# Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks

**Conference Paper** · July 2015

DOI: 10.3115/v1/P15-1017



**5 authors**, including:

Yubo Chen
Chinese Academy of Sciences
**15** PUBLICATIONS **260** CITATIONS
SEE PROFILE

Liheng Xu
Chinese Academy of Sciences
**10** PUBLICATIONS **329** CITATIONS
SEE PROFILE

Kang Liu
Chinese Academy of Sciences
**67** PUBLICATIONS **1,091** CITATIONS
SEE PROFILE

# Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks

**Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng and Jun Zhao**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{yubo.chen,lhxu,kliu,djzeng,jzhao}@nlpr.ia.ac.cn

## Abstract

Traditional approaches to the task of ACE event extraction primarily rely on elaborately designed features and complicated natural language processing (NLP) tools. These traditional approaches lack generalization, take a large amount of human effort and are prone to error propagation and data sparsity problems. This paper proposes a novel event-extraction method, which aims to automatically extract lexical-level and sentence-level features without using complicated NLP tools. We introduce a word-representation model to capture meaningful semantic regularities for words and adopt a framework based on a convolutional neural network (CNN) to capture sentence-level clues. However, CNN can only capture the most important information in a sentence and may miss valuable facts when considering multiple-event sentences. We propose a dynamic multi-pooling convolutional neural network (DMCNN), which uses a dynamic multi-pooling layer according to event triggers and arguments, to reserve more crucial information. The experimental results show that our approach significantly outperforms other state-of-the-art methods.

## 1 Introduction

Event extraction is an important and challenging task in Information Extraction (IE), which aims to discover event triggers with specific types and their arguments. Current state-of-the-art methods (Li et al., 2014; Li et al., 2013; Hong et al., 2011; Liao and Grishman, 2010; Ji and Grishman, 2008) often use a set of elaborately designed features that are extracted by textual analysis and linguistic knowledge. In general, we can divide the features into two categories: lexical features and contextual features.

Lexical features contain part-of-speech tags (POS), entity information, and morphology features (e.g., token, lemma, etc.), which aim to capture semantics or the background knowledge of words. For example, consider the following sentence with an ambiguous word *beats*:

**S1:** *Obama* ***beats*** *McCain.*

**S2:** *Tyson* ***beats*** *his opponent .*

In S1, *beats* is a trigger of type *Elect*. However, in S2, *beats* is a trigger of type *Attack*, which is more common than type *Elect*. Because of the ambiguity, a traditional approach may mislabel *beats* in S1 as a trigger of *Attack*. However, if we have the priori knowledge that *Obama* and *McCain* are presidential contenders, we have ample evidence to predict that *beats* is a trigger of type *Elect*. We call these knowledge **lexical-level clues**. To represent such features, the existing methods (Hong et al., 2011) often rely on human ingenuity, which is a time-consuming process and lacks generalization. Furthermore, traditional lexical features in previous methods are a one-hot representation, which may suffer from the data sparsity problem and may not be able to adequately capture the semantics of the words (Turian et al., 2010).

To identify events and arguments more precisely, previous methods often captured contextual features, such as syntactic features, which aim to understand how facts are tied together from a larger field of view. For example, in S3, there are two events that share three arguments as shown in Figure 1. From the dependency relation of *nsubj* between the argument *cameraman* and trigger *died*, we can induce a *Victim* role to *cameraman* in the *Die* event. We call such information **sentence-level clues**. However, the argument word *cameraman* and its trigger word *fired* are in different clauses, and there is no direct de-



*Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 167–176, Beijing, China, July 26-31, 2015. 

