model. The student model learns historical information by imitating the behaviour of the teacher model. [3] leveraged the teacher model to simultaneously distill knowledge from feature-level and prediction-level to the student model. [1] proposed a sequence-level knowledge distillation and [2] presented online distilling

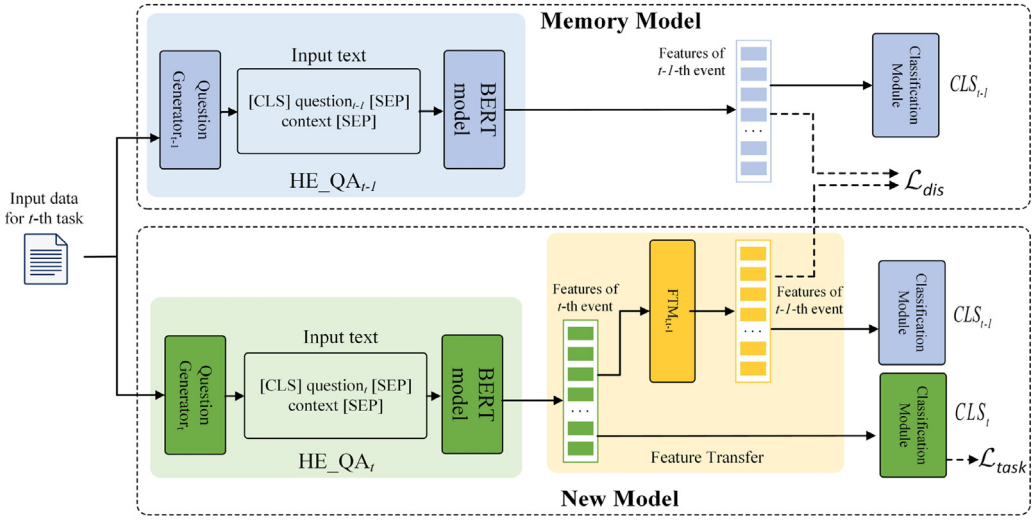


Fig. 3. The framework of the HEFT at the tth task, which consists of three main components: (a) historical-enhanced question answering module (HE_QA$_{t-1}$, HE_QA$_t$) as feature extractor; (b) feature transfer module (FTM$_{t,t-1}$, where the subscripts indicate transferring the feature from tth to $t-1$th task); (c) classification module (CLS_{t-1}, CLS_t). We copy and froze the model after training $t-1$th event as the memory model (on the top side). When learning for tth event, a FTM$_{t,t-1}$, and a CLS_t are added to original HE_QA$_t$ module to obtain a new model (on the bottom side). The classification loss $\mathcal{L}_{task}$ is leveraged on the tth task, and the transfer distillation loss $\mathcal{L}_{dis}$ is used for reconstructing the features at the $t-1$th task. Best view in colour.

from checkpoints for neural machine translation. However, as illustrated in the introduction, existing knowledge distillation-based methods are faced with two crucial problems, including: reserving/adjusting conflict and historical category information utilization. To overcome those issues, we introduce the feature transfer module to recover the features from the previous task, and propose the history-enhanced question answering module to implicitly incorporate history category information.

3. Proposed methodology

The details of our proposed HEFT framework are illustrated in this section. We first briefly introduce the overall architecture and the incremental learning workflow of HEFT. Then, we separately present the components designed in HEFT, mainly including the historical information-enhanced question answering module, feature transfer module, and classification module.

3.1. Overview

Fig. 3 illustrates the architecture of HEFT. We leverage the pre-trained language model BERT [36] as the backbone of the history-enhanced question answering (HE_QA) module and adopt it as a feature extractor. After extracting the features from the new task and the adjacent previous task (according to the order of tasks), a feature transfer module (FTM) and a task-specific classification module (*CLS*) are built. The role of the feature transfer module is to precisely map output features from the new task to the old one. As suggested in [37], we formulate incremental event detection as a binary classification task to predict whether each word in a sentence belongs to a specific observed category. Please note that [37] is a "regularization-based" method that overcomes the catastrophic forgetting problem by parameter transferring in initialization. Besides, [37] proposed a post-processing step to reduce the negative effects on classification in the testing stage. However, HEFT is a "knowledge distillation-based" method, which transfers the knowledge from seen classes to the new class by utilizing the knowledge distillation mechanism, and it does not perform any post-operation during the testing stage.

The process of learning a new task in HEFT is briefly introduced as follows: assuming that the model has already trained well on the $t-1$th task and next to train for the tth one, as shown in Fig. 3, we copy and freeze the parameters of the previous model (on the top side) as the Memory Model and use the output features from memory model to guild the New Model (on the bottom side). After that, a feature transfer model (FTM$_{t,t-1}$, where the subscripts indicate transferring the feature from tth to $t-1$th task) and a new classification model (CLS_t) are concatenated to the new model.

During the training process, classification loss $\mathcal{L}_{task}$ is used for learning the new task. Moreover, at the feature level, transfer distillation loss $\mathcal{L}_{dis}$ is used to train the feature transfer module and minimize the gaps between the output features of the Memory Model and New Model. To benefit from the knowledge from old classes, we also store a portion of samples from the previous task as memory data, and combine them with all the samples from the current task as input data for training. The specific training process is shown in Algorithm 1.

The testing process of two successive tasks is as follows: firstly, we discard the Memory Model and the stored memory data, and only keep the New Model. Secondly, we take all observed test samples before the tth task as the testing set, and input them to New Model to get the corresponding output features. Thirdly, to classify whether the input samples belong to the new task, we directly input them to CLS_t and get the classification output. To identify if those inputs belong to the old tasks, we feed them to FTM$_{t,t-1}$ and adopt the previous classification module CLS_{t-1} to get trigger predictions. When the network continues to learn new tasks, we repeat the above steps and keep all the trained feature transfer modules and classification modules. Every pair of feature transfer module and classification module are responsible for classifying a specific event category. Specifically, as shown in Fig. 4, after learning T tasks, the model contains: (1) a history-enhance question answering model at Tth task as feature extractor; (2) $T-1$ feature transfer modules FTM$_{t_i,t_i-1}$, where $2 \leq t_i \leq T$; (3) T task-specific classifiers CLS_{t_i}, which classify the features of t_ith task to get the final predictions, where $1 \leq t_i \leq T$. When it is required to learn the $T+1$th task, we create a new feature transfer module (FTM$_{T+1,T}$) and a new classification module (CLS_{T+1}), then concatenate them to all the previous T feature transfer modules and classification modules.

