Collective Event Detection via a Hierarchical and Bias Tagging Networks with Gated Multi-level Attention Mechanisms

Yubo Chen[1], **Hang Yang**[1], **Kang Liu**[1,2], **Jun Zhao**[1,2] and **Yantao Jia**[3]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Huawei Technologies Co., Ltd, Beijing, 100085, China
{yubo.chen, hang.yang, kliu, jzhao}@nlpr.ia.ac.cn, jamaths.h@163.com

Abstract

Traditional approaches to the task of ACE event detection primarily regard multiple events in one sentence as independent ones and recognize them separately by using sentence-level information. However, events in one sentence are usually interdependent and sentence-level information is often insufficient to resolve ambiguities for some types of events. This paper proposes a novel framework dubbed as **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBTNGMA) to solve the two problems simultaneously. Firstly, we propose a hierarchical and bias tagging networks to detect multiple events in one sentence collectively. Then, we devise a gated multi-level attention to automatically extract and dynamically fuse the sentence-level and document-level information. The experimental results on the widely used ACE 2005 dataset show that our approach significantly outperforms other state-of-the-art methods.

1 Introduction

Event detection (ED) is a crucial subtask of event extraction, which aims to identify event triggers and classify them into specific types from texts. According to the task defined in Automatic Context Extraction[1] (ACE), given the following sentence S1, a robust ED system should be able to recognize two events: a *Die* event triggered by *died* and an *Attack* event triggered by *fired*.

S1: *In Baghdad, a cameraman* ***died*** *when an American tank* ***fired*** *on the Palestine Hotel.*

To this end, most methods (Ahn, 2006; Hong et al., 2011; Chen et al., 2015; Nguyen and Grishman, 2016; Liu et al., 2017) model ED as a multi-classification task and predict every word in the sentence separately to determine whether it triggers a specific type of event by using sentence-level information. However, they face two problems: (1) Neglecting event interdependency by separately predicting each event; (2) Sentence-level information is usually insufficient to resolve ambiguities for some types of events. In the following, we will use examples to illustrate these two problems specifically.

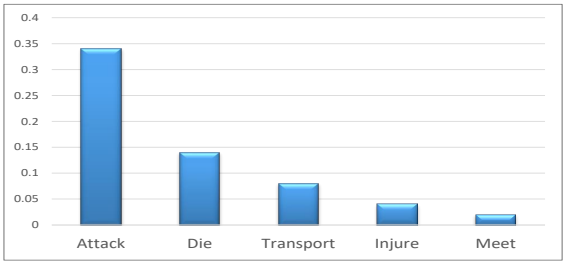

Figure 1: Top 5 event types that co-occur with *Attack* event in the same sentence in ACE 2005.

S2: *The project leader was* ***fired*** *for the* ***bankruptcy*** *of the subsidiary company.*

Event interdependency: In S1, *fired* triggers an *Attack* event, while it triggers an *End-Position* event in S2. Because of the ambiguity, a traditional approach may mislabel *fired* in S1 as a trigger of *End-Position* event. However, if we know *died* triggers a *Die* event in S1, which is easier to disambiguate, we tend to predict that *fired* triggers an *Attack* event. The reason is that the events mentioned in the same sentence tend to be semantically coherent and a *Die* event usually co-occurs with an *Attack* event. The similar phenomenon can be found in S2. We conduct a statistical analysis on ACE 2005 dataset, and find that nearly 30% sentences contain multiple events which is a proportion we can not ignore. To give an intuitive illustration, the top 5 event types that co-occur with *Attack* event in the same sentence are shown in Figure 1. We call such clues as event

[1]http://projects.ldc.upenn.edu/ace/

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1267–1276
Brussels, Belgium, October 31 - November 4, 2018.

interdependency. Some works (Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2016b) rely on a set of elaborately designed features and complicated natural language processing (NLP) tools to capture event interdependency. However, these methods lack generalization, take a large amount of human effort and are prone to error propagation problem. Though Nguyen et al. (2016) use a Recurrent Neural Networks (RNN) based classification model to capture the event interdependency between current event candidate and the former (left) predicted events, they miss the event interdependency between current event candidate and the later (right) predicted events, and the later events can not change the type of current event. The reason is that they classify the words of the sentence from left to right one by one and only use the former events to predict the later event types. We claim that both of the former and later predicted events are important to predict the event type of current trigger candidate. For example in S1, the former predicted *Die* event can help us to predict that *fired* triggers an *Attack* event, while in S2 the later predicted *bankruptcy* event can help us to predict that *fired* triggers an *End-Position* event. Thus, how to use a neural-based model to capture all event interdependencies (the interdependencies between the current event candidate and its former/later predicted events) in the whole sentence is a challenging problem.

