Leveraging Multi-Head Attention Mechanism to Improve Event Detection

Meihan Tong[1,2,3], Bin Xu[1,2,3], Lei Hou[1,2,3], Juanzi Li[1,2,3], and Shuai Wang[1,2,3]

[1] DCST, Tsinghua University, Beijing 100084, China
[2] KIRC, Institute for Artificial Intelligence, Tsinghua University
[3] Beijing National Research Center for Information Science and Technology
tongmh17@mails.tsinghua.edu.cn, xubin@tsinghua.edu.cn,
houlei@tsinghua.edu.cn, lijuanzi@tsinghua.edu.cn,
shuai-wa16@mails.tsinghua.edu.cn

Abstract. Event detection (ED) task aims to automatically identify trigger words from unstructured text. In recent years, neural models with attention mechanism have achieved great success on this task. However, existing attention methods tend to focus on meaningless context words and ignore the semantically rich words, which weakens their ability to recognize trigger words. In this paper, we propose MANN, a multi-head attention mechanism model enhanced by argument knowledge to address the above issues. The multi-head mechanism gives MANN the ability to detect a variety of information in a sentence while argument knowledge acts as a supervisor to further improve the quality of attention. Experimental results show that our approach is significantly superior to existing attention-based models.

Keywords: Event Detection · Multi-Head Attention Mechanism · Knowledge Enhancement.

1 Introduction

Event Detection (ED) intends to detect and categorize event trigger at the sentence level, and often serves as a prerequisite of Event Extraction (EE) which needs to identify its corresponding arguments simultaneously [1]. It has a wide range of applications in information retrieval, text recommendation and text summarization.

Trigger ambiguity is the major challenge in Event Detection task. For example, in the sentence "Davies is leaving to **become** chairman of the London school of economics", "become" is a trigger word indicating a "Start_Position" event. However, in the sentence "BEGALA **become** honorable by winning an election of some sort or having a high post", "become" only acts as a normal verb and does not trigger any event. If the context information of the triggers cannot be effectively perceived, it is difficult to distinguish between the two situations and simply categorizing "become" as NEGATIVE classes.

There are two branches of related researches addressing the problem, feature-based and representation-based models. Feature-based models [6, 5, 10] design

syntactic, entity-related and trigger-related features to reduce semantic confusion and leverage various classifiers for prediction. Representation-based models [1, 11, 3] determine the trigger-relevant information via dynamic-CNN, skip-CNN and memory-LSTM model and utilize full-connected neural networks to make prediction. Attention mechanism [9, 7] is an effective technique that is often incorporated into the representation-based models to improve their performance in handling trigger ambiguity. However, existing attention models still suffer from two problems, mono-attention and knowledge absence.

Mono-attention. Existing attention models [9, 7] only learn one attention weight for each context word. We call this type of model as mono-attention model. The disadvantage of mono-attention models is that they fail to absorb various information contained in a sentence. As shown in Fig. 1, without capture all of the useful words("leaving","chairman"), both of the mono-attention models mislabel "become" into NEGATIVE class. We argue that multi-head attention model is more suitable for Event Detection task. By learning multiple attention weights for each context word, multi-head attention model can capture all of the useful context words.

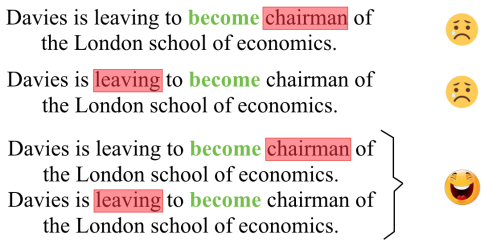

Fig. 1. Mono v.s. multiple attention. We intend to detect "become" as "Start_Position" event trigger. The first two attention weights are derived from the mono-attention model. The following attention weight is derived from our model, which can capture a wide range of information.

Knowledge absence. Existing attention models [7] lack human knowledge to directly supervise the attention weight. They usually implicitly optimize the attention weight with the final event classification target, resulting in most proportion of the attention words are meaningless words. Fig. 2 shows the difference between human attention and machine attention. Some of the researchers [9] utilize argument knowledge to guide the attention. If the model pays more attention on "chairman" (i.e., the *position* argument of the "Start-position" event), it can safely classify the trigger "become" into "Start-Position" type. We follow them to incorporate argument knowledge into our model, but the difference is that we need to supervise multiple attentions, they only need to supervise single attention.