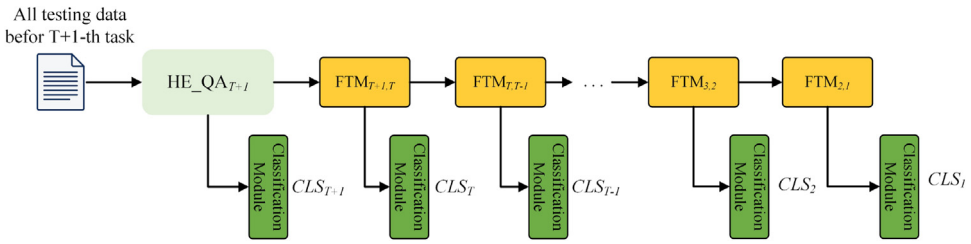


Fig. 4. The testing process of our approach HEFT, which only contains one feature extractor HE_QA$_{T+1}$. When learning the T+1th task, we concatenate a new feature transfer module (FTM$_{T+1,T}$) and a new classification module (CLS_{T+1}) to all previous FTM and CLS modules.

Algorithm 1 The process of training at tth task in HEFT.

Input: The input sentences $\mathcal{C}^{(t)}$ which contains both current task data and memory data, the weights w_{old} of the well-trained Memory Model, balance factor λ, learning rate l, and the number of epoches n.

Output:
New Model that well classify on new and old tasks.

Establish a HE_QA$_{t-1}$ model and a CLS_{t-1} as Memory Model based on w_{old}.
Generate question generator$_t$ containing all observed classes and generate new HE_QA$_t$.
Copy the weights w_{old} to w_{new}, and establish New Model based on w_{new}.
Extend FTM$_{t,t-1}$ and CLS_t to the New Model.
for $i = 1$ *to* n do
    Calculate loss $\mathcal{L}_{task}$ using the outputs of the New Model.
    Calculate loss $\mathcal{L}_{dis}$ using the outputs of both the New Model and the Memory Model.
$\mathcal{L} = \mathcal{L}_{task} + \lambda\mathcal{L}_{dis}$
Update $w_{new} \leftarrow w_{new} - l\frac{\partial\mathcal{L}}{\partial w_{new}}$
Discard Memory Model
return New Model;

It should be noted that different from the regular model ensemble strategies (e.g., bagging or stacking) which parallelly integrate diverse (or same but trained with different random seeds) models during the test stage, HEFT seeks to reconstruct the features of the previous tasks by serially concatenating the history-enhanced question answering module and the feature transfer modules.

3.2. Task formulation

We leverage the MRC-based QA model as the feature extractor and formulate incremental event detection as a binary word classification task.

Formally, when continuing to learn a new class at tth task, the training data of the new classes is denoted as $\mathcal{N}^{(t)} = \{q_i^{(t)}, c_i^{(t)}, \mathcal{Y}_i^{(t)}, 1 \le i \le K\}$, where $q_i^{(t)}$ and $c_i^{(t)}$ refer to the question and raw input context, $\mathcal{Y}_i^{(t)}$ is the trigger word category set of the ith context, and K indicates the number of training data belonging to tth class. In practice, the question $q_i^{(t)}$ could be any description about the desired trigger. For a given context $c_i^{(t)} = \{w_j\}_{j=1}^{n}$ consisting of n words, we seek to identify a set of trigger words $\mathcal{Y}_i^{(t)} = \{(y_k)\}_{k=1}^{m}$, where y_k is the role of the kth trigger word, and m denotes the total number of triggers in the given sentence. And we combine the stored data $\mathcal{M}^{(t-1)}$ with the new training data, denoted as $\mathcal{C}^{(t)} = \mathcal{N}^{(t)} \cup \mathcal{M}^{(t-1)}$, to train the new model. For the testing stage, we discard the stored data $\mathcal{M}^{(t-1)}$ and test the model with all the observed testing data before tth task (denoted as $\mathcal{T}^{(all)} = \cup_{i=1}^{t}\mathcal{T}^{(i)}$, where $\mathcal{T}^{(i)}$ is the testing set of the ith task).

3.3. History-enhanced question answering module

In this section, we introduce the history-enhanced question answering module (HE_QA) as the feature extractor for incremental event detection, which consists of two components: a question generator module and a question answering module. To solve the problem that existing incremental learning models do not make sufficient use of historical category knowledge, we leverage different question generators to incorporate such information into the question templates.

Question Answering Module: considering that recently deep transformer architecture [38] has achieved great performance, we employ pre-trained language model BERT [9] as the backbone of HE_QA. We leverage a template-based question generation strategy to extract the desired event trigger. The input text template could be formulated as follows:

$$[CLS]\ question_t\ [SEP]\ context\ [SEP]$$

where $[CLS]$ and $[SEP]$ are special tokens defined in BERT; $question_t$ refers to the tth task query generated with manually-designed template, such as "trigger", "verb", "what is the trigger" (we set "verb" as our default question template); *context* denotes the raw context sentence that is going to be extracted.

After constructing the input text, the input sequence is tokenized and converted into an embedding matrix. For those words containing more than one token, we keep the first token in this word. Formally, for every input text, we leverage the hidden state of BERT last layer to represent every token:

$$H_{BERT} = \{H_{[CLS]}, H_q, H_{[SEP]}, H_c, H_{[SEP]}\}. \quad (1)$$

where $H_q \in \mathbb{R}^{p\times d}$ and $H_c \in \mathbb{R}^{n\times d}$ denote the output of the BERT hidden layer from question part and context part for the sentence. p and n represent the length of the token in the question and the length of the context, respectively. d is the dimension number of hidden layers in the BERT model.

Question Generator: the key of question answering module is how to generate questions that could adequately describe the extracted targets. To make full use of the historical category information, we add additional clues to existing MRC-based QA model $question_t$ template and use the following question template for extracting the tth task trigger words:

verb with known class $[class_1]$ and $[class_2]$ and ...

where $[class_1]$ and $[class_2]$ denote the observed category label of the first and the second training class, respectively. After completing training a new task, we concatenate the category information of the task to the question template. For the first task that contains no observed label information, we fill in the template with *none* class.

Since BERT encoding stage makes a deep fusion between the question and the context by interactions between multi-head and multi-layer attentions. The QA-based model could be implicitly aware of such information by inserting the historical categories into the question. As a result, the model will be more likely to correctly classify old class data, thus alleviating the problem of

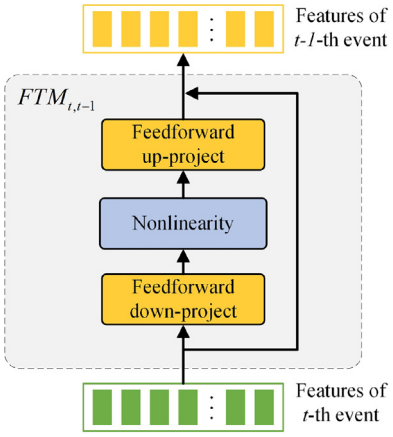


Fig. 5. The architecture of the feature transfer module, where the feed-forward up-project and the feed-forward down-project are fully connected layers. $FTM_{t,t-1}$ indicates feature transfer module mapping event features from the tth to the $t-1$th task.

catastrophic forgetting. Besides, for the samples that belong to different categories but have similar distributions, HE_QA could pull apart their output features during training, making them easier to be correctly distinguished.