Sentence-level and document-level information: Besides event interdependency, knowing that *American tank* is a weapon can also give us additional evidence to predict that *fired* triggers an *Attack* event in S1. Similarly in S2, knowing that *project leader* is a job title can also help us to predict that *fired* triggers an *End-Position* event. We call such clues as sentence-level information. However, sometimes it is difficult even for people to classify event types from an isolated sentence. We must resort to document-level information. For example, considering the following sentence with an ambiguous word *left*:

S3: *He **left** the company.*

It is hard to tell *left* triggers a *Transport* event which means that he left the place, or an *End-Position* event which means that he resigned from the company. However, if we read the whole document, a clue like "*He planned to go shopping before he went home, because he got off work early today.*" would give us more confidence to believe that *left* triggers a *Transport* event, while a clue like "*They held a party for his retirement.*" would indicate the aforementioned event is an *End-Position* event. We call such clues as document-level information. Moreover, the confidence of sentence-level and document-level information should be taken into consideration when using them together to construct a broader range of contextual information. For example in S3, document-level information will give us more evidence, while in S1 sentence-level information is enough to disambiguate the types of events. There have been some feature-based studies (Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012) that construct rules to capture document-level information for improving sentence-level ED. However, they suffer from two problems: (1) The features they used often need to be manually designed and may involve error propagation from existing NLP tools; (2) Sentence-level and document-level information are integrated by a large number of fixed rules, which is complicated to construct and it will be far from complete. Thus, how to use a neural-based model to automatically extract sentence-level and document-level information and dynamically integrate them is another challenging problem.

In this paper, we propose a **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBTNGMA) to address the two problems stated above simultaneously. To capture event interdependency and collectively detect multiple events in one sentence, we propose a hierarchical and bias tagging networks for event detection. In which, we exploit a hierarchical RNN-based tagging layer to capture all event interdependencies in the whole sentence and devise a bias objective function to reinforce the influence of trigger tags on the model[2]. To use a broader range of contextual information of the event candidate, we propose a gated multi-level attention, which can automatically extract sentence-level and document-level information and integrate them dynamically. In summary, the contributions of this paper are as follows:

- We propose a novel framework for event detection, which can automatically extract

[2]Compared with the task like Named Entities Recognition, the number of "O" tags is much more than the number of trigger tags in ED task, i.e. if we use the non-bias objective function and tag all words in one sentence as "O", we will gain a low loss. Thus we devise a bias objective function.

and dynamically integrate sentence-level and document-level information and collectively detect multiple events in one sentence.

- To capture event interdependency, we exploit a hierarchical and bias tagging networks to detect multiple events in one sentence collectively. To automatically extract and dynamically integrate contextual information, we devise a gated multi-level attention Mechanisms. To our knowledge, this is the first work to jointly use event interdependency, sentence-level information and document-level information via a neural tagging schema for event detection task.
- We conduct extensive experiments on a widely used ACE 2005 dataset, and the experimental results show that our approach significantly outperforms other state-of-the-art methods [3].

2 Task Description

Event detection (ED) is a crucial subtask of event extraction (EE). In this paper, we focus on ED task defined in ACE evaluation, where an event is defined as a specific occurrence involving one or more participants. Firstly, we introduce some ACE terminology to facilitate the understanding of this task: **Event trigger**: the main word or phrase that most clearly expresses the occurrence of an event. **Event arguments**: the mentions that are involved in an event (viz., participants). **Event mention**: a phrase or sentence within which an event is described, including a trigger and arguments.

Given an English text document, an ED system should identify event triggers and categorize their event types for each sentence. For instance, in the sentence "*He died in the hospital*", an ED system is expected to detect a *Die* event along with the trigger word "*died*". The ACE 2005 evaluation defines 8 event types and 33 subtypes, such as *Attack* or *Die*. Following previous works (Li et al., 2013; Chen et al., 2015; Liu et al., 2017; Nguyen and Grishman, 2016), we categorize triggers into these 33 subtypes.

3 Methodology

In this paper, we formulate event detection as a sequence labelling task. As shown in Figure 2, we label all words in one sentence collectively via a **H**ierarchical and **B**ias **T**agging **N**etworks with **G**ated **M**ulti-level **A**ttention Mechanisms (HBTNGMA). We assign a tag for each word to indicate whether it triggers a specific type of event. We adopt the "BIO" tags schema, where tag "O" represent the "other" tag which means that the corresponding word does not trigger any event, tags "B-EventType" and "I-EventType" represent the "Begin-EventType" and "Inside-EventType" tag respectively. "EventType" means that the word triggers a specific type of event. "B" and "I" represent the position of the word in a trigger to solve the problem that a trigger word contains multiple words such as "take over", "go off" and so on. Thus, the total number of tags is $N_t = 2 * |N_{eventType}| + 1$, where $|N_{eventType}|$ is the size of the predefined event types and $|N_{eventType}| = 33$ in this paper as stated above.

