Semi-supervised Auto-encoder Based Event Detection in Constructing Knowledge Graph for Social Good

Yue Zhao
Xiaolong Jin
Yuanzhuo Wang
Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology, CAS
School of Computer and Control Engineering, The University of CAS
{zhaoyue,jinxiaolong,wangyuanzhuo,cxq}@ict.ac.cn

ABSTRACT

Knowledge graphs have recently been extensively applied in many different areas (e.g., disaster management and relief, disease diagnosis). For example, event-centric knowledge graphs have been developed to improve decision making in disaster management and relief. This paper focuses on the task of event detection, which is the precondition of event extraction for constructing event-centric knowledge graphs. Event detection identifies trigger words of events in the sentences of a document and further classifies the types of events. It is straightforward that context information is useful for event detection. Therefore, the feature-based methods adopt cross-sentence information. However, they suffer from the complication of human-designed features. On the other hand, the representation-based methods learn document-level embeddings, which, however, contain much noise caused by unsupervised learning. To overcome these problems, in this paper we propose a new model based on Semi-supervised Auto-Encoder, which learns Context information to Enhance Event Detection, thus called SAE-CEED. This model first applies large-scale unlabeled texts to pre-train an auto-encoder, so that the embeddings of segments learned by the encoder contain the semantic and order information of the original text. It then uses the decoder to extract the context embeddings and fine-tunes them to enhance a bidirectional neural network model to identify event triggers and their types in sentences. Through experiments on the benchmark ACE-2005 dataset, we demonstrate the effectiveness of the proposed SAE-CEED model. In addition, we systematically conduct a series of experiments to verify the impact of different lengths of text segments in the pre-training of the auto-encoder on event detection.

CCS CONCEPTS

• **Applied computing → Document analysis**.

KEYWORDS

knowledge graph, event detection, neural network, auto-encoder

ACM Reference Format:
Yue Zhao, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Semi-supervised Auto-encoder Based Event Detection in Constructing Knowledge Graph for Social Good. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI '19), October 14–17, 2019, Thessaloniki, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3350546.3360736

WI '19, October 14–17, 2019, Thessaloniki, Greece

ACM ISBN 978-1-4503-6934-3/19/10…$15.00
https://doi.org/10.1145/3350546.3360736

1 INTRODUCTION

Knowledge graph, an important branch of artificial intelligence, has exerted widespread influence on many different areas, including vertical search, intelligent question answering, disaster management and relief [23], disease diagnosis [24], to name a few. For example, in [23] a disaster knowledge graph was developed for facilitating critical resource query and improving decision making throughout disaster management. A knowledge graph usually consists of a group of entities, as well as the inter-relations among them. In practical knowledge graphes, events are a typical type of entities. Therefore, detecting and further extracting events from texts is very important for constructing knowledge graph and has thus attracted numerous attention in recent years.

Event detection is a key sub-task in the field of event extraction and the basic step of event argument identification. This paper focuses on sentence-level event detection, which determines whether or not a sentence in a document contains a predefined event and if it does, it further identifies the corresponding event trigger words that can determine the type of the event. For instance, in the sentence "Israel retaliated with rocket *attacks* and terrorists *blew* a hole in a United States warship in Yemen.", contained in the ACE-2005 benchmark dataset[1], an event detection system should be able to detect an *ATTACK* event with the trigger words "attacks" and "blew". However, it is a challenging task, because the same event may be indicated by different trigger words and a trigger word may express different event types in different contexts.

Most of the existing event detection methods only pay attention to the sentence level information for event classification. They can mainly be categorized into two classes, namely, feature-based models and representation-based models. The former almost rely on a set of hand-designed features, such as, lexical features and syntactic features, extracted by natural language processing toolkits,

[1] https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf

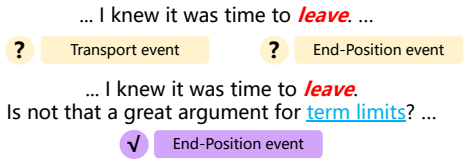

Figure 1: An example of event detection.

and then feed these features into classifiers (e.g., support vector machine [26] or maximum entropy learner [5]) to complete event classification [1, 15]. The latter employ word embeddings as the input and decode them into low-dimensional distributed representations by neural networks (e.g., convolution neural network [12], recurrent neural network [19]) to capture meaningful semantic information [2, 3, 17, 18, 20–22]. However, the same event trigger word may represent different events in different contexts due to the ambiguity of words and the flexibility of expression. It is usually difficult to distinguish the event type with only the information of a single independent sentence. For example, in Figure 1, if we only examine the first sentence, it is hard to determine whether the trigger word "leave" indicates a "Transport" event meaning that he/she wants to leave the current place, or an "End-Position" event indicating that he/she will stop working for the current organization. However, if we observe "term limits" from the context of this sentence, it will be more confident to label "leave" as the trigger word of an "End-Position" event.