3.4. Feature transfer model

In order to alleviate the reserving/adjusting conflict problem from existing knowledge distillation-based approaches, we propose a simple yet effective feature transfer module to recover the feature space of the previous task.

Fig. 5 shows the architecture of the feature transfer module and its applications for two adjacent tasks. We adopt a bottleneck architecture consisting of two fully-connected layers and a non-linearity. It first projects the original d-dimensional features to a smaller dimension h, then passes through the non-linearity and finally projects back to d-dimension. By setting $h \ll d$, we could limit the number of parameters added to every task. A skip-connection is also leveraged in it for training stability.

Formally, when learning at tth task, given a feature transfer module $FTM_{t,t-1}$, which indicates mapping features from the feature space of task t to the feature space of task $t-1$, the process can be described as:

$$H_c^{(t-1)} = FTM_{t,t-1}(H_c^{(t)}) \tag{2}$$

where $H_c^{(t-1)}$ and $H_c^{(t)}$ denote the output features of context part from BERT output of task $t-1$ and task t. It should be noted that only $H_c^{(t-1)}$ and $H_c^{(t)}$ from the text inputs $H_{BERT}^{(t-1)}$ and $H_{BERT}^{(t)}$ are used for getting the prediction, and the other parts (e.g., questions) are masked out.

When transferring from the newly trained task to the previously learned task, because the input and the output of feature transfer modules have the same size and each feature transfer module enables transfer from adjacent task feature spaces, we can implement feature transfer between any previous task by concatenating all FTM modules. The formula for sequential mapping of feature transfer modules is:

$$H_c^{(t_i)} = FTM_{t_i,t_i-1}...FTM_{t_{j-1},t_{j-2}}FTM_{t_j,t_{j-1}}(H_c^{(t_j)}) \tag{3}$$

where t_i and t_j indicate different tasks, and $t_i < t_j$.

In addition, as shown in Fig. 3, each feature transfer module is followed by a task-specific classification module. We use a fully-connected layer to get the final prediction and adopt sigmoid function to predict whether each word in the given sentence belongs to the specific observed class. For the words in the context part of input in tth task, we could get the prediction $P^{(t)}$:

$$P^{(t)} = sigmoid(CLS_t(H_c)) \qquad = sigmoid(W_t(H_c)) \tag{4}$$

where $W_t \in \mathbb{R}^{d\times 2}$ is the transformation matrix at tth task.

We leverage binary cross-entropy loss between the prediction and golden labels as the training criteria to train our model:

$$\mathcal{L}_{task} = BCE(P^{(t)}, \hat{Y}^{(t)}) \tag{5}$$

where BCE is short for the binary cross-entropy loss and $\hat{Y}^{(t)}$ is the golden binary label matrix for the tth task.

3.5. Training

In order to memorize more historical knowledge from old classes, we adopt a prototype enhanced retrospection mechanism [3] to store samples belonging to previous classes as memory data. It divides the samples of each previous category equally and leverages the prototype to select and remove training samples. We combine the stored data with the new class training data to train HEFT.

To allow the output features of the new task map to the feature space of the previous task after FTM, we adopt the knowledge distillation mechanism [35]. It leverages the teacher–student framework to transfer the knowledge from the teacher to the student. Specifically, we set the old feature extractor HE_QA_{t-1} as the teacher and then generate the features $H_c^{(t-1)}$. Likewise, the features $H_c^{(t)}$ could be obtained from the HE_QA_t model. The HE_QA_t distills knowledge through learning to have similar output features as HE_QA_{t-1}. We adopt cosine similarity loss to optimize the model, and the specific implementation process of the transfer distillation loss $\mathcal{L}_{dis}$ is as follows:

$$\mathcal{L}_{dis} = CS(\bar{H_c}^{(t-1)}, \bar{H_c}^{(t)}) \tag{6}$$

where CS is short for the cosine similarity loss. $\bar{H_c}^{(t-1)}$ and $\bar{H_c}^{(t)}$ are the L2-normalized representations of context hidden state, respectively.

Combining the classification loss and the transfer distillation loss, the overall loss of HEFT can be optimized as follows:

$$\mathcal{L} = \mathcal{L}_{task} + \lambda\mathcal{L}_{dis} \tag{7}$$

where λ is the balance coefficient to control the impact of each component.

4. Experiments

4.1. Dataset and evaluation metrics

We evaluate our model on the ACE 2005 and TAC KBP 2017 corpora. For the ACE 2005 benchmark, we use the dataset split as suggested in [39–42]. The data split includes 529 documents used for the training set, 30 documents used for the development set, and the rest 40 documents used for the test set. The TAC KBP 2017 benchmark is the official evaluation data from the Event Nugget Detection Evaluation of the Text Analysis Conference (TAC). We adopt the official training, development, and testing data of TAC KBP 2017. The specific statistic information of the ACE 2005 and TAC KBP 2017 corpora is listed in Table 1.

Due to the long-tail frequency distribution problem, following [3], we use the top 10 most frequent classes for a given event detection dataset, and the classes are arranged in a fixed order. Each method is trained in a class-incremental way on the new class training data, and tested on all the observed test data. For both of the two benchmarks, one class is available at each time

Table 1
The statistics of the ACE 2005 and TAC KBP 2017 benchmarks, where numbers indicate the number of samples..

	Training	Development	Testing
ACE 2005	2638	306	211
TAC KBP 2017	10085	1278	2154

as the training progress. We report ***Average F1*** and ***Whole F1*** metrics for incremental event detection. After training on the new class, we report the averaged F1 scores on the test data that belongs to all observed classes. For instance, after training at the tth task, and the result is denoted as F_t, we calculate the average F1 metric by averaging the F1 scores from the first task to the tth task (i.e. $\frac{1}{t}\sum_{i=1}^{t} F_i$). For the whole F1 metric, we calculate the F1 score on the whole testing data of all classes.

4.2. Experimental setups

We leverage the pre-trained BERT-base model from HuggingFace's Transformers library[1] as the backbone of our MRC-based QA model and adopt it to extract features for different tasks. We use a bottle-neck architecture for the feature transfer module consisting of two fully-connected layers with the hidden size of 35 and a Tanh non-linearity. We set the batch size as 32, the balance coefficient λ as 10, and each task is trained for 25 epochs. We fix the maximum sentence length as 90 by cutting off longer ones and padding shorter sentences. For a fair comparison, we keep the same data order and the capacity of history data storage (100 and 500 for the two benchmarks) as in [3]. The experiment adopts the stochastic gradient descent algorithm and the AdamW update rule [43] to minimize the loss functions. The gradients are calculated by back-propagation. We use a differential learning rates strategy for training, where the learning rate of BERT model and the feature transfer module are set as 0.00002 and 0.00005, respectively. All experiments are performed on one RTX-3090 GPU.