Figure 2 describes the architecture of HBTNGMA, which primarily involves the following four components: (i) embedding layer, which transforms each word into a continuous vector; (ii) BiLSTM layer, which uses a Bidirectional Long Short Term Memory (BiLSTM) to encode the semantics of each word considering the forward and backward information; (iii) gated multi-level attention, in which we propose a sentence-level and document-level attention to automatically extract sentence-level and document-level information respectively, and we devise a fusion gate to dynamically integrate them as context information; and (iv) Hierarchical tagging layer, in which we propose two Tagging LSTM (TLSTM1 and TLSTM2) and a tagging attention to automatically capture the event interdependency and tag all the words of the sequence collectively.

3.1 Embedding Layer

This paper uses the learned word embeddings as the source of basic features. Specifically, we use the Skip-gram model (Mikolov et al., 2013) to learn word embeddings on the NYT corpus.

Given a document $d = \{s_1, s_2, ..., s_i, ..., s_{N_s}\}$, where N_s is the number of sentences in the document. The i-th sentence s_i can be represented as token sequence $s_i = \{w_1, w_2, ..., w_t, ..., w_{N_w}\}$, where N_w is length of the sentence and w_t is the t-th token of the sentence. Assume that the word embedding for token w_t is e_t and we use it as the input of the following layer.

[3] Our source code, including all hyper-parameter settings and pre-trained word embeddings, is openly available at https://github.com/yubochen/NBTNGMA4ED

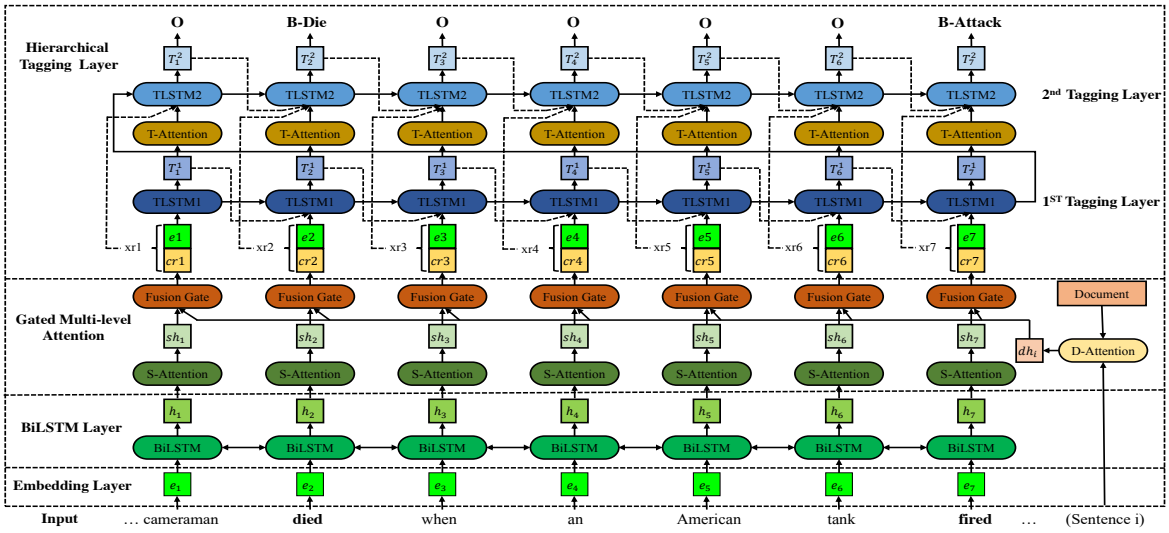

Figure 2: The architecture of our proposed hierarchical and bias tagging networks with gated multi-level attention.

3.2 BiLSTM Layer

In sequence labelling problems, the BiLSTM has been proven effective to capture the semantic information of each word (Lample et al., 2016). In this paper, we use the LSTM unit as described in (Zaremba and Sutskever, 2014). For each word w_t, the forward LSTM encodes w_t by considering the contextual information from word w_1 to w_t, which is marked as $\overrightarrow{h}_t$. Similarly, the backward LSTM encodes w_t based on the contextual information from w_{N_w} to w_t, which is marked as $\overleftarrow{h}_t$. Finally, we concatenate $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ to represent the information of the word w_t, denoted as $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$, and we concatenate $\overrightarrow{h}_{N_w}$ and $\overleftarrow{h}_1$ to represent the encoding information of the whole sentence s_i, denoted as $h_{s_i} = [\overrightarrow{h}_{N_w}, \overleftarrow{h}_1]$.