Upon such an observation, there have been some studies that aim to exploit the clues beyond sentences to improve sentence-level event detection. Some remarkable methods adopt cross-document inference [11], cross-event inference [16], cross-entity inference [9], modeling textual cohesion [10] and exploiting document-level information [7]. However, they suffer from two major limitations. First, the feature-based studies usually use manually designed features, which are not only time-consuming and complicated, but also cannot guarantee the reasoning rules as complete as possible. Meanwhile, it might involve error propagation due to natural language processing; Second, the representation-based one (i.e., [7]), which employ the Distributed Memory of Paragraph Vectors model to train document embeddings and further uses it in a RNN-based event classifier, might result in that the document-level representation cannot specifically capture event-related information, as being limited by the unsupervised training process.

Therefore, in this paper we pre-train a semi-supervised auto-encoder [4] with unlabeled texts based on recurrent neural networks to learn distributed representations of contexts for event detection. The main advantages are three-fold: (1) The recurrent neural network as the encoder and the decoder can convert a sequence into a hidden state, which is inspired by the work in sequence to sequence learning [25]. It avoids long distance syntactic parsing of the sequence and thus eliminates error propagation; (2) The encoder part of a pre-trained auto-encoder can be used as a starting point of the supervised recurrent neural network, which applies weights to learn context embeddings for initialization. This semi-supervised approach can allow for easy fine-tuning. It can then improve the performance of the supervised task and significantly stabilize the training process. We believe that it is superior to other unsupervised sequence learning methods, e.g., paragraph vectors [14]; (3) The data with labeled event information is really limited due to the diversity of events and the complexity of their structures. For example, the English ACE-2005 dataset contains only 599 documents of different types. Thus we can benefit from the unsupervised pre-training of the auto-encoder, which applies additional large quantities of unlabeled data to improve the generalization ability of recurrent neural networks. Meanwhile, the learning of sequence vector restores both sentence-level and word-level information in contrast to word embedding.

In general, in this paper we propose a new model based on a Semi-supervised Auto-Encoder, which learns Context information to Enhance Event Detection, thus called SAE-CEED. This model first apply large-scale unlabeled texts to pre-train the auto-encoder, which uses a three-layer stacked GRU-based encoder to encode long segments with fixed length into embeddings and applies an another GRU-based decoder to reconstruct the original segments. In this way, the learned embeddings of segments contains their semantic and syntactic information. Then, the model uses the weights obtained from pre-training to extract the context embedding by the encoder and enhance the bidirectional GRU model, which identifies event triggers and their types in sentences, through adaptively fine-tuning the context embedding according to the specific domain. Through experiments on the benchmark dataset, ACE-2005, we evaluate the developed SAE-CEED model and demonstrate its effectiveness. In addition, we systematically conduct a series of experiments to verify the impact of different lengths of segments in pre-training of the auto-encoder on event detection.

2 THE SAE-CEED MODEL

In this paper, event detection is formalized as a multi-class classification problem. We determine each trigger candidate to a certain predefined event type. In a general way, we treat every word in a sentence as a trigger candidate except stop-words. In the benchmark ACE-2005 dataset, there are 34 event subtypes including a special "Not Applicable (NA)" type. Figure 2 presents the illustrative diagram of the proposed SAE-CEED model, which mainly contains two modules:

- The Auto-Encoder Pre-Training (AEPT) module, which trains the encoder and decoder through large-scale unlabeled texts. In this way, the pre-trained encoder can learn distributed representations of segments and be used to extract the context information in event detection.
- The Context information Enhanced Event Detection (CEED) module, which tags each trigger candidate with a certain event type based on the learned embeddings of context segments.

2.1 The AEPT module

To learn an effective context embedding of a sentence, we apply large-scale unlabeled texts to pre-train the auto-encoder model presented in the upper block of Figure 2, which mainly consists of three steps.