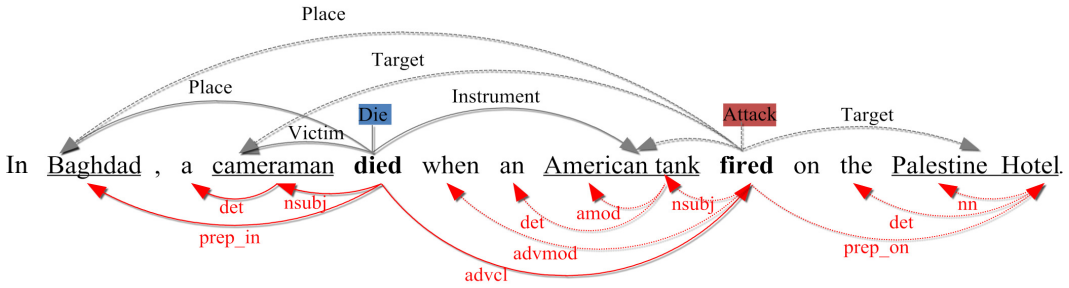


Figure 1: Event mentions and syntactic parser results of S3. The upper side shows two event mentions that share three arguments: the *Die* event mention, triggered by "died", and the *Attack* event mention, triggered by "fired". The lower side shows the collapsed dependency results.

pendency path between them. Thus it is difficult to find the *Target* role between them using traditional dependency features. In addition, extracting such features depends heavily on the performance of pre-existing NLP systems, which could suffer from error propagation.

**S3:** *In Baghdad, a cameraman **died** when an American tank **fired** on the Palestine Hotel.*

To correctly attach *cameraman* to *fired* as a *Target* argument, we must exploit internal semantics over the entire sentence such that the *Attack* event results in *Die* event. Recent improvements of convolutional neural networks (CNNs) have been proven to be efficient for capturing syntactic and semantics between words within a sentence (Collobert et al., 2011; Kalchbrenner and Blunsom, 2013; Zeng et al., 2014) for NLP tasks. CNNs typically use a max-pooling layer, which applies a max operation over the representation of an entire sentence to capture the most useful information. However, in event extraction, one sentence may contain two or more events, and these events may share the argument with different roles. For example, there are two events in S3, namely, the *Die* event and *Attack* event. If we use a traditional max-pooling layer and only keep the most important information to represent the sentence, we may obtain the information that depicts "a cameraman died" but miss the information about "American tank fired on the Palestine Hotel", which is important for predicting the *Attack* event and valuable for attaching *cameraman* to *fired* as an *Target* argument. In our experiments, we found that such multiple-event sentences comprise 27.3% of our dataset, which is a phenomenon we cannot ignore.

In this paper, we propose a dynamic multi-pooling convolutional neural network (**DMCNN**) to address the problems stated above. To capture lexical-level clues and reduce human effort, we introduce a word-representation model (Mikolov et al., 2013b), which has been shown to be able to capture the meaningful semantic regularities of words (Bengio et al., 2003; Erhan et al., 2010; Mikolov et al., 2013a). To capture sentence-level clues without using complicated NLP tools, and to reserve information more comprehensively, we devise a dynamic multi-pooling layer for CNN, which returns the maximum value in each part of the sentence according to event triggers and arguments. In summary, the contributions of this paper are as follows:

- We present a novel framework for event extraction, which can automatically induce lexical-level and sentence-level features from plain texts without complicated NLP preprocessing.
- We devise a dynamic multi-pooling convolutional neural network (DMCNN), which aims to capture more valuable information within a sentence for event extraction.
- We conduct experiments on a widely used ACE2005 event extraction dataset, and the experimental results show that our approach outperforms other state-of-the-art methods.

## 2 Event Extraction Task

In this paper, we focus on the event extraction task defined in Automatic Content Extraction[1] (ACE) evaluation, where an event is defined as a specific occurrence involving participants. First, we introduce some ACE terminology to understand this task more easily:

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

- **Event mention**: a phrase or sentence within which an event is described, including a trigger and arguments.
- **Event trigger**: the main word that most clearly expresses the occurrence of an event (An ACE event trigger is typically a verb or a noun).
- **Event argument**: an entity mention, temporal expression or value (e.g. Job-Title) that is involved in an event (viz., participants).
- **Argument role**: the relationship between an argument to the event in which it participates.

Given an English text document, an event extraction system should predict event triggers with specific subtypes and their arguments for each sentence. The upper side of figure 1 depicts the event triggers and their arguments for S3 in Section 1. ACE defines 8 event types and 33 subtypes, such as *Attack* or *Elect*.

