Modeling Skip-Grams for Event Detection with Convolutional Neural Networks

Thien Huu Nguyen
Computer Science Department
New York University
New York, NY 10003 USA
thien@cs.nyu.edu

Ralph Grishman
Computer Science Department
New York University
New York, NY 10003 USA
grishman@cs.nyu.edu

Abstract

Convolutional neural networks (CNN) have achieved the top performance for event detection due to their capacity to induce the underlying structures of the k-grams in the sentences. However, the current CNN-based event detectors only model the consecutive k-grams and ignore the non-consecutive k-grams that might involve important structures for event detection. In this work, we propose to improve the current CNN models for ED by introducing the non-consecutive convolution. Our systematic evaluation on both the general setting and the domain adaptation setting demonstrates the effectiveness of the non-consecutive CNN model, leading to the significant performance improvement over the current state-of-the-art systems.

1 Introduction

The goal of event detection (ED) is to locate event triggers of some specified types in text. Triggers are generally single verbs or nominalizations that evoke the events of interest. This is an important and challenging task of information extraction in natural language processing (NLP), as the same event might appear in various expressions, and an expression might express different events depending on contexts.

The current state-of-the-art systems for ED have involved the application of convolutional neural networks (CNN) (Nguyen and Grishman, 2015b; Chen et al., 2015) that automatically learn effective feature representations for ED from sentences. This has overcome the two fundamental limitations of the traditional feature-based methods for ED: (i) the complicated feature engineering for rich feature sets and (ii) the error propagation from the NLP toolkits and resources (i.e, parsers, part of speech taggers etc) that generate such features.

The prior CNN models for ED are characterized by the temporal convolution operators that linearly map the vectors for the k-grams in the sentences into the feature space. Such k-gram vectors are obtained by concatenating the vectors of the k consecutive words in the sentences (Nguyen and Grishman, 2015b; Chen et al., 2015). In other words, the previous CNN models for ED only focus on modeling the consecutive k-grams. Unfortunately, such consecutive mechanism is unable to capture the long-range and non-consecutive dependencies that are necessary to the prediction of trigger words. For instance, consider the following sentence with the trigger word "*leave*" from the ACE 2005 corpus:

*The mystery is that she took the job in the first place or didn't **leave** earlier.*

The correct event type for the trigger word "*leave*" in this case is "*End-Org*". However, the previous CNN models might not be able to detect "*leave*" as an event trigger or incorrectly predict its type as "*Movement*". This is caused by their reliance on the consecutive local k-grams such as "*leave earlier*". Consequently, we need to resort to the non-consecutive pattern "*job leave*" to correctly determine the event type of "*leave*" in this case.

Guided by this intuition, we propose to improve the previous CNN models for ED by operating the convolution on all possible non-consecutive k-grams

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 886–891,
Austin, Texas, November 1-5, 2016.

in the sentences. We aggregate the resulting convolution scores via the *max-pooling* function to unveil the most important non-consecutive k-grams for ED. The aggregation over all the possible non-consecutive k-grams is made efficient with dynamic programming.

Note that our work is related to (Lei et al., 2015) who employ the non-consecutive convolution for the sentence and news classification problems. Our work is different from (Lei et al., 2015) in that we model the relative distances of words to the trigger candidates in the sentences via position embeddings, while (Lei et al., 2015) use the absolute distances between words in the k-grams to compute the decay weights for aggregation. To the best of our knowledge, this is the first work on non-consecutive CNN for ED.

We systematically evaluate the proposed model in the general setting as well as the domain adaptation setting. The experiment results demonstrate that our model significantly outperforms the current state-of-the-art models in such settings.

2 Model

We formalize ED as a multi-class classification problem. Given a sentence, for every token in that sentence, we want to predict if the current token is an event trigger of some event in the pre-defined event set or not? The current token along with its context in the sentence constitute an event trigger candidate.

In order to make it compatible with the previous work, we follow the procedure in (Nguyen and Grishman, 2015b) to process the trigger candidates for CNN. In particular, we limit the context of the trigger candidates to a fixed window size by trimming longer sentences and padding shorter sentences with a special token when necessary. Let $2n + 1$ be the fixed window size, and $W = [w_0, w_1, \ldots, w_n, \ldots, w_{2n-1}, w_{2n}]$ be some trigger candidate where the current token is positioned in the middle of the window (token w_n). Before entering CNN, each token w_i is first transformed into a real-valued vector x_i using the concatenation of the following vectors:

1. The word embedding vector of w_i: This is obtained by looking up a pre-trained word embedding table D (Turian et al., 2010; Mikolov et al., 2013a).

