Learning to Select Important Context Words for Event Detection

Nghia Trung Ngo[1]⋆, Tuan Ngo Nguyen[2]⋆, and Thien Huu Nguyen[2,3]

[1] Hanoi University of Science and Technology, Hanoi, Vietnam
[2] Department of Computer and Information Science, University of Oregon, USA
[3] VinAI Research, Hanoi, Vietnam
nghia.nt152654@sis.hust.edu.vn, {tnguyen,thien}@cs.uoregon.edu

Abstract. It is important to locate important context words in the sentences and model them appropriately to perform event detection (ED) effectively. This has been mainly achieved by some fixed word selection strategy in the previous studies for ED. In this work, we propose a novel method that learns to select relevant context words for ED based on the Gumbel-Softmax trick. The extensive experiments demonstrate the effectiveness of the proposed method, leading to the state-of-the-art performance for ED over different benchmark datasets and settings.

Keywords: Event Detection · Context Selection · Deep Learning · Information Extraction.

1 Introduction

One of the important tasks of Information Extraction (IE) is Event Detection (ED) in Natural Language Processing (NLP) that seeks to identify event triggers in sentences and classify them into some predefined types of interest. Event triggers amount to the most important words (usually single verbs or nominalizations) in the sentences that are responsible for evoking the events. For instance, according to the annotation guideline of the Automatic Context Extraction evaluation (ACE) 2005[4], the word "*fired*" in the following sentence is the trigger word for an event of type *Attack*:

S1: *The police* **fired** *tear gas and water cannons in street battles with activists.*

Most of the recent studies on ED has followed the deep learning approach where the representations of the triggers and their sentences are induced automatically from data to perform classification [25, 2, 9]. It has been shown that this approach is able to mitigate the feature engineering effort and avoid the reliance on other NLP toolkits for feature generation [12], leading to ED models with better performance and robustness [9].

The typical components in the recent deep learning models for ED involve Convolutional Neural Networks [2, 21], Long Short-Term Memory Networks (LSTM)

⋆ Nghia Trung Ngo and Tuan Ngo Nguyen contribute equally to this paper.

[4] https://www.ldc.upenn.edu/collaborations/past-projects/ace

[22], attention mechanisms [16] and their combinations. Among many variations of the model architectures for ED, one key insight is some context words in the sentences are very important to correctly classify the trigger words and modeling those important context words appropriately is crucial to the success of the ED models. Consider the two following sentences as an example:

S2: *She needs to go home to meet a friend at 5, so she has to* **leave** *the company early.*

S3: *She decided to* **leave** *the company for a better job.*

Both sentences contain the word "*leave*" that triggers different event types in this case (i.e., an *Transport* event in **S2** and an *End-Position* event in **S3**). Due to this ambiguity, the word "*leave*" itself would not be sufficient to determine its event types in these sentences, necessitating the modeling of the relevant context words. In particular, the necessary context words for ED in **S2** would be "*company*", "*to*", "*go*" and "*home*" while those words in **S3** involve "*company*", "*for*", "*better*, and "*job*".

In order to model the important context words for ED, we first need to locate such words in the sentences for the event trigger candidates. The prior work on ED has mainly addressed this problem by devising some heuristics/strategy for relevant word selection based on the intuition and examination of a sample of event mentions. The two typical word selection strategies for ED have involved: (i) the entity mention strategy: selecting the words that correspond to the entity mentions and the surrounding words in the sentences [16] (e.g., the entity mention words "*She*", "*home*", "*friend*", "*she*", and "*company*" in **S2**), and (ii) the dependency strategy: selecting the words connected to the trigger words in the dependency parse trees of the sentences [27, 31, 30].