In this paper, we propose a novel multi-head attention model enhancing by arguments information named MANN to address the above issues. Multi-head

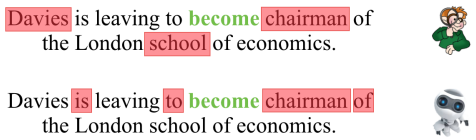

Fig. 2. Knowledge absence example. The above sentence represents the attention of human, and the following sentence represents what the machine notices.

attention mechanism allows the model to capture the context information from different aspects, and the arguments information guides the model to concentrate on human-concern words and filter out meaningless words.

MANN consists of three modules: query encoder obtains the semantic meaning of the candidate trigger and its directly adjacent words, context encoder models the sentence representation, and event detector concatenates them to feed the final classifier. Specifically, we model the multi-head attention using a scale-dot product attention network as the basic unit with each unit capturing one aspect of the sentence. Moreover, we propose an efficient unilateral supervision approach to integrate the external knowledge.

Finally, we conduct extensive experiments on the widely-used ACE2005 datasets, and the results demonstrate the effectiveness of MANN. Detailed investigation validates the effectiveness of the multi-head attention and argument knowledge integration mechanism. It is worth noting that the multi-head mechanism is more helpful for the promotion of long sentence trigger word detection.

Our contributions in this paper can be summarized as follows:

- We develop a novel multi-head attention mechanisms to capture multi-faced semantics within the sentence, including local information, action information and subject information. To the best of our knowledge, this is the first work that introduces the multi-head attention mechanism to event detection task.
- We incorporate external knowledge into the multi-head attention model via an efficient unilateral supervision approach, which significantly improves the quality of the attention.
- Extensive experiments on the ACE2005 corpus demonstrate the effectiveness of the proposed method, which achieves the best recall rate among the existing attention-based model.

2 Problem Definition

In this section, we formalize the event detection task. Before that, we first review several related basic concepts.

Definition 1 (Event). *As defined in ACE (Automatic Context Extraction) event extraction program, an event is composed of two elements: event trigger w and event arguments A. Trigger w is often a single verb or noun, most clearly*

expressing the event mention. Argument $a \in A$ could be an entity mention, temporal expression or value that serves as a participant or attribute with a specific role in an event mention.

As shown in Fig. 2, "leaving" and "become" are triggers of the **End_Position** and **Start_Position** events respectively. "Davies"(Role="People") is the argument of both events, and **Start_Position** event has two additional arguments: "chairman"(Role="Position") and "school" (Role="Entity").

Different from event extraction that is expected to extract both event trigger and arguments, event detection only detects and categorizes the event trigger, which is the focus of this paper. We first introduce the notations briefly, and then formally define the detection task.

Notations. *Given the corpus D with T sentences, each sentence $s \in D$ is denoted as a word sequence of length n, i.e., $s = \langle w_1, w_2, \ldots, w_n \rangle$. For the i-th word w_i, e_i, y_i denote its entity type and event type respectively. Because n varies in size, a window of size $L = 2l$ is introduced to fix the length. Without loss of generality, assuming w_i is the candidate trigger, then its word context and entity context are defined as $C_{w_i} = \{w_k|_{k=i-l}^{k=i+l}\}$ and $E_{e_i} = \{e_k|_{k=i-l}^{k=i+l}\}$. For convenience, we use w, C, E to represent the current candidate trigger, its word context and its entity context.*

Definition 2 (Event Detection). *Given the trigger candidate w, its word context C and entity context E, the goal is to figure out the event type y that w belongs to. Essentially, the model need to estimate the probability $p_\Phi(y|w, C, E)$.*

In this paper, we regard the detection task as a classification problem. To be specific, we treat all words in the sentence as trigger candidates. If the word is initially not an event trigger (without any event type label), we will classify it into NEGATIVE class.

3 Methodology

In this section, we present our proposed multi-head attention neural network (MANN). Fig. 3 shows its architecture, it consists of three modules: Query Encoder(QE) is a fully-connected network, aiming to encode the candidate trigger and its surrounding words. Context Encoder(CE) is the core module. It has multiple attention layers. Each attention layer is designed to focus on one aspect of information. At each layer, the candidate trigger is regarded as the query word, and the rest words are regarded as the queried words. Then they are multiplied together to obtain the final contextual representation. Event Detector(ED) absorbs the representation learned in QE and CE and assigns an event label(including NEGATIVE) to the candidate trigger via a three-layer MLP model.

Next, we will introduce module implementation details, and then highlight the approach to integrate external knowledge.