2.1.1 *Data preprocessing.*

We use English Wikipedia data in order to obtain large-scale unlabeled texts. First, the texts are cleaned by removing explicit non-text

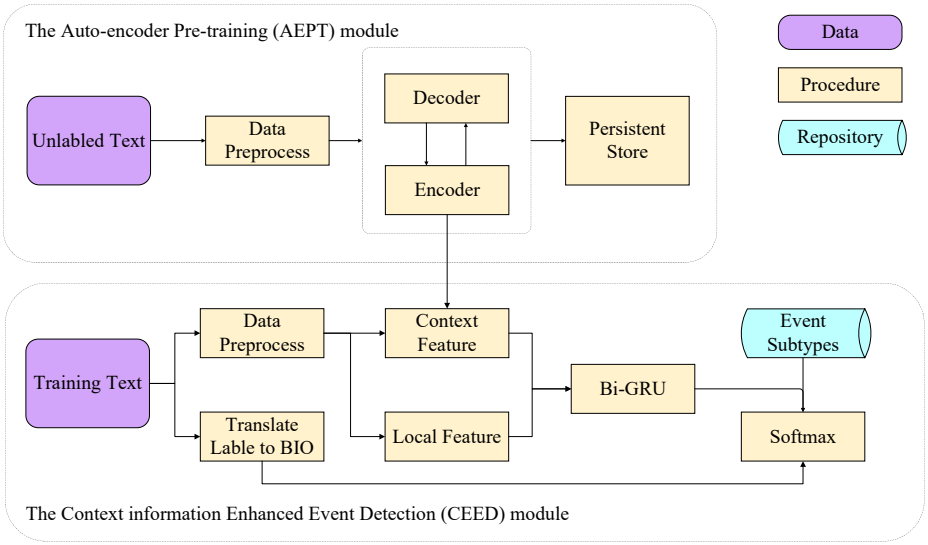

Figure 2: The illustrative diagram of the SAE-CEED model for sentence-level event detection.

fragments (e.g., web page markups) and converting words to lower case ones. Then we segment the texts according to punctuations and remove stop words. Meanwhile, special treatments such as morphological restoration and acronym reduction are performed. Finally, we construct the training set by randomly cutting texts into segments with L words avoiding the input texts too long, as the GRU based auto-encoder may suffer from the long-term dependency problem.

2.1.2 *Pre-training of the auto-encoder.*

The auto-encoder consists of an encoder and a decoder, which is shown in Figure 3. A recurrent neural network can model a continuous sequence, like textual data. In order to extract text features at different levels, we choose a three-layer stack-ed GRU-based model in the encoder, and the output of its last layer is regarded as the embeddings of the segments. We also apply a GRU-based model in the decoder, which is used to parse the segment embedding and derive the original textual sequence from it. If the decoder can correctly reconstruct the distributed representation of the segment, it can be claimed that the segment embedding learned by the encoder implies the semantic information of the original text, which can thus be used to learn the context feature of a single sentence.

(1) **Encoder** Given the training set of segments each with L words, we build the word dictionary and randomly initialize the corresponding word embedding matrix. Given a segment consisting of words $\{w_i | i = 1, 2, ..., L\}$. For each word w_i, we first obtain its embedding $\boldsymbol{w_i}$ as the input of the encoder. In the three-layer stacked GRU-based model, we learn the output of the third layer as the segment embedding $\boldsymbol{h_p^L}$:

$$\begin{aligned} \boldsymbol{h}_{\boldsymbol{p}_1}^{\boldsymbol{i}} &= \mathrm{GRU}_{encoder1}(\boldsymbol{w_i}, \boldsymbol{h}_{\boldsymbol{p}_1}^{\boldsymbol{i-1}}), i = 1, 2, ..., L, \\ \boldsymbol{h}_{\boldsymbol{p}_2}^{\boldsymbol{i}} &= \mathrm{GRU}_{encoder2}(\boldsymbol{h}_{\boldsymbol{p}_1}^{\boldsymbol{i}}, \boldsymbol{h}_{\boldsymbol{p}_2}^{\boldsymbol{i-1}}), i = 1, 2, ..., L, \\ \boldsymbol{h}_{\boldsymbol{p}}^{\boldsymbol{i}} &= \mathrm{GRU}_{encoder3}(\boldsymbol{h}_{\boldsymbol{p}_2}^{\boldsymbol{i}}, \boldsymbol{h}_{\boldsymbol{p}}^{\boldsymbol{i-1}}), i = 1, 2, ..., L, \end{aligned} \tag{1}$$

where the hidden state of the first layer is the input of the second layer, and the hidden state of the second layer is the input of the third layer.