A common issue of these previous strategies for relevant word selection for ED is that the important context words are predetermined without considering the specific context words in each sentence. Such predetermined selection rules are brittle as they cannot identify the important context words for ED in all the possible sentences for event mentions. For instance, the entity mention rule would not be able to select the import words "*for*", "*better*" and "*job* in **S2** as these words are not parts of any entity mentions (assuming the entity mentions defined in ACE 2005). Similarly, the dependency rule would be incapable of directly taking the important words "*go*" and "*home*" in **S3** as they are far apart from "*leave*" in the dependency tree of **S3**. It would be desirable if the context word selection strategies can be customized for each input sentence and its event trigger candidate so the most effective context words for each trigger candidate can be revealed and modeled.

Consequently, instead of using the fixed rules, in this work, we propose a novel method for ED that *learns to select important context words* to perform event prediction for trigger word candidates in the sentences based on deep learning. For an input sentence, we propose to model the word selection process as a sequence of word selection steps that starts with the trigger word candidate of interest. At each step, the next important word for the event prediction would be chosen from the sentence, conditioned on the words that have been previously selected. Such selected words would be composed in the end to produce the feature representation for our ED problem. In particular, along the word selection sequence, a representation vector is maintained to encode the information for the words chosen up to the current step. The representation vector is first set to the representation of the trigger word candidate obtained from an LSTM model. At each step, the current representation vector is used as the query to compute a relevance/importance score for each of the non-selected words and the word with the highest relevance score would be chosen to incorporate into the current representation vector via a composition function. After some selection steps, the final representation vector would be utilized as the features to make prediction for ED. In this way, the model is trained to automatically identify important words in a given input sentence for ED, potentially leading to better relevant word choices to improve ED performance.

A unique property of the proposed model is that we only encapsulate the learned important/relevant words in the overall representation vector for prediction, excluding the irrelevant words (i.e., hard attention). This is in contrast to all the previous models for ED where all the words in the sentences are included in the prediction representation via the weighting schema [16, 9] (i.e., soft attention), potentially preserving the noise in the irrelevant words and degrading the ED performance.

In the proposed model, we need to deal with the discrete variables to indicate which word has the highest relevance score in each word selection step. These discrete variables are not differential and the usual back-propagation technique cannot be applied directly to train the proposed model. We solve this problem by employing the Gumbel-Softmax method [10] that takes different paths for the forward and backward phases in the computational graphs in the training process of the proposed model. We conduct extensive experiments on the benchmark datasets for ED (i.e., the ACE 2005 and TAC 2015 datasets) with different settings (i.e., the general setting and the cross-domain setting). The experiments demonstrate the effectiveness of the proposed model by advancing the state-of-the-art performance on those standard datasets over different settings.

2 Model

This work formalizes ED as a multi-class word classification problem where we need to predict the event type for a given word in a sentence. The given word is referred as the event trigger candidate or the anchor that constitutes an event mention when it is associated with the corresponding sentence. The event mentions can be seen as examples in our multi-class classification problem. Besides the multiple event types of interest (i.e., the ones in the ACE 2005 dataset), we have a special types, called *Other*, to indicate that a trigger candidate is not expressing any event type of interest.

Consider an event mention with $W = w_1, w_2, \ldots, w_n$ as the input sentence and $w_a (1 \leq a \leq n)$ as the anchor word of interest (w_i is the i-th word/token in W). The following sections will describe the components in the proposed model to predict the event type for $w_a \in W$, including Sentence Encoding and Relevant Word Selection.

2.1 Sentence Encoding

In the sentence encoding component, we first transform the discrete tokens in W into real-valued vectors. Following the previous work on ED [2, 27], a word $w_i \in W$ in this work would be encoded as the concatenation vectors x_i of the three elements emb_i, pos_i and ner_i: $x_i = [emb_i, pos_i, ner_i]$, where:

- emb_i: is the pre-trained word embeddings of x_i (e.g., `word2vec` [18]).
- pos_i: is a real-valued vector to represent the relative distance from the current word w_i to the anchor word w_a in W (i.e., $i - a$). These vectors are randomly initialized and updated during the training process.
- ner_i is the embedding vector of the entity type tag of $w_i \in W$. Specifically, the entity mentions in W would be used to obtain an entity type tag for each word in W via the BIO tagging schema. ner_i is produced by looking up the tag of w_i in the entity type embedding table that involves real-valued vectors to encode the entity type tags.