3.3 Gated Multi-level Attention

Gated multi-level attention primarily involves the following three components: (i) sentence-level attention layer, which automatically captures important sentence-level information by considering the current word; (ii) document-level attention layer, which automatically captures important document-level information by considering the current sentence; and (iii) fusion gate layer, which use a fusion gate to dynamically integrate sentence-level and document-level information.

Sentence-level Attention Layer

Sentence-level attention layer aims to capture the important clues in sentence level. For each candidate word w_t in the sentence, its sentence-level semantic information sh_t is calculated as follows:

$$sh_t = \sum_{k=1}^{N_w} \alpha_s^k h_k \tag{1}$$

where α_s^k is the weight of each word representation h_k. In this paper, we define α_s^k as following:

$$\alpha_s^k = \frac{exp(z_s^k)}{\sum_{j=1}^{N_w} exp(z_s^j)} \tag{2}$$

where z_s^k is the relatedness between the t-th word representation h_t and the k-th word representation h_k, modeled by bilinear attention as:

$$z_s^k = \tanh(h_t W_{sa} h_k^T + b_{sa}) \tag{3}$$

where W_{sa} is the weight matrix and b_{sa} is the bias term. Following above sentence-level attention mechanism, we can get the sentence-level information for each word w_t by considering the semantic information of the word w_t.

Document-level Attention Layer

Similar to sentence-level attention, document-level attention captures the vital clues in the document level. The document-level semantic information dh_i for i-th sentence calculated as follows:

$$\begin{aligned} dh_i &= \sum_{k=1}^{N_s} \alpha_d^k h_{s_k} \\ \alpha_d^k &= \frac{exp(z_d^k)}{\sum_{j=1}^{N_s} exp(z_d^j)} \\ z_d^k &= \tanh(h_{s_i} W_{da} h_{s_k}^T + b_{da}) \end{aligned} \tag{4}$$

where α_d^k is the weight of each sentence representation h_{s_k}, z_d^k is the relatedness between i-th

sentence representation h_{s_i} and the k-th sentence representation h_{s_k}, W_{da} is the weight matrix and b_{da} is the bias term. Compared with sentence-level information, all words in the i-th sentence have the same document-level information dh_i.

Fusion Gate Layer

We devise a fusion gate to dynamically integrate sentence-level information sh_t and document-level information dh_i for the t-th word w_t in the i-th sentence s_i, and calculate the contextual information representation cr_t as follows:

$$cr_t = (G_t \odot sh_t) + ((1 - G_t) \odot dh_i) \tag{5}$$

where G_t is a fusion gate aims to model the confidence of clues provided by sentence-level information sh_t and document-level information dh_i, which is calculated as follows:

$$G = \sigma(W_g[sh_t, dh_i] + b_g) \tag{6}$$

where W_g is the weight matrix and b_g is the bias term, σ is a $sigmoid$ function and $\odot$ denotes element-wise multiplication. Finally, the contextual information cr_t of word w_t and its word embedding e_t are concatenated into a single vector $xr_t = [e_t, cr_t]$ as the feature representation of w_t.

3.4 Hierarchical Tagging Layer

In hierarchical tagging layer, we propose two Tagging LSTMs (TLSTM1 and TLSTM2) and a tagging attention to automatically capture the event interdependency and tag the sequence collectively.

The First Tagging Layer: TLSTM1

When detecting the tag of word w_t in TLSTM1, the inputs are: the feature representation xr_t obtained from embedding layer and gated multi-level attention layer, former predicted tag vector T^1_{t-1}, and former hidden vector h^1_{t-1} in TLSTM1. The detail operations are defined as follows:

$$\begin{aligned}
i^1_t &= \sigma(W^1_{i_x} xr_t + W^1_{i_h} h^1_{t-1} + W^1_{i_T} T^1_{t-1}) \\
f^1_t &= \sigma(W^1_{f_x} xr_t + W^1_{f_h} h^1_{t-1} + W^1_{f_T} T^1_{t-1}) \\
o^1_t &= \sigma(W^1_{o_x} xr_t + W^1_{o_h} h^1_{t-1} + W^1_{o_T} T^1_{t-1}) \\
u^1_t &= \varphi(W^1_{u_x} xr_t + W^1_{u_h} h^1_{t-1} + W^1_{u_T} T^1_{t-1}) \\
c^1_t &= i^1_t \odot u^1_t + f^1_t \odot c^1_{t-1} \\
h^1_t &= o^1_t \odot \varphi(c^1_t) \\
T^1_t &= W^1_T h^1_t + b^1_T
\end{aligned} \tag{7}$$

Where i_t is an input gate, u_t is an input modulation gate, f_t is a forget gate, o_t is an output gate, c_t is a memory cell and T^1_t is a predicted tagging vector.