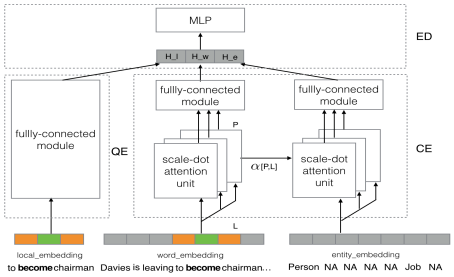

Fig. 3. The architecture of MANN

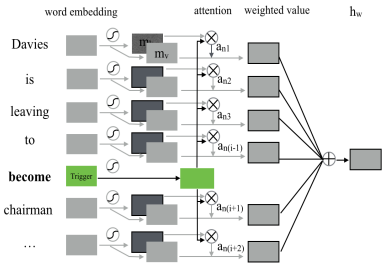

Fig. 4. The scale-dot product unit

3.1 Pre-trained Embedding

Representing word with distribution embedding has been proved to be useful in many NLP tasks. Compared with the one-hot representing, distributed representation is capable of capturing rich semantics between words. In this section, we discuss the way to obtain the word and entity embedding.

Word embedding. We choose the New York Times(NYK) corpus for word embedding training because the articles in NYK is very similar to those in the ACE2005 corpus. Following [5], we obtain the pre-trained word embedding using Skip-Gram model with the window size as 5 and the minimum word frequency as 10. The final vocabulary size is 201,370, which is large enough to cover all the words in the ACE corpus. In order to preserve the semantic information in the pre-trained embedding, we fixed the word embedding in the following steps.

Entity embedding. Entity information is quite useful for trigger disambiguation, e.g., the "End-Position" event cannot have a subject of "Country". To incorporate entity information into MANN, we adopt a uniform distribution to initialize the entity embedding and gradually optimize it during the training process.

3.2 Query Encoder

query Encoder module aims to capture the local semantics of the candidate trigger. For the candidate trigger w, we concatenate its embedding with those of its left word w_{i-1} and right word w_{i+1}, and then feed the hidden representation into a one-layer fully-connected neural network.

$$H_l = W_q[w_{i-1}; w; w_{i+1}] + b_q \tag{1}$$

where ; means the concatenation operation, H_l stands for the hidden representation of the candidate trigger w, W_q and b_q are parameters which need to be optimized during the training process.

3.3 Context Encoder

In this section, we leverage the multi-head attention mechanism to represent context words and entity information. As defined in Section 2, C and E represent

the contextual words and entities of the candidate trigger w with fixed length L. In our model, we use w and C to calculate the word contextual representation h_w, and use w and E to calculate the entity contextual representation h_e.

Word contextual. Given the candidate trigger w and its contextual words C, we exploit scale-dot attention network (SDA) to calculate the weight distribution of each attention head. To enhance the model flexibility, SDA adopts the key-value split technique. As illustrated in Fig. 4, the contextual words C are copied and assigned to C^k and C^v respectively. C^k is used in the process of calculating attention weights, as opposed to C^v in the process of calculating the final representation.

Here is the specific process of SDA. We first feed w, C^k, C^v into a linear network and activate them explicitly with the activation function. (We denote the dimensions of q, k, v as d_q, d_k, d_v respectively.)

$$\begin{aligned} q &= \sigma(W_q w + b_q) \\ \mathbf{k} &= \sigma(W_k \mathbf{C^k} + b_k) \\ \mathbf{v} &= \sigma(W_v \mathbf{C^v} + b_v) \end{aligned} \tag{2}$$

Then, we multiply q and k to obtain the attention weight α_i for the i-th word in k. It is worth noting that we add a compression operation here and the scale of the compression is proportional to d_k. This operation intends to avoid the dot-product attention being too large [16], which might push the softmax function into regions with extremely small gradients.

$$\begin{aligned} \mathbf{s} &= \frac{q \cdot \mathbf{k}}{\sqrt{d_k}} \\ \alpha_i &= \frac{s_i}{\sum_{i=1}^{L} s_i} \end{aligned} \tag{3}$$

Finally, we can obtain the word contextual representation by sum v with regard of α.

$$h_w = \alpha \mathbf{v}^{\mathrm{T}} \tag{4}$$

Now, we have obtained the word contextual representation h_w from one scale-dot attention(SDA) unit. By repeating the process multiple times, we can obtain multiple word contextual representations from multiple scale-dot attention(SDA) units. Assuming we have P SDA units, then we will get P word contextual representations for the current candidate trigger w. We concatenate them into one embedding vector and feed the vector into a fully-connected network to obtain the final word contextual representation H_w.