4.3. Baselines

We compare our approach HEFT with the following strong baselines methods:

EWC: this method is proposed by [19], which is the representative work of regularization-based methods. It allows model parameters to be constrained to change from the perspective of the interference of the new task data and prevent changes in essential parameters on the old task.

EMR: this method is proposed by [5], which is the representative work of replay-based methods. It seeks to alleviate catastrophic forgetting by randomly storing some samples of old classes.

EA-EMR: this method is an extended version of [5]. It stores samples and leverages a simple transformation to explicitly constrain the features of neural models by expanding several alignment models using one-class learning.

LwF: this method is proposed by [4], which is the representative work of distillation-based methods. It tries to leverage the knowledge distillation mechanism to match the softmax output of the network of previous models for old classes.

KCN: this method is proposed by [3], which combines the benefits of replay-based and distillation-based methods. It stores some previous samples by prototype enhanced retrospection mechanism and leverages hierarchical distillation to further reserve history information.

[1] https://github.com/huggingface/transformers.

Table 2
The overall performance of average F1 (%) on all observed classes ("Avg" column), and whole F1 (%) on the whole testing data ("Whole" column) after the last time step.

Method	ACE		TAC KBP	
	Avg	Whole	Avg	Whole
Finetune	19.5	2.7	15.4	2.2
EWC	23.4	3.6	17.7	4.8
LwF	47.7	11.2	33.1	17.4
EMR	55.3	36.1	35.5	21.5
EA-EMR	67.0	54.0	50.2	28.1
KCN	65.9	55.1	44.3	34.9
HEFT	**73.5**	**61.3**	**59.5**	**44.0**

Finetune: this method simply finetunes the pre-trained model on new arriving data, where we adopt the commonly-used model (i.e., BERT) for finetuning.

UpperBound: this method train the model with the data that belongs to all observed categories, which provides the upper bound of the model.

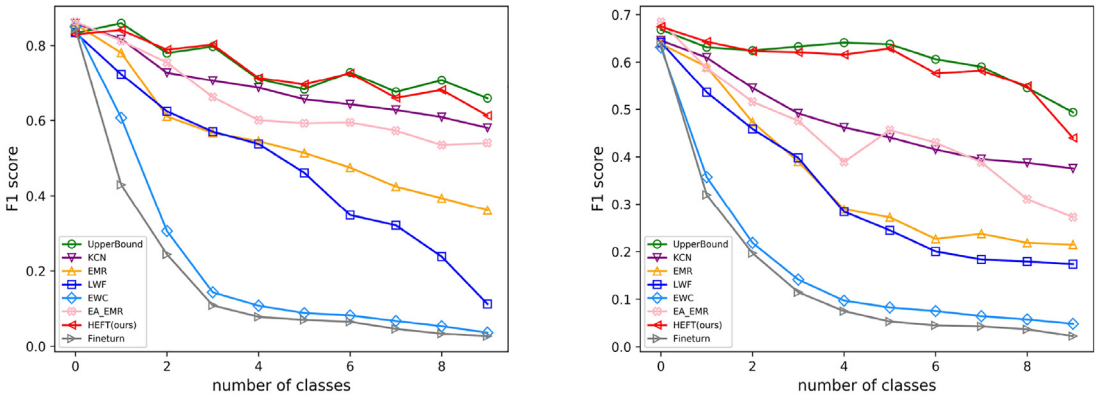
4.4. Overall performance

The overall performance of average F1 and whole F1 results are listed in Table 2. From the results, we can get the following observations: (1) our method outperforms all the baseline models by a large margin on both two benchmarks. Compared with the state-of-the-art method KCN, our method achieves 6.2% and 9.1% improvements in the whole F1 score on ACE 2005 and TAC KBP 2017 benchmarks, respectively. It shows the effectiveness of our model for the incremental event detection task. (2) there is a huge improvement between HEFT and traditional knowledge distillation-based methods (e.g., KCN and LwF). It shows that HEFT is able to alleviate the reserving/adjusting conflict problem in conventional knowledge distillation-based methods. (3) both HEFT and EA-EMR leverage feature transfer (alignment) modules at the feature level. EA-EMR uses a regulation mechanism to restrict the output features, but HEFT seeks to resume the features of the previous task. The experiments show that EA-EMR gets 7.3% and 15.9% decrease on the whole F1 metric than HEFT on two benchmarks, respectively. The results verify our hypothesis that assuming a simple transformation as a regulation term is not well suited for incremental event detection, and transferring the feature spaces between adjacent tasks is more effective.

We also draw folding line charts of the F1 scores over all observed classes during the incremental learning process, as shown in Fig. 6, and we find that: (1) at the start of the training process, HEFT and the state-of-the-art method KCN has similar experimental results. However, HEFT outperforms KCN on subsequent tasks, demonstrating that HEFT is superior to KCN in overcoming the catastrophic forgetting problem. (2) HEFT achieves comparable experimental performance to UpperBound, which reserves all observed training data. It indicates that HEFT could implicitly learn history information instead of heavily relying on the knowledge brought by stored history samples.

4.5. Ablation study

We conduct ablation studies to investigate the effectiveness of the history-enhanced question answering mechanism and the feature transfer module. The experiment results are shown in Table 3, where the "w/o FTM" represents removing the feature transfer module, and modelling incremental event detection as a binary word classification task. Besides, "w/o HE_QA" indicates deleting the history-enhanced question answering module and using the regular question answering module to detect



(a) ACE benchmark (b) TAC KBP benchmark

Fig. 6. The performance on (a) ACE benchmark and (b) TAC KBP benchmark. Our approach HEFT achieves better performance than other incremental learning methods and approaches to the upper bound method.

Table 3
The ablation study of HEFT, where w/o indicates without such component.

Method	ACE		TAC KBP	
	Avg	Whole	Avg	Whole
KCN	65.9	55.1	44.3	34.9
Full HEFT	**73.5**	**61.3**	**59.5**	**44.0**
w/o FTM	67.3	56.8	52.5	37.6
w/o HE_QA	72.8	58.9	57.9	41.2
w/o FTM w/o HE_QA	65.0	54.5	51.4	35.0

events. We leverage "Verb" as the default question template in the experiments.