The Second Tagging Layer: TLSTM2

Though the TLSTM1 can capture the interdependency between current event candidate and the former predicted event tags, it can not capture the interdependency between the current event candidate and the later predicted event tags. Thus, we devise a second tagging layer (TLSTM2) upon the LSTM1 to capture the interdependency between the current event candidate and both of former and later predicted event tags from TLSTM1. When detecting the tag of word w_t in TLSTM2, the inputs are: the feature representation xr_t, former predicted tag vector T^2_{t-1} in TLSTM2, the preliminary predicted information T^a_t calculated from TLSTM1, and former hidden vector h^2_{t-1} in TLSTM2. The detail operations are defined as follows:

$$\begin{aligned}
i^2_t &= \sigma(W^2_{i_x} xr_t + W^2_{i_h} h^2_{t-1} + W^2_{i_T} T^2_{t-1} + W^2_{i_a} T^a_t) \\
f^2_t &= \sigma(W^2_{f_x} xr_t + W^2_{f_h} h^2_{t-1} + W^2_{f_T} T^2_{t-1} + W^2_{f_a} T^a_t) \\
o^2_t &= \sigma(W^2_{o_x} xr_t + W^2_{o_h} h^2_{t-1} + W^2_{o_T} T^2_{t-1} + W^2_{o_a} T^a_t) \\
u^2_t &= \varphi(W^2_{u_x} xr_t + W^2_{u_h} h^2_{t-1} + W^2_{u_T} T^2_{t-1} + W^2_{u_a} T^a_t) \\
c^2_t &= i^2_t \odot u^2_t + f^2_t \odot c^2_{t-1} \\
h^2_t &= o^2_t \odot \varphi(c^2_t) \\
T^2_t &= W^2_T h^2_t + b^2_T
\end{aligned} \tag{8}$$

The unit structure of TLSTM2 is similar to the unit of TLSTM1. The parts need to pay attention are follows: (1) the initial hidden input h^2_0 of the TLSTM2 is the last hidden vector $h^1_{N_w}$ of the TLSTM1. (2) the preliminary predicted information T^a_t is calculated from TLSTM1 by using a tagging attention as follows.

Tagging Attention

Tagging attention aims to automatically encode the preliminary predicted information T^a_t for the word w_t and the details are as follows:

$$\begin{aligned}
T^a_t &= \sum\nolimits_{k=1}^{N_w} \alpha^k_T T^1_k \\
\alpha^k_T &= \frac{exp(z^k_T)}{\sum_{j=1}^{N_w} exp(z^j_T)} \\
z^k_T &= \tanh(T^1_t W_{ta} (T^1_k)^T + b_{ta})
\end{aligned} \tag{9}$$

where α^k_T is the weight of each preliminary predicted tag T^1_k, z^k_T is the relatedness between t-th preliminary predicted tag T^1_t and the k-th preliminary predicted tag T^1_k, W_{ta} is the weight matrix and b_{ta} is the bias term.

The final normalized tag probability for word w_t is based on the predicted tag vector T^2_t from

TLSTM2 and computed as follows:

$$O_t = W_y T_t^2 + b_y$$
$$p(O_t^i|s_j,\theta) = \frac{exp(O_t^i)}{\sum_{k=1}^{N_t} exp(O_t^k)} \quad (10)$$

where $p(O_t^i|s_j,\theta)$ is the probability that assigning the i-th tag to word w_t in sentence s_j when parameters is θ, and N_t is the total number of tags.

3.5 Training with Bias Objective Function

In one sentence, the number of "O" tags is much more than the number of trigger tags. Thus, we devise a bias objective function $J(\theta)$ to reinforce the influence of trigger tags on the model, which is defined as follows:

$$J(\theta) = max \sum_{j=1}^{N_{ts}} \sum_{t=1}^{N_w} (\log p(O_t^{y_t}|s_j,\theta) \cdot I(O) + \alpha \log p(O_t^{y_t}|s_j,\theta) \cdot (1 - I(O))) \quad (11)$$

where N_{ts} is the number of training sentences, N_w is the length of sentence s_j, $p(O_t^{y_t}|s_j,\theta)$ is the normalized probabilities of tags defined in Formula 10 and y_t is the golden tag of word w_t in sentence s_j, α is the bias weight and the larger α will bring the greater influence of trigger tags on the model. Besides, $I(O)$ is a switching function to distinguish the loss of tag "O" and trigger tags, which is defined as follows:

$$I(O) = \begin{cases} 1, & if \quad tag = \text{“}O\text{”} \\ 0, & if \quad tag \neq \text{“}O\text{”} \end{cases} \quad (12)$$

To compute the network parameter θ, we maximize the log likelihood $J(\theta)$ through stochastic gradient descent over shuffled mini-batches with the Adadelta (Zeiler, 2012) update rule.