After the word-to-vector transformations $w_i \rightarrow x_i$, we obtain a sequence of vector $X = x_1, x_2, \ldots, x_n$ for the input sentence W that is specific to the anchor word w_a for ED. As each vector $x_i \in X$ only encapsulates the information about the word w_i itself, we feed the whole vector sequence X into a bidirectional LSTM network to further abstract W and incorporate the sentence context information into the representation vector for each word [22]. The output of this bidirectional LSTM network is another hidden vector sequence $h_1, h_2, \ldots, h_n$ that would be used in the next component to select and compose the important context words in W for ED.

2.2 Relevant Word Selection

As described in the introduction, our principle to design the ED model in this work is to follow an iterative procedure to select important words in the sentences to facilitate the event type prediction for the anchor words. In particular, for the input sentence W, we perform a sequence of word selections where the most relevant word is selected at each step, conditioning on the chosen words in the previous steps. At the selection step i, denote v^i as the overall representation vector to accumulate the information about the selected words up to the current step. Also, let $G_i = w_1^i, w_2^i, \ldots, w_{N_i}^i$ be the list of words that have not been selected so far in the input sentence W (i.e., the non-selected words) (N_i is the number of words in G_i: $N_i = |G_i|$). Here, we assume that the words in G_i preserve their orders in the original sentence W (i.e., w_s^i would appear before w_t^i in both G_i and W if $s < t$). Finally, for convenience, let $u_1^i, u_2^i, \ldots, u_{N_i}^i$ be the representation vectors for the words in $G_i = w_1^i, w_2^i, \ldots, w_{N_i}^i$ at step i respectively. In this work, $u_1^i, u_2^i, \ldots, u_{N_i}^i$ are always set to the LSTM hidden vectors $h_1, h_2, \ldots, h_n$ of the corresponding words in G_i from the sentence encoding.

At the initial selection step $i = 1$, the representation vector v^1 is set to the hidden vector of the anchor word obtained from the bidirectional LSTM network (i.e., $v^1 = h_a$). This assignment emphasizes the intuition that the anchor word w_a is the most important word in the sentence for ED and it should be selected first to guide the relevant word selections in the next steps. The non-selected words at the initial step thus belong to the set $G_1 = [w_1, w_2, \ldots, w_n] \setminus w_a = w^1_1, w^1_2, \ldots, w^1_{N_1}$ with $N_1 = n - 1$ elements. The word representations $u^1_1, u^1_2, \ldots, u^1_{N_1}$ are then the LSTM hidden vectors for the words in G_1.

Relevance Score Computation As we only select one word at each step, at step $i > 1$, there should be $n - i$ words in W that have not been selected in the previous $i - 1$ steps (i.e., $N_i = |G_i| = n - i$). In order to choose the next word for step $i+1$, we estimate a relevance score for each of the non-selected words in G_i from which a relevant word would be nominated. In particular, the relevance score $r(w^i_j)$ for the word $w^i_j \in G_i$ is obtained via the dot product:

$$r(w^i_j) = q^T.cm(u^i_j, v^i, c^i) \qquad (1)$$

where q is a query vector to be learned during training and cm is a function to incorporate the information in u^i_j into v^i to form a new representation vector. Essentially, q helps to quantify how well u^i_j can be integrated into the overall representation v^i, thus determining the relevance of w^i_j with respect to the current knowledge about the sentence for ED. c^i is the cell state of the composition function at the step i (c^1 is set to the zero vector). The $cm(u^i_j, v^i, c^i)$ function in this work is motivated by the LSTM units, taking the representation vector u^i_j for the word w^i_j, the overall representation vector v^i, and the cell state c^i as the inputs to produce the composed vector as follows:

$$\begin{aligned}
\begin{bmatrix} f \\ i \\ o \\ g \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\mathbf{W}_{com} \begin{bmatrix} u^i_j \\ v^i \end{bmatrix} + \mathbf{b}_{com} \right) \\
m^{i+1}_j &= f \odot c^i + i \odot g \\
cm(u^i_j, v^i, c^i) &= o \odot tanh(m^{i+1}_j)
\end{aligned} \qquad (2)$$

where σ is a non-linear function.