$$H_w = W_o[h_{w_1}; h_{w_2}; \ldots; h_{w_P}] + b_o \tag{5}$$

Entity contextual. Similarly, we first transform the entity contextual E of current candidate trigger w into the hidden representation.

$$\mathbf{u} = \sigma(W_u \mathbf{E} + b_u) \tag{6}$$

Note that we do not use w's entity type as keyword to calculate the attention weight once again. Instead, we share the attention weight α between the word and entity context.

$$h_e = \alpha \mathbf{u}^{\mathrm{T}} \tag{7}$$

We do this for two reasons. On the one hand, most entity types are NEGATIVE, which cannot reflect the semantics of the query. On the other hand, entity embedding is randomly initialized and does not include any semantic information while word embedding contains a lot.

Finally, we feed the hidden representation h_e into multiple SDA units to obtain different entity contextual representation and integrate them via the linear network.

$$H_e = W_o[h_{e_1}; h_{e_2}; \ldots; h_{e_P}] + b_o \tag{8}$$

3.4 Event Detector

In this section, we aim to illustrate the event detector(ED) module. We will first introduce the basic ED model, then we analyze the characteristics of multi-head attentions and exploit external knowledge to supervise the attention.

Basic MLP model. As illustrated in Fig. 3, we first splice the local representation H_l, word representation H_w and entity representation H_e, i.e., $H = \sigma(H_l \oplus H_c \oplus H_e)$, and then feed H into a three-layer perception model [4], which has proven to be very effective for event detection task [1, 8].

Let $x = \langle w, C, E \rangle$ denote a training sample, where w, C, E represent the candidate trigger as well as its word and entity context, H denote its hidden representation (i.e., the input of the MLP model), MLP will output a result vector O, where the k-th entry of O represents the probability that x belongs to the corresponding event class. Specifically, the conditional probability is calculated by softmax function.

$$P(y_t|x_t) = \frac{exp(o_{tk})}{\sum_{k=1}^{K} exp(o_{tk})} \tag{9}$$

Given the input corpus $D = \{x_t, y_t\}|_{t=1}^{T}$, the negative loss function is defined as:

$$J(\theta) = -\sum_{t=1}^{T} \log p(y_{(t)}|x_{(t)}, \theta) \tag{10}$$

The model can be optimized using SGD with Adadelta rule. To prevent overfitting, we adopt L_2 norm and dropout mechanism.

3.5 Argument-Enhanced Event Detector

In this section, we introduce how to leverage the argument knowledge to guide the attention.

Gold Attention Generation. Liu [9] proved that argument information is very useful to supervise the attention. We follow them to use a Gaussian distribution to generate the gold attention.

Multi-Head Attention Integration . Existing attention models normally have one attention layer, while MANN leverages multi-head attention mechanism. To facilitate external knowledge incorporation, we need to transform multiple attentions into one representation. Thus we develop the following two methods:

I1: Sum Supervision assumes that the arguments knowledge is generated after people comprehensively analyze all the information in the sentence. Under this assumption, we integrate multi-head attention by adding all the attention together.

I2: Unilateral Supervision assumes that argument words attention is an aspect of the multi-head attention. Without loss of generality, we choose the first head attention as argument words attention.

Jointly detection model. Given the gold attention α^* and the integrated attention α', we adopt Mean Square Error(MSR) to define the attention loss function.

$$A = \sum_{t=1}^{T} (\alpha_t^* - \alpha_t') \tag{11}$$

Combining the classification loss and the attention loss, We rewrite the final loss function as follows:

$$J^{'}(\theta) = J(\theta) + \lambda A \tag{12}$$

where λ controls the weight of the supervised attention loss.

4 Experiment

In this section, we evaluate the proposed MANN using widely-used benchmark ACE2005. We will first introduce the experimental settings, then present the comparison results, and finally investigate some method details.

4.1 Experimental Settings

Datasets. As mentioned previously, ACE2005 is chosen as the evaluation benchmark and New York Times corpus is used for pre-trained word embedding training. ACE2005 task defines 8 superclasses and 33 subclasses of the event. In the experiment, we ignore the hierarchical information and regard it as a 34 multiple classification problem (33 subclasses and 1 NEGATIVE class). Note that we exclude the NEGATIVE class when computing the model precision. Following [6], we split the dataset into training, validate and test sets with the size of 529/30/40.