(2) **Decoder** The decoder takes the segment embedding $\boldsymbol{h_p^L}$ as its input into the GRU and obtains the representation $\boldsymbol{q_j}$ for each w_j in the reconstructed segment. Finally, we get the predicted probability vector $\boldsymbol{z_j}$ of K dimensions through a softmax layer for the j-th position in the reconstructed segment, where the k-th element indicates the probability of classifying the j-th position to the k-th word. And the predicted $index_j^{'}$ is finally obtained according to the maximum probability in $\boldsymbol{z_j}$.

$$\begin{aligned} [\boldsymbol{q}_1, \boldsymbol{q}_2, ..., \boldsymbol{q}_L] &= \mathrm{GRU}_{decoder}(\boldsymbol{h_p^L}), \\ \boldsymbol{z_j} &= \mathrm{SoftMax}(\boldsymbol{q_j}), j = 1, 2, ..., L, \\ index_j^{'} &= \mathrm{argmax}\,(\boldsymbol{z_j}), j = 1, 2, ..., L. \end{aligned} \tag{2}$$

The loss function, $\mathbb{J}(\boldsymbol{index}, \boldsymbol{z})$, can thus be defined in terms of the cross-entropy error between the real word $index_j$ and the predicted probability $z_j^{(k)}$ as follows:

$$\mathbb{J}(\boldsymbol{index}, \boldsymbol{z}) = -\sum_{j=1}^{T}\sum_{k=1}^{K} \mathrm{I}(index_j = k) \log z_j^{(k)}, \tag{3}$$

where $I(\cdot)$ is the indicator function.

2.1.3 *Persistent store.*

We persistently store the parameters of the auto-encoder after finishing the training process, including the word embeddings and all parameters of the encoder and the decoder. The parameters of the encoder will be fine-tuned during event detection when it is used to extract the distributed representation of the context feature.

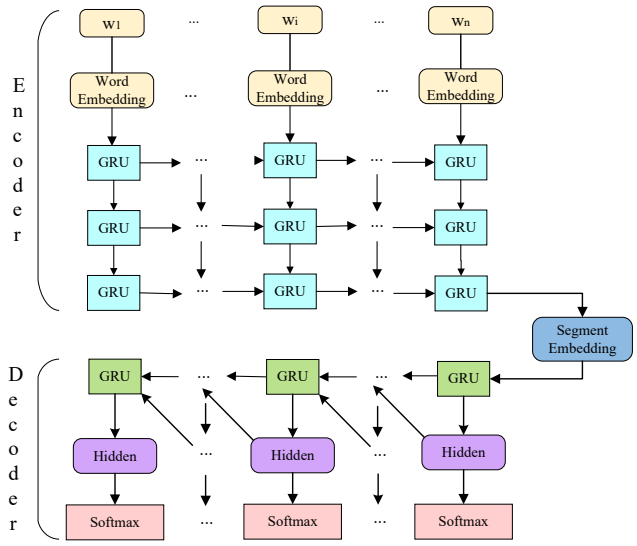

Figure 3: The framework of the auto-encoder.

2.2 The CEED module

In this module, we conduct sentence-level event detection, which is presented in the bottom block of Figure 2. Firstly, the training corpus where events are labeled is preprocessed, including word segmentation and morphological restoration. Then, we convert the labels into the "BIO" labelling mode, which can handle event triggers with multiple words appeared frequently in English. There are three categories of tags in the "BIO" labelling mode, which are "B (Begin)", indicating that the word is the start of the target phrase, "I (In)", indicating that the word is a non-starting word of the target phrase, and "O (Other)", indicating that the word is not the target phrase, respectively. In the actual annotation, if the trigger word consists of only one word, only the "B-event type" is marked; if it consists of two or more words, such as "take over", it marks "take" as "B-Event type" and "over" as "I-event type"; other non-trigger words are directly marked as "O".

The event detection model mainly consists of feature extraction and context information enhanced detection, which is shown in Figure 4. In what follows we describe the specific steps.

2.2.1 *Feature extraction.*

We extract and concatenate both the context feature and the word features as the input of the event detection model. Given a sentence $s_i (i = 1, 2, ..., D)$, we learn the distributed representation $\boldsymbol{c_i}$ of its preceding and succeeding segments through the pre-trained encoder, which is viewed as its context information.

We treat every word $\{w_{i,j}|j = 1, 2, ..., T\}$ in the sentence s_i as a candidate trigger word. For each word $w_{i,j}$, its word features consist of its word embedding $\boldsymbol{w_{i,j}}$ and its entity type embedding[2] $\boldsymbol{e_{i,j}}$ [21]. Note that, the embedding of a word is a kind of distributed representation, which expresses a word as a continuous and dense vector of fixed length. Compared with the one-hot representation, it cannot only contain the semantic information of the word, but also save storage space. The entity type embedding is complementary to the embedding of the word, where the entity refers to a specific type in the sentence.

Finally, we concatenate the context feature $\boldsymbol{c_i}$ and word features $\boldsymbol{w_{i,j}}$ and $\boldsymbol{e_{i,j}}$ as the whole feature $\boldsymbol{u_{i,j}}$ of a candidate trigger word and input it into the detection model. At the same time during the training of the event detection model, the parameters in the encoder, the word embedding and the entity type embedding are also updated and fine-tuned according to the supervision information.