2. The position embedding vector of w_i: We obtain this vector by looking up the position embedding table for the relative distance $i - n$ from the token w_i to the current token w_n. The position embedding table is initialized randomly.

3. The real-valued embedding vector for the entity type of w_i: This vector is generated by looking up the entity type embedding table (initialized randomly) for the entity type of w_i. Note that we employ the BIO annotation schema to assign entity type labels to each token in the sentences using the entity mention heads as in (Nguyen and Grishman, 2015b).

The transformation from the token w_i to the vector x_i ($x_i \in \mathbb{R}^d$) essentially converts the input candidate W into a sequence of real-valued vectors $X = (x_0, x_1, \ldots, x_{2n})$. This sequence is used as input in the following CNN models.

2.1 The Traditional CNN

Given the window size k, the traditional CNN models for ED consider the following set of $2n + 1$ consecutive k-gram vectors:

$$C = \{u_i : 0 \leq i \leq 2n\} \quad (1)$$

Vector u_i is the concatenation of the k consecutive vectors preceding position i in the sequence X: $u_i = [x_{i-k+1}, x_{i-k+2}, \ldots, x_i] \in \mathbb{R}^{dk}$ where the out-of-index vectors are simply set to all zeros.

The core of the CNN models is the convolution operation, specified by the filter vector $\mathbf{f} \in \mathbb{R}^{dk}$. In CNN, $\mathbf{f}$ can be seen as a feature extractor for the k-grams that operates via the dot product with each element in C. This produces the following convolution score set: $S(C) = \{\mathbf{f}^T u_i : 0 \leq i \leq 2n\}$.

In the next step, we aggregate the features in S with the $\max$ function, resulting in the aggregation score:

$$p_k^{\mathbf{f}} = \max S(C) = \max\{s_i : 0 \leq i \leq 2n\} \quad (2)$$

Afterward, $p_k^{\mathbf{f}}$ is often transformed by a non-linear function G[1] to generate the transformed score $G(p_k^{\mathbf{f}})$, functioning as the extracted feature for the initial trigger candidate W.

[1] The `tanh` function in this work.

We can then repeat this process for different window sizes k and filters $\mathbf{f}$, generating multiple features $G(p_k^{\mathbf{f}})$ to capture various aspects of the trigger candidate W. Finally, such features are concatenated into a single representation vector for W, to be fed into a feed-forward neural network with a softmax layer in the end to perform classification.

2.2 The Non-consecutive CNN

As mentioned in the introduction, the limitation of the previous CNN models for ED is the inability to encode the non-consecutive k-grams that might be crucial to the trigger prediction. This limitation originates from Equation 1 in which only the consecutive k-gram vectors are considered. In order to overcome such limitation, we propose to model all possible non-consecutive k-grams in the trigger candidate, leading to the following set of non-consecutive k-gram vectors:

$$N = \{v_{i_1 i_2 \ldots i_k} : 0 \leq i_1 < i_2 < \ldots < i_k \leq 2n\}$$

where: $v_{i_1 i_2 \ldots i_k} = [x_{i_1}, x_{i_2}, \ldots, x_{i_k}] \in \mathbb{R}^{dk}$ and the number of elements in N is $|N| = \binom{2n+1}{k}$.

The non-consecutive CNN model then follows the procedure of the traditional CNN model in Section 2.1 to compute the representation vector for classification. The only difference is that the computation is done on the input set N instead of C. In particular, the convolution score set in this case would be $S(N) = \{\mathbf{f}^T v : v \in N\}$, while the aggregating score would be:

$$p_k^{\mathbf{f}} = \max S(N) = \max\{s : s \in S(N)\} \quad (3)$$

2.3 Implementation

Note that the maximum operation in Equation 2 only requires $O(n)$ operations while the naive implementation of Equation 3 would need $O(|N|) = O(n^k)$ operations. In this work, we employ the dynamic programming (DP) procedure below to reduce the computation time for Equation 3.