From the experiments, we can observe meaningful patterns: (1) removing either feature transfer module or history-enhanced question answering module leads to significant declines in average F1 and whole F1 metrics, which indicates that both components are effective. (2) compared to the model without FTM (w/o FTM), HEFT further improves the whole F1 metric from 56.8% to 61.3% on the ACE benchmark. This result illustrates that the ability of fine-tuning on the classification module is limited. Furthermore, by mimicking the feature output from the previous model, the feature transfer module preserves more history information. (3) after removing the history-enhanced question answering module (w/o HE_QA), the whole F1 metric of HEFT drops from 61.3% to 58.9% on the ACE benchmark. It demonstrates that adding extra history category information to the question part of a question answering module is an effective way to preserve more historical information, which helps to alleviate catastrophic forgetting.

4.6. Effect of balance factors and hidden size

There are two crucial hyperparameters that have a crucial impact on the final results: the balance factor λ and the hidden layer size h of feature transfer module. Between the two, λ controls the proportion of information from previous classes, and h determines the learning ability of the feature transfer module. In order to illustrate the impact of the two hyperparameters and find the best hyperparameters setting, we further conduct experiments by setting the different combinations of balance factors λ and the hidden layer size h.

The experiment for the values of balance factor and hidden layer size is conducted on ACE 2005 benchmark. From the results in Table 4, we can observe that: (1) the difference on the whole F1 metric between the highest and lowest results is greater than 6 points. It demonstrates that various combinations of the two hyperparameters have a significantly different impact on final performances. (2) for the situation where hidden layer size h is equal to 10, since the hidden size is relatively small and the learning capacity of the fully connected layer is limited, the feature transfer module could not well achieve feature transfer between two adjacent tasks. Besides, the average F1 metric of the model decreases as the balance factor λ increases. The results indicate that the under-fitting problem makes the model hard to recover previous feature space. (3) for the case where hidden layer size h is equal to 50, the performance is worse than that of h equals 40. An explanation is that the ACE 2005 benchmark has a relatively small data size, and excessive hidden layer size would likely cause the over-fitting problem. (4) likewise, when hidden layer size h is 35, paying less attention to historical knowledge (λ equals (5) and excessive focusing on historical information (λ equals 15) will contribute to performance degradation. By setting balance factor λ as 10, the model reaches the best performance on both average F1 and whole F1 metrics.

4.7. Effect of templates

In this section, we compare the performance of the HEFT with different templates. Besides, we also compare the output performance with or without history category information in the question answering module. The experiment is conducted on the ACE 2005 benchmark, and the results are listed in Table 5. We observe that: (1) similar to [44] that investigates the impact of different question templates, we do not find a substantial gap between different templates. And by using the "Verb" as the question template, HEFT achieves the best results on the evaluation metrics. (2) under various question templates, those QA models containing history information tend to have better results than those QA models without such information. It indicates that history category information is effective to alleviate catastrophic forgetting. Besides, it further shows that by adding the history category information to the question, the history-enhanced question answering module is able to implicitly incorporate and preserve such historical knowledge.

4.8. Effect of the number of reserved samples

Reserving a few samples has been proven to be effective for many NLP tasks [5,25]. Table 6 compares our approach with the state-of-the-art KCN model reserving a different number of samples per class on the ACE 2005 benchmark. From the results, we observe that: (1) the more samples that are reserved in the memory, the better performance of both KCN and our approach

Table 4
Experiment results of ablation study for different hidden layer size h and balance factor λ settings on ACE 2005.

Hidden layer size h	10		20		35		40		50	
Balance factor λ	Avg	Whole	Avg	Whole	Avg	Whole	Avg	Whole	Avg	Whole
5	69.6	54.6	73.3	56.5	72.8	57.1	71.8	55.8	70.4	55.5
10	69.1	54.8	73.1	59.3	**73.5**	**61.3**	71.8	57.0	70.8	56.0
15	68.3	54.2	71.4	57.8	72.6	56.6	72.6	58.0	71.5	56.8

Table 5
Experiment results of different question templates with or without history information on ACE 2005.

Template	With history information		w/o history information	
	Avg	Whole	Avg	Whole
Empty	72.1	57.9	71.7	56.4
Trigger	71.8	57.6	71.1	56.5
What happened	72.6	58.6	**73.2**	57.8
What is the trigger	72.9	58.8	72.2	56.1
Verb	**73.5**	**61.3**	72.8	**58.9**

Table 6
The effect of the number of reserved samples. We compare our method HEFT with the state-of-the-art method KCN on the ACE 2005 benchmark.

	KCN		HEFT	
	Avg	Whole	Avg	Whole
1	41.7	40.0	62.0	48.6
10	64.2	53.7	69.7	55.1
20	65.4	55.3	70.7	57.8
30	66.5	56.8	72.8	59.1
40	67.4	57.7	72.9	62.1
50	68.2	59.1	73.7	63.6
60	**69.1**	**60.8**	**74.2**	**64.9**

Table 7
The time/space complexity comparison between the state-of-the-art model KCN and our proposed HEFT framework. Whole(.) denotes the whole F1 metric.

	FLOPs (GMac)	Parameters (M)	Whole (ACE)	Whole (TAC KBP)
KCN	271.86	109.48	55.1	34.9
HEFT	290.44	110.12	61.3	44.0

HEFT, it verifies the superiority of replay-based methods. (2) in each case, the result of HEFT is superior to the KCN model, which shows HEFT could more effectively preserve historical information under the same scenario when retaining the same number of historical samples. (3) for the situation where reserve few samples, HEFT has a significant performance improvement over KCN. Especially for the situation when reserving only one sample, our method has a 20.3% improvement on average F1, and the improvement ratio is more than 48%. It demonstrates that HEFT does not heavily rely on the history information provided by the stored memory data, and could preserve more historical category information.

4.9. Effect of feature transfer module

To further study the effectiveness of the feature transfer module, we conduct visualization experiments on the output features. For the experiments, we leverage mean-pooling operations over the sentences to obtain the sentence representation and then leverage t-distributed Stochastic Neighbour Embedding (t-SNE) [45] to extract the principal components. We input the same samples to the models trained after different tasks. Fig. 7 visualizes the experiment results of two cases on three kinds of features, including the shift features transferred by the feature transfer module of the current model, the features extracted by the feature extractor of the current model (i.e., without feature transfer), and the features output by the feature extractor of the previous model.

From the experiments, we observe that: as shown in Fig. 7, since many parameters have been adjusted during training, the positions of the same data points shift significantly in the feature space after passing through the previous model and current model, which makes it difficult for the classification module to accurately classify. Furthermore, such phenomenon could be more severe when continuing training in new classes, and finally results in catastrophic forgetting. Nevertheless, through FTM, the output features of the current model are transformed back to the features of the old model, and the distance between the features after FTM and the features of the previous model is relatively close. By using such recovered features, the classifier is able to classify the triggers correctly, thus alleviating the forgetting problem. By visualizing the features, we show the effectiveness of the proposed feature transfer module. It is qualified to transfer the features from the new model back to the previous one even if the output features drift a lot.