2.2.2 *Context information enhanced detection.*

We consider the extraction of event trigger words as a multi-label classification task and employ a Bi-GRU [6] model to learn . In general, the semantic information of a word is not only related to its succeeding words, but also related to its preceding words considering the linguistic flexibility. Therefore, we employ the bi-directional GRU network to encode the sentence, which enters a sequence of length T and get the forward hidden state $\left[\boldsymbol{h_1^f}, \boldsymbol{h_2^f}, ..., \boldsymbol{h_T^f}\right]$ and the backward hidden state $\left[\boldsymbol{h_1^b}, \boldsymbol{h_2^b}, ..., \boldsymbol{h_T^b}\right]$, respectively. Given a sentence $s_i (i = 1, 2, ..., D)$ consisting of words $\{w_{i,j}|j = 1, 2, ..., T\}$, we input the feature $\boldsymbol{u_{i,j}}$ to the Bi-GRU network and obtain the hidden state $\boldsymbol{r_{i,j}}$:

$$\boldsymbol{r_{i,j}} = \left[\overrightarrow{\mathrm{GRU}_{event}}(\boldsymbol{u_{i,j}}), \overleftarrow{\mathrm{GRU}_{event}}(\boldsymbol{u_{i,j}})\right], \quad i = 1, 2, ..., D, j = 1, 2, ..., |s_i|. \tag{4}$$

Finally, we get the probability vector $\boldsymbol{o_{i,j}}$ of K dimensions through a softmax layer for $w_{i,j}$. The k-th element of $\boldsymbol{o_{i,j}}$, i.e., $o_{i,j}^{(k)}$ indicates the probability of classifying $w_{i,j}$ to the k-th event type. And the predicted event type $\boldsymbol{e_{i,j}}$ is finally obtained according to the maximum probability in $\boldsymbol{o_{i,j}}$.

$$\begin{aligned} \boldsymbol{o_{i,j}} &= \mathrm{SoftMax}(\boldsymbol{r_{i,j}}), i = 1, 2, ..., D, j = 1, 2, ..., |s_i|, \\ \boldsymbol{e_{i,j}} &= \mathrm{argmax}(\boldsymbol{o_{i,j}}), i = 1, 2, ..., D, j = 1, 2, ..., |s_i|. \end{aligned} \tag{5}$$

The loss function, $J(\boldsymbol{y}, \boldsymbol{o})$, can thus be defined in terms of the cross-entropy error between the real event type $y_{i,j}$ and the predicted probability $o_{i,j}^{(k)}$ as follows:

$$J(\boldsymbol{y}, \boldsymbol{o}) = -\sum_{i=1}^{D}\sum_{j=1}^{T}\sum_{k=1}^{K} \mathrm{I}(y_{i,j} = k) \log o_{i,j}^{(k)}, \tag{6}$$

where $I(\cdot)$ is the indicator function.

3 EXPERIMENTS

In this section, we validate the proposed SAE-CEED model through experimental comparison with existing state-of-the-art methods on the benchmark ACE-2005 dataset and examine the pre-training of the auto-encoder on the English Wikipedia dataset. The related settings and experiment results are described respectively in what follows.

3.1 Context Information Enhanced Detection

[2]The words in the ACE-2005 dataset are annotated with their entity types (annotated as "NA" if they are not an entity).

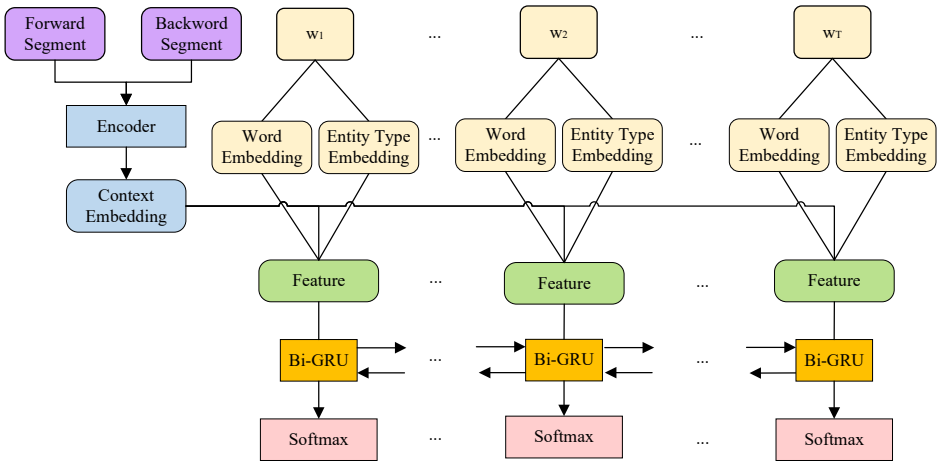

Figure 4: Event detection enhanced with context information in the SAE-CEED model.