Selecting the Relevant Word Given the relevance score $r(w^i_j)$ for each word $w^i_j \in G_i$, an obvious way to select a relevant word for step $i + 1$ is to take the word $w^i_{d_i}$ with the highest score, i.e., $d_i = \text{argmax}_j r(w^i_j)$. However, this obvious approach would cause a problem to train the whole model as the variable d_i to specify the index of the word chosen at step i is discrete and non-differential, blocking the back-propagation via d_i during training. In order to avoid this problem, we seek to approximate the argmax operation by using the relevance scores of the words in G_i to form a categorical distribution over G_i from which a word from G_i is sampled and chosen as the next relevant word. Two requirements arise when we apply this approximation approach: (i) an efficient method to sample from the categorical distribution over G_i, and (ii) a mechanism to allow the model to be trained with the discrete sample from G_i. In this work, we apply the Straight-Through Gumbel-Softmax (STGS) estimator proposed in [10, 17] to satisfy these requirements. In particular, in STGS, the relevance scores $r(w_j^i)$ $(j = 1 \dots N_i)$ would be transformed into the Gumbel-Softmax distribution $(y_1, y_2, \dots, y_{N_i})$ over G_i via the following softmax function (y_j represents the probability that the word $w_j^i \in G_i$ is selected): $y_j = \frac{\exp((r(w_j^i)+g_j)/\tau)}{\sum_{t=1}^{N_i} \exp((r(w_t^i)+g_t)/\tau)}$.

In this formula, $g_1, g_2, \dots, g_{N_i}$ are the i.i.d samples drawn from Gumbel(0,1) distribution [8]: $g_j = -\log(-\log(s_j)),\ s_j \sim \text{Uniform}(0, 1)$ and τ is the temperature parameter. As τ reaches zero, the distribution $(y_1, y_2, \dots, y_{N_i})$ becomes very skew toward the index with the highest score in $(r(w_1^i), r(w_2^i), \dots, r(w_{N_i}^i))$, causing the sample drawn from the distribution to be more likely the word with the highest relevance score. The first benefit of STGS with the Gumbel noise g_j is that it allows efficient sampling from the categorical distribution $(y_1, y_2, \dots, y_{N_i})$. Specifically, in order to sample an index d_i for the words in G_i from this distribution, ones only need to perform the argmax function: $d_i = \text{argmax}_j y_j$ [8, 17]. Once d_i is sampled from the distribution, we would select $w_{d_i}^i$ as the next relevant word, and consequently compute the overall representation vector and the cell state vector for the next word selection step $i+1$ via: $v^{i+1} = cm(u_{d_i}^i, v^i, c^i),\ c^{i+1} = m_{d_i}^{i+1}$.

Gumbel-Softmax Training The second benefit of STGS is that the model can use the sampled value of d_i directly by allowing the training process to take different paths in the forward and backward computation. In particular, the forward phase would compute the values of v^{i+1} and c^{i+1} via the sampled value of d_i. In contrast, the backward phase would treat v^{i+1} as the weighted sum of $cm(u_j^i, v^i, c^i)$ $(j = 1, \dots, N_i)$ with $(y_1, y_2, \dots, y_{N_i})$ as the weights and directly use m_j^{i+1} as the cell states for each word $w_j^i \in G_i$. In this way, the back-propagation algorithm can still be used to update the model parameters in the backward phase. Although the STGS estimator is biased, it has been shown to work well for different problems in practice [5, 4, 7, 33].

Finally, after k steps (a hyper-parameter) of word selection, the overall vector representation v^k would be used as the features and fed into a feed-forward neural network with a softmax layer in the end to perform event type prediction for the input sentence W with the anchor word w_a. k is a hyper-parameter of the model. We train the proposed model by optimizing the the negative log-likelihood function of the training data with shuffled mini-batches and the Adam optimizer.