4.10. Time/space complexity exploration

We conduct the experiment to demonstrate the time and space complexity of the state-of-the-art model KCN and the proposed HEFT. Specifically, for time complexity, we utilize the floating point operations (FLOPs) [46], a widely-used evaluation criteria to measure the calculation volume of the algorithm/model, which could indirectly reflect the time complexity. In general, the smaller the FLOPs value is, the less computation cost is needed. Meanwhile, for space complexity, the number of model parameters is leveraged as the evaluation metric. We utilize the open-source toolkit ptflops[2] to obtain the FLOPs and parameters of the models. From the experiment results in Table 7, we could observe that: compared to KCN, since HEFT introduces the feature transfer modules to recover the features from the previous tasks, it increases the amount of computation cost by a small amount (the relevant increase is only 6 points). However, the parameters of HEFT are nearly the same as KCN, and HEFT consistently outperforms KCN on the ACE 2005 and TAC KBP 2017 benchmarks. Such observations demonstrate the light-weighting advantage of the feature transfer modules and the effectiveness of HEFT.

4.11. Error analysis

Following the experiment settings in [47,48], we conduct an error analysis by sampling out 100 error cases from the test set of ACE 2005 corpus. From the experiment, we find three typical errors: (1) **Predicting multiple triggers**, which means when predicting a specific category, ground-truth label has only one trigger, but HEFT predicts multiple triggers. For example, the ground-truth trigger is "violence" for *Conflict.Attack* event type, but HEFT predicts "conflict" and "violence" as triggers. It accounts for 15%. (2) **Miscategorized in similar contexts**, which

[2] https://github.com/sovrasov/flops-counter.pytorch.

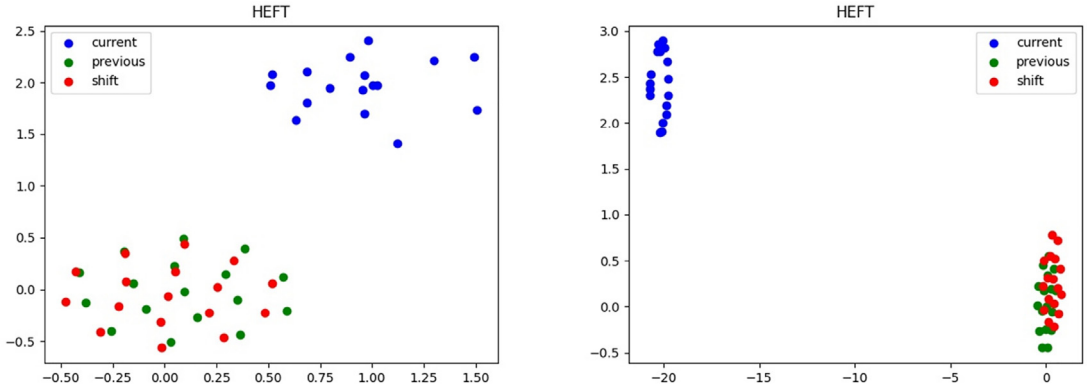


Fig. 7. The t-SNE visualization of the input sentence features, where we input the same samples to the models trained after different tasks. 'Shift' points denote the samples after passing through both feature extractor and FTM.

Table 8
Case study of state-of-the-art method KCN and our method HEFT on ACE 2005 benchmark.

Category	Example	Ground truth	KCN prediction	HEFT prediction
Reserving/adjusting conflict	Amid a chill in relations over the war in Iraq, which Canada opposed, Bush indefinitely postponed a visit to Canada, instead choosing to host Australian Prime Minster John Howard, who endorsed that military campaign.	Word: visit Index: 20 Class: Movement.Transport	Word: None Index: None Class: None	Word: visit Index: 20 Class: Movement.Transport
Historical category information utilization	Faisalabad's Catholic Bishop John Joseph, who had been campaigning against the law, shot himself in the head outside a court in Sahiwal district when the judge convicted Christian Ayub Masih under the law in 1998.	Word: shot Index: 16 Class: Life.Die	Word: shot Index: 16 Class: Life. Injure	Word: shot Index: 16 Class: Life.Die
	Welch also wants details on Jane Beasley Welch's salary, benefits, retirement plan and other compensation paid to her.	Word: retirement Index:14 Class: Personnel.End-Position	Word: None Index: None Class: None	Word: retirement Index: 14 Class: Personnel.End-Position

means some words have similar contexts, but HEFT does not identify correctly. For instance, the ground-truth trigger is "war", but HEFT predicts "conflict". It accounts for 5%. (3) **Location mismatch**, which means the same word appears several times in the sentence, but HEFT does not find the correct trigger position exactly. Those errors are owing to HEFT does not fully understand the sentence semantic. It accounts for 3%.

4.12. Case study

In this section, we illustrate how HEFT could alleviate reserving/adjusting conflict and historical category information utilization problems. As shown in Table 8, we get the following observations: (1) for the scenario of reserving/adjusting conflict in E1, as the training process proceeds, the features of tasks forget a certain level of history knowledge, finally making KCN miss to classify. However, HEFT leverage FTM to separate the reserving and adjusting sub-functions, and finally classify trigger word *visit* as *Movement.Transport* event type correctly. (2) for the historical category information utilization problem in E2, since the event types *Life.Die* and *Life. Injure* has similar distributions, it is confused for an ED system to classify, especially for the incremental event detection situation where not all observed training data is preserved. But HEFT employs the history-enhanced question answering mechanism to implicitly pull apart similar categories, which assists in solving this problem. (3) in E3, because multiple tasks have been learned from *End-position* to the new category, KCN gradually forgets the old information. Nevertheless, the HE_QA model implicitly reinforces the historical category information of the previous tasks, making HEFT correctly predicts the trigger word.

5. Conclusion

In this paper, we introduce a history-enhanced feature transfer framework, HEFT, to alleviate the catastrophic forgetting problem in incremental event detection. Specifically, we proposed a feature transfer module (FTM) to overcome the reserving/adjusting conflict in conventional knowledge-distillation-based methods by reconstructing the features of old classes. By using the reconstructed features to preserve history knowledge, the feature extractor and the followed classifier only need to focus on adjusting to the new task. Besides, a history-enhanced question answering (HE_QA) mechanism is further proposed to fully utilize the historical category information and pull apart tasks with similar distributions implicitly. We conduct experiments on the ACE 2005 and TAC KBP 2017 benchmarks in incremental event extraction and carry out extensive experimental results as well as empirical analyses, showing the effectiveness of HEFT.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Kim, A.M. Rush, Sequence-level knowledge distillation, in: J. Su, X. Carreras, K. Duh (Eds.), Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016, The Association for Computational Linguistics, 2016, pp. 1317–1327, http://dx.doi.org/10.18653/v1/d16-1139.
[2] H. Wei, S. Huang, R. Wang, X. Dai, J. Chen, Online distilling from checkpoints for neural machine translation, in: J. Burstein, C. Doran, T.