Table 1: The statistics on the documents of the ACE-2005 English dataset.

Words		Files
	NORM	NORM
Newsgroups	48399	106
Broadcast News	55967	226
Broadcast Conversations	40415	60
Weblog	37897	119
Usenet	37366	49
Conversation Telephone Speech	39845	39
Total	259889	599

3.1.1 Datasets and settings.

We use the ACE-2005 English dataset to validate the effectiveness of the proposed model on event detection. The dataset has 599 documents, as shown in Table 1, and the data sources contain Newsgroups, Broadcast News, Broadcast Conversations, Weblog, Usenet, and Conversation Telephone Speech, respectively, in different folders. There are several annotation statuses including 1P, DUAL, ADJ, NORM and we adopt NORM, where the texts have been calibrated and normalized.

This dataset has 33 event subtypes and we regard the event detection as a multi-label classification problem of 34 class-es, added None (non-event type) as a special class in this paper.

In the experiments, the validation set has 30 documents randomly extracted from different genres, the test set has 40 documents extracted from Broadcast News and the training set contains the remained 529 documents. All the data preprocessing and evaluation criteria follow those in [8].

We tune the hyper-parameters on the validation set. The dimension of the hidden layers corresponding to $\overrightarrow{\text{GRU}_{event}}$ and $\overleftarrow{\text{GRU}_{event}}$ in the event detection model are set to 300. Thus the output size of the hidden state $\boldsymbol{r_{i,t}}$ is $300 \times 2 = 600$. The dimension of the word embedding is also 300, and the dimension of the entity type embedding is 50. In addition, we utilize the pre-trained word embedding in the auto-encoder for initialization. For entity types, their embeddings are randomly initialized.

In the training process, we apply the Stochastic Gradient Descent (SGD) over shuffled mini-batched and use dropout [13] for regularization, where the batch size and the dropout rate is set to 80 and 0.5, respectively.

3.1.2 Baselines.

In the experiment, we take four existing methods as our baselines.

- Cross-event (2010) [16]: This method proposes an artificially constructed document-level feature to improve the performance of event extraction, which learns relations among event types from the training corpus and further helps predict the occurrence of events.
- Sentence-level (2011) [9]: This is a feature-based model for event detection, which regards entity type consistency as a key feature to predict event mentions and uses its reasoning mechanism to improve traditional event detection at sentence level.
- JRNN (2016) [20]: This is a representation-based model, which proposes a joint framework based on bidirectional recurrent neural networks and exploits the inter-dependency between event triggers and argument roles via discrete structures.
- DLRNN (2017) [7]: This is also a representation-based model, which automatically extracts cross-sentence clues via learning document embeddings with the unsupervised PV-DM model to improve sentence-level event detection.

In this paper, precision, recall and F1-measure are used to evaluate the performance of the model on the event detection task, which indicate the correctness, completeness and comprehensive performance of the model respectively. Therefore, we mainly focus on F1- measure. Precision, recall and F1-measure are specifically defined as follows:

Table 2: Performance comparison between the SAE-CEED model and other baselines on the ACE-2005 English dataset.

Model	Precision	Recall	F1
Sentence-level(2011)	67.6	53.5	59.7
JRNN(2016)	66.0	73.0	69.3
Cross-event(2010)	68.7	68.9	68.8
DLRNN (2017)	77.2	64.9	70.5
SAE-CEED	68.1	74.2	71.0

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive},$$
$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}, \quad (7)$$
$$F1 - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall},$$

where $TrusePositive$ is the number of sentences with labeled events, which are correctly extracted and identified, $FalsePositive$ is the number of sentences without labeled events, but they are extracted, and $FalseNegative$ is the number of sentences with labeled events, which are not correctly extracted.

3.1.3 Performance comparison on event detection.

Table 2 presents performance comparison between different event detection methods on ACE-2005. We can see that our proposed model outperforms the existing methods in terms of both recall and F1-measure, while its precision is comparable to that of others. The better performance of SAE-CEED can be explained by the following reasons: (1) Compared with the methods only focusing on sentence-level information, including feature-based methods, *Sentence-level* [9], and the representation-based method, *JRNN* [20], our method employs context information to enhance its capability of identifying trigger words. Therefore, all the three evaluation metrics have better performance; (2) Compared with the methods exploiting information beyond sentence level, such as the feature-based method, *Cross-event* [16], and the representation-based method, *DLRNN* [7], out method can not only automatically capture rich context information via an end-to-end Bi-GRU based model without manually designed rules, but also learn event detection oriented embeddings of the contexts through the fine-tuning process in the training of event detection. On the other hand, the experimental results show that the pre-training based on unlabeled English Wikipedia data can effectively improve the performance of our model in the news field, which reflects the better robustness.