Although event extraction depends on name identification and entity mention co-reference, it is another difficult task in ACE evaluation and not the focus in the event extraction task. Thus, in this paper, we directly leverage the entity label provided by the ACE, following most previous works (Hong et al., 2011; Liao and Grishman, 2010; Ji and Grishman, 2008).

## 3 Methodology

In this paper, event extraction is formulated as a two-stage, multi-class classification via dynamic multi-pooling convolutional neural networks with the automatically learned features. The first stage is called *trigger classification*, in which we use a DMCNN to classify each word in a sentence to identify trigger words. If one sentence has triggers, the second stage is conducted, which applies a similar DMCNN to assign arguments to triggers and align the roles of the arguments. We call this *argument classification*. Because the second stage is more complicated, we first describe the methodology of argument classification in Section 3.1∼3.4 and then illustrate the difference between the DMCNNs that are used for trigger classification and those used for argument classification in Section 3.5.

Figure 2 describes the architecture of argument classification, which primarily involves the following four components: (i) word-embedding learning, which reveals the embedding vectors of words in an unsupervised manner; (ii) lexical-level feature representation, which directly uses embedding vectors of words to capture lexical clues; (iii) sentence-level feature extraction, which proposes a DMCNN to learn the compositional semantic features of sentences; and (iv) argument classifier output, which calculates a confidence score for each argument role candidate.

### 3.1 Word Embedding Learning and Lexical-Level Feature Representation

Lexical-level features serve as important clues for event extraction (Hong et al., 2011; Li et al., 2013). Traditional lexical-level features primarily include *lemma, synonyms* and *POS tag of the candidate words*. The quality of such features depends strongly on the results of existing NLP tools and human ingenuity. Additionally, the traditional features remain unsatisfactory for capturing the semantics of words, which are important in event extraction, as showed in S1 and S2. As Erhan et al. (2010) reported, word embeddings learned from a significant amount of unlabeled data are more powerful for capturing the meaningful semantic regularities of words. This paper uses unsupervised pre-trained word embedding as the source of base features. We select the word embeddings of candidate words (candidate trigger, candidate argument) and the context tokens (left and right tokens of the candidate words). Then, all of these word embeddings are concatenated into the lexical-level features vector $L$ to represent the lexical-level features in argument classification.

In this work, we use the Skip-gram model to pre-train the word embedding. This model is the state-of-the-art model in many NLP tasks (Baroni et al., 2014). The Skip-gram model trains the embeddings of words $w_1, w_2...w_m$ by maximizing the average log probability,

$$\frac{1}{m}\sum_{t=1}^{m}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t) \qquad (1)$$

where c is the size of the training window. Basically, $p(w_{t+j}|w_t)$ is defined as,

$$p(w_{t+j}|w_t) = \frac{\exp(e'^{T}_{t+j}e_t)}{\sum_{w=1}^{m}\exp(e'^{T}_{w}e_t)} \qquad (2)$$

where $m$ is the vocabulary of the unlabeled text. $e'_i$ is another embedding for $e_i$, see Morin and Bengio (2005) for details.

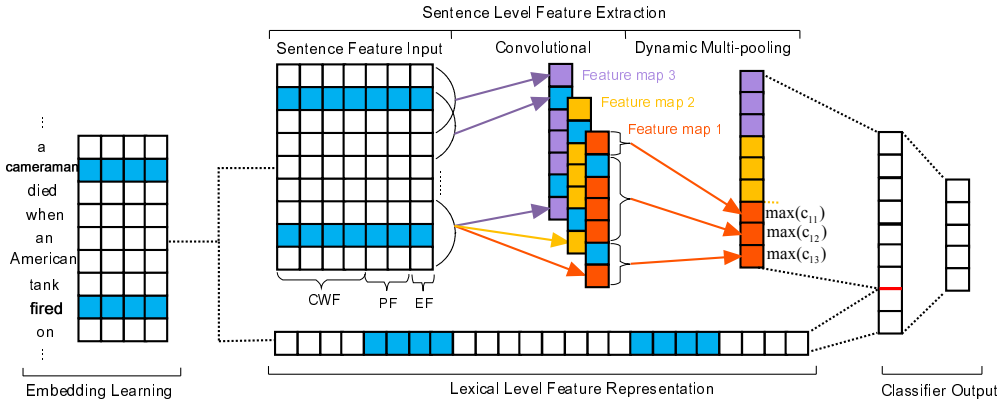


Figure 2: The architecture for the stage of argument classification in the event extraction. It illustrates the processing of one instance with the predict trigger *fired* and the candidate argument *cameraman*.