Baselines. We denote the proposed method as MANN, and the argument knowledge integrated version as MANN-Aug. To validate the effectiveness, we compare our models to three kinds of mainstream models to demonstrate the advancement of our model. **CrossEvent**: a Max-Entropy model adopting document-level information [6]. **Combined-PSL**: a probabilistic soft logic model aiming to

exploit global information, the best reported feature-based system [10]. **Skip-CNN**: a CNN model with non-continue n-grams as input [13]. **DLRNN**: a LSTM-based model extracting cross-sentence clues to improve sentence-level event detection [2]. **ANN-Aug**: a attention model with additional arguments information [9]. **GMLATT**: a gated multilingual attention approach. It is the best reported sentence-level attention approaches [7].

Training Details. The dim of input word embedding, word attention and output word attention are 200/300/400, and those for entity are all 100. The number of attention head P is 5 for MANN model and 10 for MANN-Aug, and the context window size L is set to 60. The training is conducted on a TitanX GPU with learning rate as 1e-6 and batch size as 100, the training time of each epoch is 80 milliseconds in average and the optimal epoch number is around 100.

4.2 Overall Performance

Compared with feature-based methods, representation-based approaches perform better and attention-based model surpasses the representation-based methods and achieve the best F score in recent years. This is caused by the powerful representation ability of the attention mechanism.

Among the attention-based models, our approaches achieve the best F value. Besides, comparing with the model without external knowledge, incorporating the argument knowledge can improve the F score by 0.3%. Compared with the existing attention models like ANN, GMLATT, our model significantly improves the recall($\geq$ 4%), which proves that our model can handle the mono-attention problem effectively.

Methods	P	R	F1
CrossEvent	68.7	68.9	68.8
Combined-PSL	75.3	64.4	69.4
DLRNN	77.2	64.9	70.5
Skip-CNN	n/a	n/a	71.3
ANN-Aug	78.0	66.3	71.7
GMLATT	78.9	66.9	72.4
MANN(our)	73.2	**72.3**	72.7
MANN-Aug (our)	76.3	69.8	**73.0**

Table 1. Overall Comparison Results

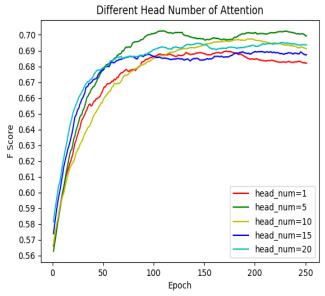

Fig. 5. The number of attention heads

4.3 The Effectiveness of Multi-head Attention

In this section, we discuss the effectiveness of multi-head attention in terms of 1) the length of sentence and 2) the number of attention heads.

The Length of Sentence. We first analyze how different sentence lengths influence the performance. We define short sentences as sentences of less than 10 words and long sentences as sentences of more than 10 words. Table 2 presents the results of our proposed MANN and a mono-attention model ANN.

Methods		Precision	Recall	F1
ANN	short	100.0	66.7	80.0
	long	71.5	67.3	69.3
MANN	short	80.0	80.0	80.0
	long	73.2	72.3	72.8

Table 2. Results with different sentence lengths

As we can see, both of the models perform better on short sentences. The reason is that the semantics of the short sentences are more uniform, reducing the ambiguity of the trigger words. Comparing with the short sentence, our model significantly improve the F1-value of long sentences(80%-80% v.s 69%-73%), which proves that the multi-head attention mechanism can better capture the multi-level and multi-angle information in long sentences.

The Number of Attention Heads. In order to illustrate the effect of the number of attention heads on the experimental results, we plot the training process of MANN with different heads in Fig. 5. For a more rigorous comparison, we use the MANN model without external knowledge. Experiment results show that the number of attention heads greatly influence the performance. As the number of attention heads increases, the optimal F1-value is gradually improved and the model can converge faster. However, it is worth noting that the model is more likely to encounter the over-fitting problem with a larger head number.

4.4 The Effectiveness of Argument Knowledge

We demonstrate the effectiveness of argument knowledge by comparing the performance of argument-enhanced MANN with basic MANN (MANN without argument knowledge). In order to ensure the rigor of the experiment, we set the head number as 10 and guarantee other hyper parameters are completely identical. Table 3 presents the comparison results.