Solorio (Eds.), 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), NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019, pp. 1932–1941, http://dx.doi.org/10.18653/v1/n19-1192.

[3] P. Cao, Y. Chen, J. Zhao, T. Wang, Incremental event detection via knowledge consolidation networks, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, Association for Computational Linguistics, 2020, pp. 707–717, http://dx.doi.org/10.18653/v1/2020.emnlp-main.52.

[4] Z. Li, D. Hoiem, Learning without forgetting, 2016, CoRR abs/1606.09282, arXiv:1606.09282.

[5] H. Wang, W. Xiong, M. Yu, X. Guo, S. Chang, W.Y. Wang, Sentence embedding alignment for lifelong relation extraction, in: J. Burstein, C. Doran, T. Solorio (Eds.), 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), NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019, pp. 796–806, http://dx.doi.org/10.18653/v1/n19-1086.

[6] K. Wei, X. Sun, Z. Zhang, J. Zhang, Z. Guo, L. Jin, Trigger is not sufficient: Exploiting frame-aware knowledge for implicit event argument extraction, in: C. Zong, F. Xia, W. Li, R. Navigli (Eds.), Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, (Volume 1: Long Papers), ACL/IJCNLP 2021, Virtual Event, August 1-6, 2021, Association for Computational Linguistics, 2021, pp. 4672–4682, http://dx.doi.org/10.18653/v1/2021.acl-long.360.

[7] J. Liu, Y. Chen, K. Liu, W. Bi, X. Liu, Event extraction as machine reading comprehension, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, Association for Computational Linguistics, 2020, pp. 1641–1651, http://dx.doi.org/10.18653/v1/2020.emnlp-main.128.

[8] X.D. Wang, L. Weber, U. Leser, Biomedical event extraction as multi-turn question answering, in: E. Holderness, A. Jimeno-Yepes, A. Lavelli, A. Minard, J. Pustejovsky, F. Rinaldi (Eds.), Proceedings of the 11th International Workshop on Health Text Mining and Information Analysis, LOUHI@EMNLP 2020, Online, November 20, 2020, Association for Computational Linguistics, 2020, pp. 88–96, http://dx.doi.org/10.18653/v1/2020.louhi-1.10.

[9] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), 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), NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019, pp. 4171–4186, http://dx.doi.org/10.18653/v1/n19-1423.

[10] Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao, 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), Association for Computational Linguistics, Beijing, China, 2015, pp. 167–176, http://dx.doi.org/10.3115/v1/P15-1017, URL https://www.aclweb.org/anthology/P15-1017.

[11] T.H. Nguyen, K. Cho, R. Grishman, Joint event extraction via recurrent neural networks, in: K. Knight, A. Nenkova, O. Rambow (Eds.), NAACL HLT 2016, the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, The Association for Computational Linguistics, 2016, pp. 300–309, http://dx.doi.org/10.18653/v1/n16-1034.

[12] T.H. Nguyen, R. Grishman, Graph convolutional networks with argument-aware pooling for event detection, in: S.A. McIlraith, K.Q. Weinberger (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 5900–5907, URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16329.

[13] S. Liu, Y. Chen, K. Liu, J. Zhao, Exploiting argument information to improve event detection via supervised attention mechanisms, in: R. Barzilay, M. Kan (Eds.), Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers, ACL 2017, Vancouver, Canada, July 30 - August 4, Association for Computational Linguistics, 2017, pp. 1789–1798, http://dx.doi.org/10.18653/v1/P17-1164.

[14] Y. Chen, S. Liu, X. Zhang, K. Liu, J. Zhao, Automatically labeled data generation for large scale event extraction, in: R. Barzilay, M. Kan (Eds.), Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers, ACL 2017, Vancouver, Canada, July 30 - August 4, Association for Computational Linguistics, 2017, pp. 409–419, http://dx.doi.org/10.18653/v1/P17-1038.

[15] H. Yang, Y. Chen, K. Liu, Y. Xiao, J. Zhao, DCFEE: A document-level Chinese financial event extraction system based on automatically labeled training data, in: F. Liu, T. Solorio (Eds.), Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations, Association for Computational Linguistics, 2018, pp. 50–55, http://dx.doi.org/10.18653/v1/P18-4009, URL https://www.aclweb.org/anthology/P18-4009/.

[16] Y. Zeng, Y. Feng, R. Ma, Z. Wang, R. Yan, C. Shi, D. Zhao, Scale up event extraction learning via automatic training data generation, in: S.A. McIlraith, K.Q. Weinberger (Eds.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018, AAAI Press, 2018, pp. 6045–6052, URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16119.

[17] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: M.A. Walker, H. Ji, A. Stent (Eds.), Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Association for Computational Linguistics, 2018, pp. 2227–2237, http://dx.doi.org/10.18653/v1/n18-1202.

[18] S. Yang, D. Feng, L. Qiao, Z. Kan, D. Li, Exploring pre-trained language models for event extraction and generation, in: A. Korhonen, D.R. Traum, L. Màrquez (Eds.), Proceedings of the 57th Conference of the Association for Computational Linguistics, Volume 1: Long Papers, ACL 2019, Florence, Italy, July 28- August 2, 2019, Association for Computational Linguistics, 2019, pp. 5284–5294, http://dx.doi.org/10.18653/v1/p19-1522.

[19] J. Kirkpatrick, R. Pascanu, N.C. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, 2016, CoRR abs/1612.00796, arXiv:1612.00796.

[20] X. Liu, M. Masana, L. Herranz, J. van de Weijer, A.M. López, A.D. Bagdanov, Rotate your networks: Better weight consolidation and less catastrophic forgetting, in: 24th International Conference on Pattern Recognition, ICPR 2018, Beijing, China, August 20-24, 2018, IEEE Computer Society, 2018, pp. 2262–2268, http://dx.doi.org/10.1109/ICPR.2018.8545895.

[21] A.V.M. Barone, B. Haddow, U. Germann, R. Sennrich, Regularization techniques for fine-tuning in neural machine translation, in: M. Palmer, R. Hwa, S. Riedel (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, Association for Computational Linguistics, 2017, pp. 1489–1494, http://dx.doi.org/10.18653/v1/d17-1156.

[22] H. Khayrallah, B. Thompson, K. Duh, P. Koehn, Regularized training objective for continued training for domain adaptation in neural machine translation, in: A. Birch, A.M. Finch, M. Luong, G. Neubig, Y. Oda (Eds.), Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018, Association for Computational Linguistics, 2018, pp. 36–44, http://dx.doi.org/10.18653/v1/w18-2705.