4 Experiments

4.1 Experimental Setting

Dataset and Evaluation Metrics

We conduct experiments on the widely used ACE 2005 dataset. For comparison, as the same as previous works (Liao and Grishman, 2010; Li et al., 2013; Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2017), we used the same test set with 40 documents and the same development set with 30 documents and the rest 529 documents are used for training. Finally, we use *Precision (P)*, *Recall (R)* and *F measure (F_1)* as the evaluation metrics as the same as previous work.

Hyper-parameter Setting

Hyper-parameters are tuned on the development dataset by grid search. We train the word embedding using Skip-gram algorithm [4] on the NYT corpus [5]. We set the dimension of word embeddings as 100, the dimension of tag vector as 20, all the size of LSTM in BiLSTM layer, TLSTM1 and TLSTM2 layer as 100, the bias parameter α in Formula 11 as 5, the batch size as 20, the learning rate as 0.001, the dropout rate as 0.5.

4.2 Our Method vs. State-of-the-art Methods

We select the following state-of-the-art methods for comparison, which can be classified as two types: separate and collective methods:

Separate methods: 1) **Li's MaxEnt**: the method that detects events in one sentence separately by using human-designed features (Li et al., 2013). 2) **Liao's CrossEvent** : the method that uses cross event information (Liao and Grishman, 2010). 3) **Hong's CrossEntity**: the method that uses cross entity information (Hong et al., 2011). 4) **Chen's DMCNN**: the dynamic multi-pooling convolutional neural networks method (Chen et al., 2015). 5) **Chen's DMCNN+**: the DMCNN method argumented with automatically labeled data (Chen et al., 2017). 6) **Liu's FrameNet** : the method that leverages FrameNet as extended training data to improve ED (Liu et al., 2016a). 7) **Liu's ANN-Aug**: the method that use the annotated argument information via a supervised attention to improve ED (Liu et al., 2017).

Collective methods: 1) **Li's Structure**: the method that collectively detects events by using human-designed features (Li et al., 2013). 2) **Yang's JointEE**: the method that detects events and entities in one sentence jointly based on human-designed features (Yang and Mitchell, 2016). 3) **Nguyen's JRNN**: the method that exploits a RNN model to collectively detects events by only using sentence-level information (Nguyen et al., 2016). 4) **Liu's PSL** : the method that uses a probabilistic soft logic to detect events by using human-designed features (Liu et al., 2016b).

Experimental results are shown in Table 1. From the table, we have the following observations: (1) Among all the methods, our HBTNGMA achieves the best performance. It can improve the best collective method's F_1 by

[4] https://code.google.com/p/word2vec/
[5] https://catalog.ldc.upenn.edu/LDC2008T19

Methods	P	R	F_1
Li's MaxEnt (2013)	74.5	59.1	65.9
Liao's CrossEvent (2010)	68.7	68.9	68.8
Hong's CrossEntity (2011)	72.9	64.3	68.3
Chen's DMCNN (2015)	75.6	63.6	69.1
Chen's DMCNN+ † (2017)	75.7	66.0	70.5
Liu's FrameNet † (2016a)	77.6	65.2	70.7
Liu's ANN-Aug † (2017)	76.8	67.5	71.9
Li's Structure (2013)	73.7	62.3	67.5
Yang's JointEE (2016)	75.1	63.3	68.7
Nguyen's JRNN (2016)	66.0	73.0	69.3
Liu's PSL (2016b)	75.3	64.4	69.4
Ours HBTNGMA	**77.9**	**69.1**	**73.3**

Table 1: Overall performance on blind test data. The upper table illustrates the performance of separate ED systems and the lower illustrates collective ED systems. † means the method that uses external resources.