3 Experiments

3.1 Experiment Settings

Datasets

Following the prior work on ED [2, 26, 9], we evaluate the models in this paper using two benchmark datasets, i.e., the ACE 2005 and the TAC KBP 2015 datasets [19]. For the ACE 2005 dataset, there are 33 event subtypes, leading to a 34-class classification problem for this dataset (adding a special class *Other* for non-trigger words). We follow the same setting as the previous studies on this dataset to ensure comparable comparison. In particular, we use the same data split with 40 newswire articles for the test set, 30 other documents for the development set and 529 remaining documents for the training set. We also use the entity mentions provided by the annotation in this dataset to generate the BIO tags for the entity type embedding table [26, 9].

The TAC KBP 2015 dataset involves 38 event subtypes that introduces a 39-class classification problem with the *Other* type. Similar to the other work on this dataset, we employ the official data split provided by the 2015 Event Nugget Detection evaluation [19], including 360 documents for the training dataset and 202 documents for the test dataset [27]. We do not use the entity type embedding table in this case as the entity mentions are not provided for this dataset.

Parameters and Resources

We tune the model parameters on the ACE 2005 development dataset. We find the following values for the model parameters: 600 hidden units for the forward and backward LSTM networks in the sentence encoding component, $\tau = 1.0$ for the temperature in the Gumbel-Softmax distribution, 8.10^{-5} for the learning rate, 50 dimensions for the vectors for the relative distances and the entity type embedding table, 300 dimensions for the other intermediate vectors, 64 for the batch size, and $k = 4$ for the number of word selection steps. Finally, similar to the previous work [9, 3], we utilize the pre-trained word embeddings provided by the `word2vec` model with 300 dimensions [18].

3.2 Evaluation on ACE 2005

This section compares the proposed model (called *LearnToSelectED*) with the state-of-the-art models on the ACE 2005 dataset. In particular, the following state-of-the-art models are selected for comparison: 1) **CNN**: the CNN model [24, 25], 2) **DM-CNN**: the dynamic multi-pooling CNN [2], 3) **DM-CNN+**: the DM-CNN model augmented with additional data [1], 4) **JRNN**: the bidirectional recurrent neural network (RNN) model [22], 5) **NCNN**: the non-consecutive CNN [26], 6) **FrameNet**: the model with additional data from FrameNet [15], 7) **CNN-LSTM**: the ensemble model of CNN and LSTM [6], 8) **SupervisedAtt**: the supervised attention model [16], 9) **DepTensor**: the dependency-bridge RNN [31], 10) **MultiAtt**: the multilingual attention model [14], 11) **GCNN-ENT**: the graph-based CNN model with entity mention-based pooling [27], 12) **SELF-GAN**: the generative adversarial network for ED [9], 13) **HBT**: the multi-level attention model [3], and 14) **DEEB-RNN**: the document embedding enhanced RNN model [35]. Figure 1 presents the performance.

Note that in order to demonstrate the benefit of hard attention over soft attention, we also implement the Transformer model [32] for our ED problem

Model	P	R	F
CNN [25]	71.8	66.4	69.0
DM-CNN [2]	75.6	63.6	69.1
DM-CNN+ [1]‡	75.7	66.0	70.5
JRNN [22]	66.0	73.0	69.3
CNN-LSTM [6]	84.6	64.9	73.4
FrameNet [15]‡	77.6	65.2	70.7
DepTensor [31]	-	-	69.6
SELF-GAN [9]	71.3	74.7	73.0
NCNN [26]	-	-	71.3
GCNN-ENT [27]	77.9	68.8	73.1
SupervisedAtt [16]‡	76.8	67.5	71.9
MultiAtt [14]‡	78.9	66.9	72.4
HBT [3]†	77.9	69.1	73.3
DEEB-RNN [35]	72.3	75.8	74.0
Transformer [32]	73.4	69.5	71.4
LearnToSelectED	75.4	75.0	**75.2**