### 3.2 Extracting Sentence-Level Features Using a DMCNN

The CNN, with max-pooling layers, is a good choice to capture the semantics of long-distance words within a sentence (Collobert et al., 2011). However, as noted in the section 1, traditional CNN is incapable of addressing the event extraction problem. Because a sentence may contain more than one event, using only the most important information to represent a sentence, as in the traditional CNN, will miss valuable clues. To resolve this problem, we propose a DMCNN to extract the sentence-level features. The DMCNN uses a dynamic multi-pooling layer to obtain a maximum value for each part of a sentence, which is split by event triggers and event arguments. Thus, the DMCNN is expected to capture more valuable clues compared to traditional CNN methods.

#### 3.2.1 Input

This subsection illustrates the input needed for a DMCNN to extract sentence-level features. The semantic interactions between the predicted trigger words and argument candidates are crucial for argument classification. Therefore, we propose three types of input that the DMCNN uses to capture these important clues:

- Context-word feature (CWF): Similar to Kalchbrenner et al. (2014) and Collobert et al. (2011), we take all the words of the whole sentence as the context. CWF is the vector of each word token transformed by looking up word embeddings.

- Position feature (PF): It is necessary to specify which words are the predicted trigger or candidate argument in the argument classification. Thus, we proposed the PF, which is defined as the relative distance of the current word to the predicted trigger or candidate argument. For example, in S3, the relative distances of *tank* to the candidate argument *cameraman* is 5. To encode the position feature, each distance value is also represented by an embedding vector. Similar to word embedding, Distance Values are randomly initialized and optimized through back propagation.

- Event-type feature (EF): The event type of a current trigger is valuable for argument classification (Ahn, 2006; Hong et al., 2011; Liao and Grishman, 2010; Li et al., 2013), so we encode event type predicted in the trigger classification stage as an important clue for the DMCNN, as in the PF.

Figure 2 assumes that word embedding has size $d_w = 4$, position embedding has size $d_p = 1$ and event-type embedding has size $d_e = 1$. Let $x_i \in \mathbb{R}^d$ be the $d$-dimensional vector representation corresponding to the $i$-th word in the sentence, where $d = d_w + d_p * 2 + d_e$. A sentence of length $n$ is represented as follows:

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n \quad (3)$$

where $\oplus$ is the concatenation operator. Thus, combined word embedding, position embedding and event-type embedding transform an instance as a matrix $X \in \mathbb{R}^{n \times d}$. Then, $X$ is fed into the convolution part.

#### 3.2.2 Convolution

The convolution layer aims to capture the compositional semantics of a entire sentence and compress these valuable semantics into feature maps. In general, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, ..., x_{i+j}$. A convolution operation involves a filter $w \in \mathbb{R}^{h \times d}$, which is applied to a window of $h$ words to produce a new feature. For example, a feature $c_i$ is generated from a window of words $x_{i:i+h-1}$ by the following operator,

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{4}$$

where $b \in R$ is a bias term and $f$ is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}$ to produce a feature map $c_i$ where the index $i$ ranges from 1 to $n - h + 1$.

We have described the process of how one feature map is extracted from one filter. To capture different features, it usually use multiple filters in the convolution. Assuming that we use m filters $W = w_1, w_2, ..., w_m$, the convolution operation can be expressed as:

$$c_{ji} = f(w_j \cdot x_{i:i+h-1} + b_j) \tag{5}$$

where j ranges from 1 to $m$. The convolution result is a matrix $C \in \mathbb{R}^{m \times (n-h+1)}$.