3.9% (*Liu's PSL*) and improve the best separate method's F_1 by 1.4% (*Liu's ANN-Aug*) although *Liu's ANN-Aug* uses FrameNet as external resources. We also perform a t-test ($p \leqslant 0.05$), which indicates that our method significantly outperforms all of the compared methods. (2) Comparing our HBTNGMA to separate methods, it achieves a better performance. It proves that collectively predicting multiple events in one sentence is effective. (3) Our HBTNGMA outperforms feature-based collective methods (*Li's Structure*, *Yang's JointEE* and *Liu's PSL*), it proves that our automatically learned features can efficiently capture semantic information from plain texts. (4) Compared with Nguyen's JRNN, HBTNGMA gains a 4.0% improvement on F_1 value. The reason is that Nguyen's JRNN only uses sentence-level information while our model exploits multi-level information, and our model can capture the interdependencies between the current event candidate and its former/later predicted events simultaneously.

4.3 Effect of The Hierarchical and Bias Tagging Networks

In this subsection, we prove the effectiveness of hierarchical and bias tagging networks for collective ED. We select following methods as baselines: 1) **LSTM+Softmax**: a simplified version of our HBTNGMA, which directly use a softmax layer to separately detect events after we get the feature representation xr_t of each word w_t. 2) **LSTM+CRF**: the method is similar to our HBTNGMA, which uses a CRF layer to tag words instead of our Hierarchical TLSTM (HTLSTM) tagging layer. 3) **LSTM+TLSTM**: the method is similar to our HBTNGMA, which only use a TLSTM1 and takes all tags have same influence in training loss (i.e. α in is set as 1) . 4) **LSTM+HTLSTM**: the method is similar to our HBTNGMA, which use a HTLSTM (TLSTM1+TLSTM2) and do not use bias objective function. And *LSTM+HTLSTM+Bias* is our proposed HBTNGMA. Moreover, we divide the testing data into two parts according the event number in a sentence (single event and multiple events) and perform evaluations separately.

Method	1/1	1/N	all
LSTM+Softmax	74.7	44.6	66.8
LSTM+CRF	75.1	49.5	68.5
LSTM+TLSTM	76.8	51.2	70.2
LSTM+HTLSTM	77.9	57.3	72.4
LSTM+HTLSTM+Bias	**78.4**	**59.5**	**73.3**

Table 2: Performance of different ED systems. 1/1 means one sentence that only has one event and 1/N means that one sentence has multiple events.

Table 2 shows the results. And we have the following observations: 1) Compared with *LSTM+Softmax*, LSTM-based collective ED methods (*LSTM+CRF*, *LSTM+TLSTM*, *LSTM+HTLSTM*, *LSTM+HTLSTM+Bias*) achieves a better performance. Surprisingly, the *LSTM+HTLSTM+Bias* yields a 14.9% improvement on the sentence contains multiple events over the *LSTM+Softmax*. It proves neural tagging schema is effective for ED task especially for the sentences contain multiple events. 2) The *LSTM+TLSTM* achieve better performances than *LSTM+CRF*. And the *LSTM+HTLSTM* achieve better performances than *LSTM+TLSTM*. The results prove the effectiveness of the TLSTM layer and HTLSTM layer. 3) Compared with *LSTM+HTLSTM*, the *LSTM+HTLSTM+Bias* gains a 0.9% improvement on all sentence. It demonstrates the effectiveness of our proposed bias objective function.

4.4 Effect of The Gated Multi-level Attention

This subsection studies the effectiveness of our gated multi-level attention. We adopt same architecture of our HBTNGMA as shown in Figure 2 with different level clues as baselines: 1) **Word Only** is the method only uses word embedding e_t to identify events. 2) **Word+SA** uses sentence-level attention to capture important sentence-level information as additional clues. 3) **Word+DA** uses document-level attention to capture important document-level information as additional clues. 4) **Word+Average MA** uses both of sentence-level

and document-level attention to capture multi-level information and integrate them with a average gate (all the dimension of the fusion gate are set as 0.5), which is a special case of our proposed HBTNGMA. And *Word+Gated MA* is our proposed HBTNGMA model.

Method	P	R	F_1
Word Only	70.1	63.4	66.6
Word+SA	75.6	68.2	71.7
Word+DA	73.1	65.8	69.3
Word+Average MA	76.5	68.7	72.4
Word+Gated MA	**77.9**	69.1	**73.3**

Table 3: Performance of gated multi-level attention.

Results are shown in Table 3. From the results, we have the following observations: 1) Compared with *word only*, *Word+SA* achieves a better performance. We can make the same observation when comparing *Word+DA* with *word only*. It proves that both sentence-level and document-level information are helpful for ED task. 2) Compared with *Word+DA*, *Word+SA* achieves a better performance. It proves that in most of cases sentence-level information provides more clues than document-level information. 3) *Word+Gated MA* gains a 0.9% improvement than *Word+Average MA*. It demonstrates that the effectiveness of our fusion gate to dynamically integrate clues from multiple levels.