Model	Precision	Recall	F1
MANN	74.3	69.0	71.6
MANN-Aug	**76.3**	**69.8**	**73.0**
MANN-Aug(I1)	69.8	**72.2**	71.0
MANN-Aug(I2)	**76.3**	69.8	**73.0**

Table 3. MANN with/without argument knowledge

As we can see from the Table 3, argument-enhanced MANN significantly outperform the basic MANN(71.6 v.s 73.0), which demonstrate the attention guide coming from argument knowledge is quite helpful. We further analyze the performance of argument-enhanced MANN by validating the effectiveness of I1/I2. Reviewing section 3.5, I1/I2 are two approaches to integrate multi-head attention. As shown in Table 3, I2 consistently outperforms I1 (73.0 v.s 71.0). This indicates that argument information is just an aspect of attention that MANN takes into account when making decisions. If we sum all the attention and supervise them with arguments information (what I1 exactly does), MANN will lose the ability to discover other useful information.

5 Related Work

Early **Feature-based method** designs various features [6] by domain experts, and later researches exploit the maximum entropy classifier or SVM [15] to automatically distinguish important features. **Representation-based methods** have yielded impressive results in event detection. DMCNN [1] is the pioneering work that employs convolution neural network (CNN) in this task, followed by various CNN [12, 13] and LSTM models [11, 3]. However, these methods still have the problem of unexplainable and high computational costs. **Attention mechanism** is a widely-used technique, which has been successfully applied to machine translation [16] and text classification [14]. In the field of event detection, [9] and [7] leverage semantic and multilingual attention mechanism to address the trigger ambiguity issue. However, these methods still suffer from mono-attention and knowledge absence problem.

6 Conclusion

We propose a multi-head attention neural network (MANN-Aug) incorporating argument knowledge to address the mono-attention and knowledge absence problem. Extensive experiments on ACE2005 dataset show that our method significant outperforms existing attention-based methods.

7 Acknowledgements

This work is supported by the National Key Research and Development Program of China (2018YFB1005100 and 2018YFB1005101) and NSFC key projects (U1736204, 61533018, 61661146007), Ministry of Education and China Mobile Joint Fund (MCM20170301), a research fund supported by Alibaba Group, and THUNUS NExT Co-Lab. It also got partial support from National Engineering Laboratory for Cyberlearning and Intelligent Technology, and Beijing Key Lab of Networked Multimedia.

References

1. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multi-pooling convolutional neural networks. In: The Meeting of the Association for Computational Linguistics. pp. 167–176 (2015)
2. Duan, S., He, R., Zhao, W.: Exploiting document level information to improve event detection via recurrent neural networks. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing. pp. 352–361 (2017)
3. Feng, X., Qin, B., Liu, T.: A language-independent neural network for event detection. Science China Information Sciences **61**(9), 092106 (2018)
4. Hagan, M.T., Demuth, H.B., Beale, M.H., De Jesús, O.: Neural network design, vol. 20. Pws Pub. Boston (1996)
5. Li, Q., Ji, H., Huang, L.: Joint event extraction via structured prediction with global features. In: Meeting of the Association for Computational Linguistics. pp. 73–82 (2013)
6. Liao, S., Grishman, R.: Using document level cross-event inference to improve event extraction. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. pp. 789–797 (2010)
7. Liu, J., Chen, Y., Liu, K., Zhao, J.: Event detection via gated multilingual attention mechanism. Statistics **1000**, 1250 (2018)
8. Liu, S., Chen, Y., He, S., Liu, K., Zhao, J.: Leveraging framenet to improve automatic event detection. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. pp. 2134–2143 (2016)
9. Liu, S., Chen, Y., Liu, K., Zhao, J.: Exploiting argument information to improve event detection via supervised attention mechanisms. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. pp. 1789–1798 (2017)
10. Liu, S., Liu, K., He, S., Zhao, J.: A probabilistic soft logic based approach to exploiting latent and global information in event classification. In: AAAI. pp. 2993–2999 (2016)
11. Nguyen, T.H., Cho, K., Grishman, R.: 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. pp. 300–309 (2016)
12. Nguyen, T.H., Grishman, R.: 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. pp. 365–371 (2015)
13. Nguyen, T.H., Grishman, R.: Modeling skip-grams for event detection with convolutional neural networks. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 886–891 (2016)
14. Pappas, N., Popescu-Belis, A.: Multilingual hierarchical attention networks for document classification. CoRR (2017), http://arxiv.org/abs/1707.00896
15. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9**(3), 293–300 (1999)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems 30. pp. 5998–6008 (2017)