Table 1. Performance on the ACE 2005 dataset. †indicates the use of information beyond the sentence level. ‡specifies the use of additional training data. *LearnToSelectED* is significantly better than the other models with $p < 0.05$.

using the same resources for *LearnToSelectED*. The most important observation from the table is that the proposed model *LearnToSelectED* significantly outperforms all the compared methods in this work. In particular, although *LearnToSelect* is only trained on the ACE 2005 training dataset, is also better than the other methods that utilize additional data for training (i.e., **DM-CNN+**, **FrameNet** and **MultiAtt**) (an improvement of 2.8% over the best data-augmented method **MultiAtt** [14]). Comparing *LearnToSelectED* with the other methods that explicitly select the relevant context words for ED via fixed strategies (e.g., **NCNN**, **SupervisedAtt**, **GCNN-ENT**), we see that *LearnToSelectED* has significantly better performance (an improvement of 2.1% absolute F1 score over the best model **GCNN-ENT** of this type). This testifies to the effectiveness of the automatic identification of important context words for ED in this work. Finally, *LearnToSelectED* is significantly better than the soft attention models (i.e., **SupervisedAtt**, **MultiAtt**, **HBT**, and **Transformer**), demonstrating the advantage of the hard attention mechanism in this work over the soft attention for ED.

3.3 Evaluation on TAC KBP 2015

For TAC KBP 2015, we compare *LearnToSelectED* with the following state-of-the-art systems (trained and evaluated on the standard split for this dataset): 1) **TAC TOP, TAC SECOND, TAC THIRD**: the three best systems for the Event Nugget Detection task of the TAC KBP 2015 evaluation [19], 2) **SSED**: the supervised structured event detection model with semantic features and predicates from semantic role labeling [29], 3) **MSEP**: the minimally supervised event pipeline model [29], 4) **GCNN-ANC**: the graph-based CNN model with anchor-based pooling [27], and 5) **GCNN-ENT**: the graph-based CNN model with entity mention-based pooling [27]. The results are shown in Table 2.

Model	P	R	F
TAC TOP [19]	75.2	47.7	58.4
TAC SECOND [19]	74.0	46.6	57.2
TAC THIRD [19]	73.7	44.9	55.8
SSED [29]	69.9	48.8	57.5
MSEP [29]	69.2	47.8	56.6
GCNN-ANC [27]	67.3	50.8	57.9
GCNN-ENT [27]	70.3	50.6	58.8
LearnToSelectED	63.7	58.7	**61.1**

Table 2. Performance on TAC KBP 2015.

It is clear from the table that *LearnToSelectED* is significantly better than the previous ED models on the TAC KBP 2015 dataset. The performance improvement is 2.7% on the absolute F1 score over the best reported system in the TAC KBP 2015 evaluation. In addition, *LearnToSelectED* is 2.3% better than **GCNN-ENT**, the recent state-of-the-art system for ED that explicitly selects and models the relevant context words via the syntactic structures. This further confirms the advantage of automatic word selection in this work.