3.2 Auto-Encoder Pre-Training

3.2.1 Datasets and settings.

English Wikipedia is the English version of this free online encyclopedia, which was founded on January 15, 2001 and is its first version. It is the most widely published version of Wikipedia in all languages. As of March 20, 2019, there were 5,826,555 articles in English Wikipedia, and the detailed statistics is presented in Table 3. English Wikipedia has a mechanism for assessing the quality and importance of the article, which guarantees the reliability of the corpus while satisfying the various criteria. Therefore, we use articles in English Wikipedia to pre-train the auto-encoder.

Table 3: The statistics of English Wikipedia.

Number of user accounts	Number of articles	Number of files	Number of administrators
35,939,162	5,826,555	882,475	1,183

We build the training set and the validation set by randomly sampling segments of length L, where L is a hyper-parameter. As aforesaid, the recurrent neural network may suffer from the long-term dependency problem if L is too large. But, the segment embedding may lack of semantic information, if L is set too short. Therefore, we have calculated the document length distribution of the ACE-2005 English dataset, as shown in Figure 5, which shows that about 70% of the documents is less than 600 words. We set the segment length to 70, 100, 150, and 300 for training, and use the trained encoder to extract the preceding and succeeding context feature of length 140, 200, 300 and 600, respectively. The training set contains 337,944 segments and the validation set contains 36,263 segments.

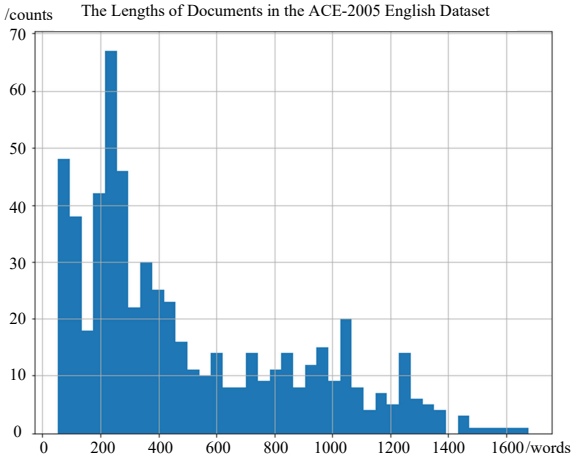

Figure 5: The statistics on document length of the ACE-2005 English dataset.

We tune the hyper-parameters on the validation set. The dimensions of the hidden layers corresponding to $\text{GRU}_{encoder1}$, $\text{GRU}_{encoder2}$, $\text{GRU}_{encoder3}$ and $\text{GRU}_{decoder}$ are set to 600, 600, 300 and 300, respectively. The word dictionary contains 30,000 words with the highest frequency.

There are two main evaluation indicators for the auto-encoder. The first one is the loss function $\mathbb{J}(\boldsymbol{index}, \boldsymbol{z})$ during training, which measures the difference between the predicted vector and the actual vector. The magnitude of loss reflects the convergence of the model. The second one is the accuracy in reconstructing segments. A word w_j is considered to be correctly restored when $index_j^{'}$ predicted by the decoder is the same as the original word index. Here, the accuracy is defined as the ratio of the number of correctly reconstructed words to the total number of words in the training set.

3.2.2 Impacts of segment length on the auto-encoder.

Table 4: The performance of auto-encoder based pre-training with different length of segments on English Wikipedia.

Length of Segment	Loss	Accuracy
70	3.64	35.91%
100	3.59	36.47%
150	3.43	37.82%
300	3.51	36.22%

Table 4 shows the value of loss function and classification accuracy of the auto-encoder of different segment lengths on the validation set. We can see that as the length of segments increases from 70 to 150, both measurements are gradually improved; However, when the length of the segment further increases to 300, the performance of the model is degraded. This is because that the GRU based model has a limited ability to reconstructing too long texts. It is thus not suggested to choose a segment that is very long for training.

In order to verify the validity of the encoder, we calculate the cosine similarity between the last hidden layer state, which is considered as the reconstructed word embedding, as shown in Table 5. It can be found that "cat" and "dog", "apple" and "banana", "king" and "queen", "beijing" and "China" have the highest similarity, respectively, which reflects that the reconstructed word embeddings basically contain the correct semantic information. In other words, the distributed representation of context learned by the encoder effectively models the semantic and order information of the segment.