#### 3.2.3 Dynamic Multi-Pooling

To extract the most important features (max value) within each feature map, traditional CNNs (Collobert et al., 2011; Kim, 2014; Zeng et al., 2014) take one feature map as a pool and only get one max value for each feature map. However, single max-pooling is not sufficient for event extraction. Because in the task of this paper, one sentence may contain two or more events, and one argument candidate may play a different role with a different trigger. To make an accurate prediction, it is necessary to capture the most valuable information with regard to the change of the candidate words. Thus, we split each feature map into three parts according to the candidate argument and predicted trigger in the argument classification stage. Instead of using one max value for an entire feature map to represent the sentence, we keep the max value of each split part and call it dynamic multi-pooling. Compared to traditional max-pooling, dynamic multi-pooling can reserve more valuable information without missing the max-pooling value.

As shown in Figure 2, the feature map output $c_j$ is divided into three sections $c_{j1}, c_{j2}, c_{j3}$ by "cameraman" and "fired". The dynamic multi-pooling can be expressed as formula 6,where $1 \leq j \leq m$ and $1 \leq i \leq 3$.

$$p_{ji} = \max(c_{ji}) \tag{6}$$

Through the dynamic multi-pooling layer, we obtain the $p_{ji}$ for each feature map. Then, we concatenate all $p_{ji}$ to form a vector $P \in \mathbb{R}^{3m}$, which can be considered as higher-level features (sentence-level features).

### 3.3 Output

The automatically learned lexical and sentence-level features mentioned above are concatenated into a single vector $F = [L, P]$. To compute the confidence of each argument role, the feature vector $F \in \mathbb{R}^{3m+d_l}$, where $m$ is the number of the feature map and $d_l$ is the dimension of the lexical-level features, is fed into a classifier.

$$O = W_s F + b_s \tag{7}$$

$W_s \in \mathbb{R}^{n_1 \times (3m+d_l)}$ is the transformation matrix and $O \in \mathbb{R}^{n_1}$ is the final output of the network, where $n_1$ is equal to the number of the argument role including the "None role" label for the candidate argument which don't play any role in the event. For regularization, we also employ dropout(Hinton et al., 2012) on the penultimate layer, which can prevent the co-adaptation of hidden units by randomly dropping out a proportion $p$ of the hidden units during forward and backpropagation.

### 3.4 Training

We define all of the parameters for the stage of argument classification to be trained as $\theta = (E, PF_1, PF_2, EF, W, b, W_S, b_s)$. Specifically, E is the word embedding, $PF_1$ and $PF_2$ are the position embedding, $EF$ is the embedding of the event type, $W$ and $b$ are the parameter of the filter, $W_s$ and $b_s$ are all of the parameters of the output layer.

Given an input example s, the network with parameter $\theta$ outputs the vector $O$, where the $i$-th component $O_i$ contains the score for argument role $i$. To obtain the conditional probability $p(i|x, \theta)$, we apply a softmax operation over all argument

role types:

$$p(i|x,\theta) = \frac{e^{o_i}}{\sum_{k=1}^{n_1} e^{o_k}} \quad (8)$$

Given all of our (suppose T) training examples $(x_i; y_i)$, we can then define the objective function as follows:

$$J(\theta) = \sum_{i=1}^{T} \log p(y^{(i)}|x^{(i)}, \theta) \quad (9)$$

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

### 3.5 Model for Trigger Classification

In the above sections, we presented our model and features for argument classification. The method proposed above is also suitable for trigger classification, but the task only need to find triggers in the sentence, which is less complicated than argument classification. Thus we can used a simplified version of DMCNN.

In the trigger classification, we only use the candidate trigger and its left and right tokens in the lexical-level feature representation. In the feature representation of the sentence level, we use the same CWF as does in argument classification, but we only use the position of the candidate trigger to embed the position feature. Furthermore, instead of splitting the sentence into three parts, the sentence is split into two parts by a candidate trigger. Except for the above change in the features and model, we classify a trigger as the classification of an argument. Both stages form the framework of the event extraction.

## 4 Experiments

### 4.1 Dataset and Evaluation Metric

We utilized the ACE 2005 corpus as our dataset. For comparison, as the same as Li et al. (2013), Hong et al. (2011) and Liao and Grishman (2010), we used the same test set with 40 newswire articles and the same development set with 30 other documents randomly selected from different genres and the rest 529 documents are used for training. Similar to previous work (Li et al., 2013; Hong et al., 2011; Liao and Grishman, 2010; Ji and Grishman, 2008), we use the following criteria to judge the correctness of each predicted event mention:

- A trigger is correct if its event subtype and offsets match those of a reference trigger.
- An argument is correctly identified if its event subtype and offsets match those of any of the reference argument mentions.
- An argument is correctly classified if its event subtype, offsets and argument role match those of any of the reference argument mentions.