[23] D. Varis, O. Bojar, Unsupervised pretraining for neural machine translation using elastic weight consolidation, in: F.E. Alva-Manchego, E. Choi, D. Khashabi (Eds.), Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 2: Student Research Workshop, Association for Computational Linguistics, 2019, pp. 130–135, http://dx.doi.org/10.18653/v1/p19-2017.

[24] B. Thompson, J. Gwinnup, H. Khayrallah, K. Duh, P. Koehn, Overcoming catastrophic forgetting during domain adaptation of neural machine translation, in: J. Burstein, C. Doran, T. Solorio (Eds.), 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), NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Association for Computational Linguistics, 2019, pp. 2062–2068, http://dx.doi.org/10.18653/v1/n19-1209.

[25] S. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, 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, IEEE Computer Society, 2017, pp. 5533–5542, http://dx.doi.org/10.1109/CVPR.2017.587.

[26] X. Han, Y. Dai, T. Gao, Y. Lin, Z. Liu, P. Li, M. Sun, J. Zhou, Continual relation learning via episodic memory activation and reconsolidation, in: D. Jurafsky, J. Chai, N. Schluter, J.R. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, Association for Computational Linguistics, 2020, pp. 6429–6440, http://dx.doi.org/10.18653/v1/2020.acl-main.573.

[27] C. de Masson d'Autume, S. Ruder, L. Kong, D. Yogatama, Episodic memory in lifelong language learning, in: H.M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.B. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver,

BC, Canada, 2019, pp. 13122–13131, URL https://proceedings.neurips.cc/paper/2019/hash/f8d2e80c1458ea2501f98a2cafadb397-Abstract.html.
[28] F. Sun, C. Ho, H. Lee, LAMOL: language modeling for lifelong language learning, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020, URL https://openreview.net/forum?id=Skgxcn4YDS.
[29] M. Mancini, E. Ricci, B. Caputo, S.R. Bulò, Adding new tasks to a single network with weight transformations using binary masks, in: L. Leal-Taixé, S. Roth (Eds.), Computer Vision - ECCV 2018 Workshops - Munich, Germany, September 8-14, 2018, Proceedings, Part II, in: Lecture Notes in Computer Science, vol. 11130, Springer, 2018, pp. 180–189, http://dx.doi.org/10.1007/978-3-030-11012-3_14.
[30] A. Mallya, S. Lazebnik, PackNet: Adding multiple tasks to a single network by iterative pruning, in: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, IEEE Computer Society, 2018, pp. 7765–7773, http://dx.doi.org/10.1109/CVPR.2018.00810, URL http://openaccess.thecvf.com/content_cvpr_2018/html/Mallya_PackNet_Adding_Multiple_CVPR_2018_paper.html.
[31] J. Serrà, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: J.G. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, ICML 2018, StockholmsmäSsan, Stockholm, Sweden, July 10-15, 2018, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 4555–4564, URL http://proceedings.mlr.press/v80/serra18a.html.
[32] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, 2016, CoRR abs/1606.04671, arXiv:1606.04671.
[33] J. Yoon, E. Yang, J. Lee, S.J. Hwang, Lifelong learning with dynamically expandable networks, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018, URL https://openreview.net/forum?id=Sk7KsfW0-.
[34] J. Xu, Z. Zhu, Reinforced continual learning, in: S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, 2018, pp. 907–916, URL https://proceedings.neurips.cc/paper/2018/hash/cee631121c2ec9232f3a2f028ad5c89b-Abstract.html.
[35] G.E. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, 2015, CoRR abs/1503.02531, arXiv:1503.02531.
[36] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, 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), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186, http://dx.doi.org/10.18653/v1/N19-1423, URL https://www.aclweb.org/anthology/N19-1423.
[37] W. Hu, Q. Qin, M. Wang, J. Ma, B. Liu, Continual learning by using information of each class holistically, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, the Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021, AAAI Press, 2021, pp. 7797–7805, URL https://ojs.aaai.org/index.php/AAAI/article/view/16952.
[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 2017, pp. 5998–6008, URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
[39] Q. Li, H. Ji, L. Huang, 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, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, The Association for Computer Linguistics, 2013, pp. 73–82, URL https://www.aclweb.org/anthology/P13-1008/.
[40] Y. Chen, L. Xu, K. Liu, D. Zeng, J. Zhao, 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 of the Asian Federation of Natural Language Processing, , Volume 1: Long Papers, ACL 2015, July 26-31, 2015, Beijing, China, The Association for Computer Linguistics, 2015, pp. 167–176, http://dx.doi.org/10.3115/v1/p15-1017.
[41] Y. Lu, H. Lin, X. Han, L. Sun, Distilling discrimination and generalization knowledge for event detection via delta-representation learning, in: A. Korhonen, D.R. Traum, L. Màrquez (Eds.), Proceedings of the 57th Conference of the Association for Computational Linguistics, Volume 1: Long Papers, ACL 2019, Florence, Italy, July 28- August 2, 2019, Association for Computational Linguistics, 2019, pp. 4366–4376, http://dx.doi.org/10.18653/v1/p19-1429.
[42] J. Liu, Y. Chen, K. Liu, J. Zhao, Neural cross-lingual event detection with minimal parallel resources, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019, Association for Computational Linguistics, 2019, pp. 738–748, http://dx.doi.org/10.18653/v1/D19-1068.
[43] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, la, USA, May 6-9, 2019, OpenReview.net, 2019, URL https://openreview.net/forum?id=Bkg6RiCqY7.
[44] X. Du, C. Cardie, Event extraction by answering (Almost) natural questions, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020, Association for Computational Linguistics, 2020, pp. 671–683, http://dx.doi.org/10.18653/v1/2020.emnlp-main.49.
[45] L. van der Maaten, Accelerating t-SNE using tree-based algorithms, J. Mach. Learn. Res. 15 (1) (2014) 3221–3245, URL http://dl.acm.org/citation.cfm?id=2697068.
[46] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings, OpenReview.net, 2017, URL https://openreview.net/forum?id=SJGCiw5gl.
[47] J. Liu, Y. Chen, J. Xu, Machine reading comprehension as data augmentation: A case study on implicit event argument extraction, in: M. Moens, X. Huang, L. Specia, S.W. Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, Association for Computational Linguistics, 2021, pp. 2716–2725, http://dx.doi.org/10.18653/v1/2021.emnlp-main.214.
[48] Z. Zhang, X. Kong, Z. Liu, X. Ma, E.H. Hovy, A two-step approach for implicit event argument detection, in: D. Jurafsky, J. Chai, N. Schluter, J.R. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020, Association for Computational Linguistics, 2020, pp. 7479–7485, http://dx.doi.org/10.18653/v1/2020.acl-main.667.