3.4 Analysis

There are two important components for the operation of the *LearnToSelectED* model, i.e., the function cm to compose the overall representation vector v^i and word representation u^i_j in Section 2.2 based on the LSTM unit (called **LSTM-Compose**), and the procedure to select relevant words in Section 2.2 with the STGS estimator (called **LearnSelection**). We write *LearnToSelectED* = **LSTM-Compose** + **LearnSelection** to summarize the main components in *LearnToSelectED*. This section aims to demonstrate the importance of these two components for *LearnToSelectED* in this work. In particular, we try different variations of these two components to see how well they compare with *LearnToSelectED*. For the **LSTM-Compose** component, an alternative, called **RNN-Compose** is to avoid the gating mechanisms in cm and only use a simple feed-forward network to transform the concatenation of v^i and u^i_j into a new vector as in vanilla RNN units. Regarding the **LearnSelection** component, the possible variations include the fixed strategies to select important words for ED motivated from the previous work. Specifically, we consider the following fixed strategies for important word selection in this section: 1) **All**: selecting all the words in the sentences, 2) **Entity**: selecting the words within the entity mentions of the sentences, motivated by [16], 3) **Syntax**: selecting the words surrounding the anchor words in the dependency trees, motivated by [27], 4) **Entity-Syntax**: selecting the words in the unions of those for **Entity** and **Syntax** [27], and 5) **Window**: selecting the adjacent words in a window of size win of the anchor words. The best value of win we find via the ACE 2005 development dataset for this variation is $win = 5$. Note that once a set of words is selected according to one of those five strategies, we perform the composition function cm over these words from left to right, keeping their natural order in the sentences. The anchor words are always used to initialize the process. Table 3 presents the performance of *LearnToSelectED* on the ACE 2005 development dataset when its components are replaced by the variations. The main observations from the table include: (i)

Model	P	R	F
LSTM-Compose + LearnSelection (a.k.a *LearnToSelectED*)	74.6	71.5	73.0
RNN-Compose + LearnSelection	73.6	65.8	69.5
LSTM-Compose + All	73.3	63.6	68.1
LSTM-Compose + Entity	74.4	67.8	71.0
LSTM-Compose + Syntax	75.7	66.7	70.9
LSTM-Compose + Entity-Syntax	74.9	69.5	72.1
LSTM-Compose + Window	76.2	67.6	71.6

Table 3. Ablation Study. Performance on the ACE 2005 dev set.

the **LSTM-Compose** composition is very important for *LearnToSelectED* as replacing it with **RNN-Compose** would worsen the performance significantly, (ii) among the five variations of **LearnSelection**, **All** has the worst performance as it composes all the words in the sentences, potentially encoding many noisy and irrelevant words, (iii) **Entity** and **Syntax** seem to complement each other as they do not perform well when they are applied individually. However, once they are combined (i.e., **Enity-Syntax**) as in [27], the performance becomes much better, (iv) **Window** has better performance than **Entity** and **Syntax**, but it is not enough to produce the best performance, and (v) most importantly, **LearnSelection** significantly outperforms its five variations with $p < 0.05$, eventually demonstrating the effectiveness of *LearnSelection* for ED.

4 Related Work

ED has been studied extensively in the last decade. The early machine learning approach for ED has mainly hand-designed features to capture linguistic structures of event mentions [11–13, 34]. Recently, a greater attention has been put on deep learning where various networks are proposed to solve ED. The typical network components include CNN [2, 1, 25, 23], RNN [22, 28], attention mechanisms [16, 14, 20], syntactic structures [27, 31], Generative Adversarial Networks [9] and document-level representation [3]. These prior work often captures the important context words via the soft attention mechanism and/or the fixed word selection rules. However, none of these explicitly learns to select and model important words within sentences in a hard attention style as we do.

5 Conclusion

We present a novel method for ED that features the automatic identification of important context words in the sentences. It performs a sequence of word selections for each sentence conditional on the previously chosen words. The Gumbel-Softmax trick is employed to handle the discrete variables for training.

The proposed model achieves the state-of-the-art performance on both the general setting and the cross-domain setting for ED. In the future, we plan to extend and evaluate the proposed model to the related tasks (e.g., relation extraction).

Acknowledgments

This research has been supported in part by Vingroup Innovation Foundation (VINIF) in project code VINIF.2019.DA18 and Adobe Research Gift. This research is also based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