Finally we use *Precision (P)*, *Recall (R)* and *F measure ($F_1$)* as the evaluation metrics.

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

We select the following state-of-the-art methods for comparison.
1) **Li's baseline** is the feature-based system proposed by Li et al. (2013), which only employs human-designed lexical features, basic features and syntactic features.
2) **Liao's cross-event** is the method proposed by Liao and Grishman (2010), which uses document-level information to improve the performance of ACE event extraction.
3) **Hong's cross-entity** is the method proposed by Hong et al. (2011), which extracts event by using cross-entity inference. To the best of our knowledge, it is the best-reported feature-based system in the literature based on gold standards argument candidates.
4) **Li's structure** is the method proposed by Li et al. (2013), which extracts events based on structure prediction. It is the best-reported structure-based system.

Following Li et al. (2013), we tuned the model parameters on the development through grid search. Moreover, in different stages of event extraction, we adopted different parameters in the DMCNN. Specifically, in the trigger classification, we set the window size as 3, the number of the feature map as 200, the batch size as 170 and the dimension of the PF as 5. In the argument classification, we set the window size as 3, the number of the feature map as 300, the batch size as 20 and the dimension of the PF and EF as 5. Stochastic gradient descent over shuffled mini-batches with the Adadelta update rule(Zeiler, 2012) is used for training and testing processes. It mainly contains two parameters $p$ and $\varepsilon$. We set $p = 0.95$ and $\varepsilon = 1e^{-6}$. For the dropout operation, we set the

| Methods | Trigger Identification(%) | | | Trigger Identification + Classification(%) | | | Argument Identification(%) | | | Argument Role(%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F | P | R | F |
| Li's baseline | 76.2 | 60.5 | 67.4 | 74.5 | 59.1 | 65.9 | 74.1 | 37.4 | 49.7 | 65.4 | 33.1 | 43.9 |
| Liao's cross-event | N/A | | | 68.7 | 68.9 | 68.8 | 50.9 | 49.7 | 50.3 | 45.1 | 44.1 | 44.6 |
| Hong's cross-entity | N/A | | | 72.9 | 64.3 | 68.3 | 53.4 | 52.9 | 53.1 | 51.6 | 45.5 | 48.3 |
| Li's structure | 76.9 | 65.0 | 70.4 | 73.7 | 62.3 | 67.5 | 69.8 | 47.9 | 56.8 | 64.7 | 44.4 | 52.7 |
| **DMCNN model** | 80.4 | 67.7 | **73.5** | 75.6 | 63.6 | **69.1** | 68.8 | 51.9 | **59.1** | 62.2 | 46.9 | **53.5** |

Table 1: Overall performance on blind test data

$rate = 0.5$. We train the word embedding using the Skip-gram algorithm [2] on the NYT corpus [3].

Table 1 shows the overall performance on the blind test dataset. From the results, we can see that the DMCNN model we proposed with the automatically learned features achieves the best performance among all of the compared methods. DMCNN can improve the best $F_1$ (Li et al., 2013) in the state-of-the-arts for trigger classification by $1.6\%$ and argument role classification by $0.8\%$. This demonstrates the effectiveness of the proposed method. Moreover, a comparison of *Liao's cross-event* with *Li's baseline* illustrates that *Liao's cross-event* achieves a better performance. We can also make the same observation when comparing *Hong's cross-entity* with *Liao's cross-event* and comparing *Li's structure* with *Hong's cross-entity*. It proves that richer feature sets lead to better performance when using traditional human-designed features. However, our method could obtain further better results on the condition of only using automatically learned features from original words. Specifically, compared to *Hong's cross-entity*, it gains $0.8\%$ improvement on trigger classification $F_1$ and $5.2\%$ improvement on argument classification $F_1$. We believe the reason is that the features we automatically learned can capture more meaningful semantic regularities of words. Remarkably, compared to *Li's structure*, our approach with sentence and lexical features achieves comparable performance even though we do not use complicated NLP tools.