4.5 Case Study

Interesting Cases: Our neural tagging schema not only can model the interdependency between multiple events in one sentence as proved in Subsection 4.3, but also the "BIO" tagging schema can solve the multiple words trigger inherently. We conduct a statistical analysis on the experimental results, and find that nearly 50% cases with multiple word trigger was solved by our model. Example is shown in Figure 3.

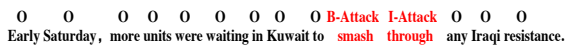

Figure 3: The example of case solved by our model.

Attention Visualization: As limited of space, we take one sentence with high sentence-level gated weight (example 1) and one sentence with high document-level gated weight (example 2) as examples for attention visualization. As shown in Figure 4, in example 1, sentence-level information plays more important role in disambiguating *fired*, and the words (*tank*, *died* and *Baghdad*) give

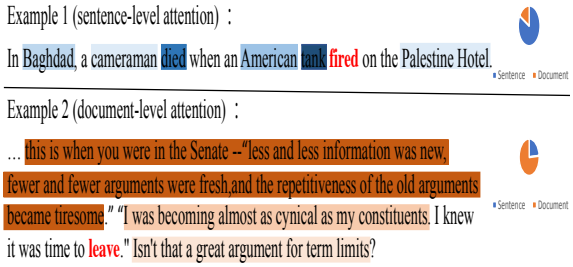

Figure 4: Attention visualization. The heat map indicate the contextual attention. Blue for sentence-level and orange for document-level. The pie chart indicate the fusion gate weight.

us ample evidence to predict that *fired* triggers an *Attack* event. While document-level information plays a more important role in example 2. The surrounding sentence "*this is ... tiresome*." gives us more confidence to predict that *leave* triggers an *End-Position* event.

5 Related Works

Event detection is an increasingly hot and challenging research topic in NLP. Generally, existing approaches could roughly be divided into two groups: separate and collective methods.

Separate methods: These methods regard multiple events in one sentence as independent ones and recognize them separately. These methods include feature-based methods which exploit a diverse set of strategies to convert classification clues into feature vectors (Ahn, 2006; Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011; Huang and Riloff, 2012), and neural-based methods which use neural networks to automatically capture clues from plain texts (Chen et al., 2015; Nguyen and Grishman, 2015; Feng et al., 2016; Nguyen and Grishman, 2016; Chen et al., 2017; Duan et al., 2017; Liu et al., 2017). Though effective these methods, they neglect event interdependency by separately predicting each event.

Collective methods: These methods try to model the event interdependency and detect multiple events in one sentence collectively. However, nearly all of these methods are feature-based methods (McClosky et al., 2011; Li et al., 2013; Yang and Mitchell, 2016; Liu et al., 2016b), which rely on elaborately designed features and suffer error propagation from existing NLP tools. Nguyen et al. (2016) exploits a neural-based method to detect multiple events collectively. However, they only use the sentence-level information and ne-

glect document-level clues, and can only capture the interdependencies between the current event candidate and its former predicted events. Moreover, there method can not handle the multiple words trigger problem.

6 Conclusion

This paper proposes a novel framework for event detection, which can automatically extract and dynamically integrate sentence-level and document-level information and collectively detect multiple events in one sentence. A hierarchical and bias tagging networks is proposed to detect multiple events in one sentence collectively. A gated multi-level attention is devised to automatically extract and dynamically integrate contextual information. The experimental results on the widely used dataset prove the effectiveness of the proposed method.

Acknowledgments

The research work is supported by the Natural Science Foundation of China (No.61533018 and No.61702512), and the independent research project of National Laboratory of Pattern Recognition. This work is also supported in part by Huawei Technologies Co., Ltd.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8.

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 (ACL)*, pages 409–419, Vancouver, Canada. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistic (ACL)*, pages 167–176.

Shaoyang Duan, Ruifang He, and Wenli Zhao. 2017. Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of IJNLP*, volume 1, pages 352–361.

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

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 (ACL)*, pages 1127–1136.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of AAAI*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 254–262.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL*, pages 260–270.

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 (ACL)*, pages 73–82.

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 (ACL)*, pages 789–797.

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu, and Jun Zhao. 2016a. Leveraging framenet to improve automatic event detection. In *Proceedings of ACL*, pages 2134–2143. Association for Computational Linguistics.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1789–1798.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016b. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of AAAI*, pages 2993–2999.

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1626–1635.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Huu Thien 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 Linguistic (ACL)*, pages 365–371.

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 (NAACL)*, pages 300–309.

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 (EMNLP)*, pages 886–891.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of NAACL*, pages 289–299, San Diego, California. Association for Computational Linguistics.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.