Table 6 presents the performance of the proposed model using context representations pre-trained with segments of different lengths. We can see that the performance of event detection is significantly and positively correlated with that of the pre-trained auto-encoder, and particularly, the best F1-measure is achieved when the segment length is 150.

4 CONCLUSIONS

As one of the hottest research branches of artificial intelligence, knowledge graph has attracted tremendous attention from diversified areas, including vertical search, intelligent question answering, disaster management and relief, disease diagnosis etc. Particularly, event-centric knowledge graphs have gained successful application in such fields as finance and disaster management. In this paper, we investigated the problem of event detection from texts, which is fundamental for constructing event-centric knowledge graphs. Specifically, we proposed a semi-supervised auto-encoder based model, called SAE-CEED, to learn context information for enhancing event detection. Moreover, we used large-scale unlabeled text data to pre-train the auto-encoder for keeping both the semantic and order information in segment embeddings, and fine-tuned the auto-encoder to learn the context embedding in the training process of event detection, which can learn the event-oriented distributed representation. In order to prove the effectiveness of the proposed method, we systematically conducted a series of experiments on the ACE-2005 dataset. Experimental results show that the proposed method is better than both the sentence-level models and the models exploiting information beyond sentence level in terms of recall and F1-measure. Meanwhile, we verified the impact of segments of different lengths in pre-training the auto-encoder on event detection.

In this paper, we search the better length of segments by experiments on the validation set. In the future, we may improve the SAE-CEED model to automatically determine the optimal length of segments.

ACKNOWLEDGMENTS

This work is supported by National Key Research and Development Program of China under grant 2016YFB1000902, National Natural Science Foundation of China under grants 61772501, 61572473, 61572469, and 91646120, and the GF Innovative Research Program.

REFERENCES

[1] David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. 1–8.
[2] Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 409–419.
[3] 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)*, Vol. 1. 167–176.
[4] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*. 3079–3087.
[5] Hal Daumé III. [n.d.]. Notes on CG and LM-BFGS optimization of logistic regression. ([n. d.]).
[6] Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. 2016. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549* (2016).
[7] Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Vol. 1. 352–361.
[8] Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2018. Event nugget detection with forward-backward recurrent neural networks. *arXiv preprint arXiv:1802.05672* (2018).
[9] Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. 1127–1136.
[10] Ruihong Huang and Ellen Riloff. 2012. Modeling Textual Cohesion for Event Extraction. In *AAAI*. 1664–1670.
[11] Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. *Proceedings of ACL-08: HLT* (2008), 254–262.
[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. 1097–1105.
[14] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.
[15] 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)*, Vol. 1. 73–82.
[16] Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. 789–797.
[17] Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016. Leveraging framenet to improve automatic event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 2134–2143.
[18] Shulin Liu, Yubo Chen, Kang Liu, Jun Zhao, et al. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. (2017).

Table 5: Cosine similarity between word embeddings reconstructed by the decoder.

	Cat	Dog	Apple	Banana	King	Queen	Beijing	China
Cat	1	**0.914**	0.836	**0.875**	0.704	0.737	0.821	0.749
Dog	**0.914**	1	0.796	0.844	0.672	0.708	0.790	0.716
Apple	0.836	0.796	1	0.862	0.696	0.725	0.803	0.740
Banana	0.875	0.844	**0.862**	1	0.693	0.738	0.835	0.768
King	0.704	0.672	0.696	0.693	1	**0.865**	0.670	0.617
Queen	0.737	0.708	0.725	0.738	**0.865**	1	0.695	0.646
Beijing	0.821	0.790	0.803	0.835	0.670	0.695	1	**0.842**
China	0.749	0.716	0.740	0.768	0.617	0.646	**0.842**	1

Table 6: Event detection performance of SAE-CEED with different length of segments on the ACE-2005 English dataset.

Length of Segment	P	R	F1
70	66.7	72.9	69.7
100	67.5	73.6	70.4
150	68.1	74.2	71.0
300	67.7	73.8	70.6

[19] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association.*

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

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

[22] Thien Huu Nguyen and Ralph Grishman. 2016. Modeling skip-grams for event detection with convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 886–891.

[23] Hemant Purohit, Rajaraman Kanagasabai, and Nikhil Deshpande. 2019. Towards Next Generation Knowledge Graphs for Disaster Management. In *Proceedings of the 2019 IEEE 13th International Conference on Semantic Computing (ICSC).*

[24] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. 2017. Learning a Health Knowledge Graph from Electronic Medical Records. *Scientific Reports* 7, 1 (2017).

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[26] Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters* 9, 3 (1999), 293–300.