### 4.3 Effect of The DMCNN on Extracting Sentence-Level Features

In this subsection, we prove the effectiveness of the proposed DMCNN for sentence-level feature extraction. We specifically select two methods as baselines for comparison with our DMCNN: Embeddings+T and CNN. Embeddings+T uses word embeddings as lexical-level features and traditional sentence-level features based on human design (Li et al., 2013). A CNN is similar to a DMCNN, except that it uses a standard convolutional neural network with max-pooling to capture sentence-level features. By contrast, a DMCNN uses the dynamic multi-pooling layer in the network instead of the max-pooling layer in a CNN. Moreover, to prove that a DMCNN could capture more precise sentence-level features, especially for those sentences with multiple events, we divide the testing data into two parts according the event number in a sentence (single event and multiple events) and perform evaluations separately. Table 2 shows the proportion of sentences with multiple events or a single event and the proportion of arguments that attend one event or more events within one sentence in our dataset. Table 3 shows the results.

| **Stage** | **1/1 (%)** | **1/N (%)** |
|---|---|---|
| Trigger | 72.7 | **27.3** |
| Argument | 76.8 | **23.2** |

Table 2: The proportion of multiple events within one sentence. 1/1 means that one sentence only has one trigger or one argument plays a role in one sentence; otherwise, 1/N is used.

Table 3 illustrates that the methods based on convolutional neural networks (CNN and DMCNN) outperform Embeddings+T. It proves that convolutional neural networks could be more effective than traditional human-design strategies for sentence-level feature extraction. In table 3, for all sentences, our method achieves improvements of approximately $2.8\%$ and $4.6\%$ over the CNN. The results prove the effectiveness of the dynamic multi-pooling layer. Interestingly, the DMCNN yields a $7.8\%$ improvement for trigger classification on the sentences with multiple events. This improvement is larger than in sentences with a single event. Similar observations can be made for

[2] https://code.google.com/p/word2vec/

[3] https://catalog.ldc.upenn.edu/LDC2008T19

the argument classification results. This demonstrates that the proposed DMCNN can effectively capture more valuable clues than the CNN with max-pooling, especially when one sentence contains more than one event.

| Stage | Method | 1/1 | 1/N | all |
|---|---|---|---|---|
| | | $F_1$ | $F_1$ | $F_1$ |
| Trigger | Embedding+T | 68.1 | 25.5 | 59.8 |
| | CNN | 72.5 | 43.1 | 66.3 |
| | DMCNN | **74.3** | **50.9** | **69.1** |
| Argument | Embedding+T | 37.4 | 15.5 | 32.6 |
| | CNN | 51.6 | 36.6 | 48.9 |
| | DMCNN | **54.6** | **48.7** | **53.5** |

Table 3: Comparison of the event extraction scores obtained for the Traditional, CNN and DMCNN models

### 4.4 Effect of Word Embedding on Extracting Lexical-Level Features

This subsection studies the effectiveness of our word embedding for lexical features. For comparison purposes, we select the baseline described by Li et al. (2013) as the traditional method, which uses traditional lexical features, such as n-grams, POS tags and some entity information. In contrast, we only use word embedding as our lexical feature. Moreover, to prove that word embedding could capture more valuable semantics, especially for those words in the test data that never appear to be the same event type or argument role in the training data, we divide the triggers and arguments in the testing data into two parts (1: appearing in testing data only, or 2: appearing in both testing and training data with the same event type or argument role) and perform evaluations separately. For triggers, 34.9% of the trigger words in the test data never appear to be the same event type in the training data. This proportion is 83.1% for arguments. The experimental results are shown in Table 4.