References

1. Chen, Y., Liu, S., Zhang, X., Liu, K., Zhao, J.: Automatically labeled data generation for large scale event extraction. In: ACL (2017)
2. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: ACL-IJCNLP (2015)
3. Chen, Y., Yang, H., Liu, K., Zhao, J., Jia, Y.: Collective event detection via a hierarchical and bias tagging networks with gated multi-level attention mechanisms. In: EMNLP (2018)
4. Choi, J., Yoo, K.M., Lee, S.: Learning to compose task-specific tree structures. In: AAAI (2018)
5. Chung, J., Ahn, S., Bengio, Y.: Hierarchical multiscale recurrent neural networks (2017)
6. Feng, X., Huang, L., Tang, D., Ji, H., Qin, B., Liu, T.: A language-independent neural network for event detection. In: ACL (2016)
7. Gu, J., Im, D.J., Li, V.O.K.: Neural machine translation with gumbel-greedy decoding. In: AAAI (2018)
8. Gumbel, E.J.: Statistical theory of extreme values and some practical applications: A series of lectures. In: Number 33. US Govt. Print. Office, 1954 (1954)
9. Hong, Y., Zhou, W., zhang jingli, j., Zhou, G., Zhu, Q.: Self-regulation: Employing a generative adversarial network to improve event detection. In: ACL (2018)
10. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. In: ICLR (2017)
11. Ji, H., Grishman, R.: Refining event extraction through cross-document inference. In: ACL (2008)

12. Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: ACL (2013)
13. Li, X., Nguyen, T.H., Cao, K., Grishman, R.: Improving event detection with abstract meaning representation. In: Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS) (2015)
14. Liu, J., Chen, Y., Liu, K., Zhao, J.: Event detection via gated multilingual attention mechanism. In: AAAI (2018)
15. Liu, S., Chen, Y., He, S., Liu, K., Zhao, J.: Leveraging framenet to improve automatic event detection. In: ACL (2016b)
16. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting argument information to improve event detection via supervised attention mechanisms. In: ACL (2017)
17. Maddison, C.J., Tarlow, D., Minka, T.: A$*$ sampling. In: NeurIPS (2014)
18. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS (2013b)
19. Mitamura, T., Liu, Z., Hovy, E.: Overview of tac kbp 2015 event nugget track. In: TAC (2015)
20. Nguyen, M., Nguyen, T.H.: Who is killed by police: Introducing supervised attention for hierarchical lstms. In: COLING (2018b)
21. Nguyen, T.H., , Meyers, A., Grishman, R.: New york university 2016 system for kbp event nugget: A deep learning approach. In: TAC (2016e)
22. Nguyen, T.H., Cho, K., Grishman, R.: Joint event extraction via recurrent neural networks. In: NAACL (2016a)
23. Nguyen, T.H., Fu, L., Cho, K., Grishman, R.: A two-stage approach for extending event detection to new types via neural networks. In: Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP) (2016b)
24. Nguyen, T.H., Grishman, R.: Relation extraction: Perspective from convolutional neural networks. In: Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM) (2015a)
25. Nguyen, T.H., Grishman, R.: Event detection and domain adaptation with convolutional neural networks. In: ACL-IJCNLP (2015b)
26. Nguyen, T.H., Grishman, R.: Modeling skip-grams for event detection with convolutional neural networks. In: EMNLP (2016d)
27. Nguyen, T.H., Grishman, R.: Graph convolutional networks with argument-aware pooling for event detection. In: AAAI (2018a)
28. Nguyen, T.M., Nguyen, T.H.: One for all: Neural joint modeling of entities and events. In: AAAI (2019)
29. Peng, H., Song, Y., Roth, D.: Event detection and co-reference with minimal supervision. In: EMNLP (2016)
30. Pouran Ben Veyseh, A., Nguyen, T.H., Dou, D.: Graph based neural networks for event factuality prediction using syntactic and semantic structures. In: ACL (2019)
31. Sha, L., Qian, F., Chang, B., Sui, Z.: Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In: AAAI (2018)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS (2017)
33. Williams, A., Drozdov, A., Bowman, S.R.: Do latent tree learning models identify meaningful structure in sentences? In: TACL (2018)
34. Yang, B., Mitchell, T.M.: Joint extraction of events and entities within a document context. In: NAACL-HLT (2016)
35. Zhao, Y., Jin, X., Wang, Y., Cheng, X.: Document embedding enhanced event detection with hierarchical and supervised attention. In: ACL (2018)