Table 4 illustrates that for all situations, our method makes significant improvements compared with the traditional lexical features in the classification of both the trigger and argument. For situation B, the lexical-level features extracted from word embedding yield a $18.8\%$ improvement for trigger classification and an $8.5\%$ improvement for argument classification. This occurs because the baseline only uses discrete features, so they suffer from data sparsity and could not adequately handle a situation in which a trigger or argument does not appear in the training data.

| Stage | Method | A | B | All |
|---|---|---|---|---|
| | | $F_1$ | $F_1$ | $F_1$ |
| Trigger | Traditional | 68.8 | 14.3 | 61.2 |
| | Ours | **70.7** | **33.1** | **64.9** |
| Argument | Traditional | 58.5 | 22.2 | 34.6 |
| | Ours | **59.5** | **30.7** | **40.2** |

Table 4: Comparison of the results for the traditional lexical feature and our lexical feature. A denotes the triggers or arguments appearing in both training and test datasets, and B indicates all other cases.

### 4.5 Lexical features vs. Sentence Features

To compare the effectiveness of different levels of features, we extract events by using lexical features and sentence features separately. The results obtained using the DMCNN are shown in table 5. Interestingly, in the trigger-classification stage, the lexical features play an effective role, whereas the sentence features play a more important role in the argument-classification stage. The best results are achieved when we combine lexical-level and sentence-level features. This observation demonstrates that both of the two-level features are important for event extraction.

| Feature | Trigger | Argument |
|---|---|---|
| | $F_1$ | $F_1$ |
| Lexical | **64.9** | 40.2 |
| Sentence | 63.8 | **50.7** |
| Combine | **69.1** | **53.5** |

Table 5: Comparison of the trigger-classification score and argument-classification score obtained by lexical-level features, sentence-level features and a combination of both

## 5 Related Work

Event extraction is one of important topics in NLP. Many approaches have been explored for event extraction. Nearly all of the ACE event extraction use supervised paradigm. We further divide supervised approaches into feature-based methods and structure-based methods.

In feature-based methods, a diverse set of strategies has been exploited to convert classification clues (such as sequences and parse trees) into feature vectors. Ahn (2006) uses the lexical features(e.g., full word, pos tag), syntactic features (e.g., dependency features) and external-knowledge features(WordNet) to extract the event. Inspired by the hypothesis of "One Sense Per Dis-

course"(Yarowsky, 1995), Ji and Grishman (2008) combined global evidence from related documents with local decisions for the event extraction. To capture more clues from the texts, Gupta and Ji (2009), Liao and Grishman (2010) and Hong et al. (2011) proposed the cross-event and cross-entity inference for the ACE event task. Although these approaches achieve high performance, feature-based methods suffer from the problem of selecting a suitable feature set when converting the classification clues into feature vectors.

In structure-based methods, researchers treat event extraction as the task of predicting the structure of the event in a sentence. McClosky et al. (2011) casted the problem of biomedical event extraction as a dependency parsing problem. Li et al. (2013) presented a joint framework for ACE event extraction based on structured perceptron with beam search. To use more information from the sentence, Li et al. (2014) proposed to extract entity mentions, relations and events in ACE task based on the unified structure. These methods yield relatively high performance. However, the performance of these methods depend strongly on the quality of the designed features and endure the errors in the existing NLP tools.

## 6 Conclusion

This paper proposes a novel event extraction method, which can automatically extract lexical-level and sentence-level features from plain texts without complicated NLP preprocessing. A word-representation model is introduced to capture lexical semantic clues and a dynamic multi-pooling convolutional neural network (DMCNN) is devised to encode sentence semantic clues. The experimental results prove the effectiveness of the proposed method.

## Acknowledgments

This work was supported by the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332 and No. 61202329)

## References

David Ahn. 2006. The stages of event extraction. In *Proceedings of ACL*, pages 1–8.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Dont count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of ACL*, pages 238–247.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

Chen Chen and V Incent NG. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *Proceedings of COLING*, pages 529–544.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of ACL-IJCNLP*, pages 369–372.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

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 ACL-HLT*, pages 1127–1136.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL*, pages 254–262.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of AAAI*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*, pages 73–82.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of EMNLP*, pages 1846–1851.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*, pages 789–797.

David McClosky, Mihai Surdeanu, and Christopher D Manning. 2011. Event extraction as dependency parsing. In *Proceedings of ACL-HLT*, pages 1626–1635.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, pages 246–252.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL*, pages 189–196.

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

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.