Assuming the filter vector $\mathbf{f}$ is the concatenation of the k vectors $\mathbf{f}_1, \ldots, \mathbf{f}_k \in \mathbb{R}^d$: $\mathbf{f} = [\mathbf{f}_1, \ldots, \mathbf{f}_k]$, Equation 3 can be re-written by:

$$\begin{aligned} p_k^{\mathbf{f}} = &\max\{\mathbf{f}_1^T x_{i_1} + \ldots + \mathbf{f}_k^T x_{i_k} \\ &: 0 \leq i_1 < i_2 < \ldots < i_k \leq 2n\} \end{aligned}$$

Let D_t^j be the dynamic programming table representing the maximum convolution score for the subfilter $[\mathbf{f}_1, \ldots, \mathbf{f}_j]$ over all possible non-consecutive j-gram vectors in the subsequence $(x_0, x_1, \ldots, x_t)$ of X:

$$\begin{aligned} D_t^j = &\max\{\mathbf{f}_1^T x_{i_1} + \ldots + \mathbf{f}_j^T x_{i_j} \\ &: 0 \leq i_1 < i_2 < \ldots < i_j \leq t\} \end{aligned}$$

where $1 \leq j \leq k$, $j - 1 \leq t \leq 2n$.

Note that $p_k^{\mathbf{f}} = D_{2n}^k$.

We can solve this DP problem by the following recursive formulas[2]:

$$D_t^j = \max\{D_{t-1}^j, D_{t-1}^{j-1} + \mathbf{f}_j^T x_t\}$$

The computation time for this procedure is $O(kn)$ and remains linear in the sequence length.

2.4 Training

We train the networks using stochastic gradient descent with shuffled mini-batches, the AdaDelta update rule, back-propagation and dropout. During the training, we also optimize the embedding tables (i.e, word, position and entity type embeddings) to achieve the optimal states. Finally, we rescale the weights whose l_2-norms exceed a predefined threshold (Nguyen and Grishman (2015a)).

3 Experiments

3.1 Dataset, Parameters and Resources

We apply *the same parameters and resources* as (Nguyen and Grishman, 2015b) to ensure the compatible comparison. Specifically, we employ the window sizes in the set {2, 3, 4, 5} for the convolution operation with 150 filters for each window size. The window size of the trigger candidate is 31 while the dimensionality of the position embeddings and entity type embeddings is 50. We use `word2vec` from (Mikolov et al., 2013b) as the pretrained word embeddings. The other parameters include the dropout rate $\rho = 0.5$, the mini-batch size = 50, the predefined threshold for the l_2 norms = 3.

Following the previous studies (Li et al., 2013; Chen et al., 2015; Nguyen and Grishman, 2015b), we evaluate the models on the ACE 2005 corpus

[2]We ignore the base cases as they are trivial.

with 33 event subtypes. In order to make it compatible, we use the same test set with 40 newswire articles, the same development set with 30 other documents and the same training set with the remaining 529 documents. All the data preprocessing and evaluation criteria follow those in (Nguyen and Grishman, 2015b).

3.2 The General Setting

We compares the non-consecutive CNN model (**NC-CNN**) with the state-of-the-art systems on the ACE 2005 dataset in Table 1. These systems include:

1) The feature-based systems with rich hand-designed feature sets, including: the MaxEnt model with local features in (Li et al., 2013) (**MaxEnt**); the structured perceptron model for joint beam search with local features (**Joint+Local**), and with both local and global features (**Joint+Local+Global**) in (Li et al., 2013); and the sentence-level and cross-entity models in (Hong et al., 2011).

2) The neural network models, i.e, the CNN model in (Nguyen and Grishman, 2015b) (**CNN**), the dynamic multi-pooling CNN model (**DM-CNN**) in (Chen et al., 2015) and the bidirectional recurrent neural networks (**B-RNN**) in (Nguyen et al., 2016a).

3) The probabilistic soft logic based model to capture the event-event correlation in (Liu et al., 2016).

Methods	F
Sentence-level in Hong et al (2011)	59.7
MaxEnt (Li et al., 2013)	65.9
Joint+Local (Li et al., 2013)	65.7
Joint+Local+Global (Li et al., 2013)	67.5
Cross-entity in Hong et al. (2011) †	68.3
Probabilistic soft logic (Liu et al., 2016) †	69.4
CNN (Nguyen and Grishman, 2015b)	69.0
DM-CNN (Chen et al., 2015)	69.1
B-RNN (Nguyen et al., 2016a)	69.3
NC-CNN	**71.3**

Table 1: Performance with Gold-Standard Entity Mentions and Types. † beyond sentence level.

The most important observation from the table is that the non-consecutive CNN model significantly outperforms all the compared models with large margins. In particular, *NC-CNN* is 2% better than B-RNN (Nguyen et al., 2016a), the state-of-the-art system that only relies on the context information within the sentences of the trigger candidates. In addition, although *NC-CNN* only employs the sentence-level information, it is still better than the other models that further exploit the document-level information for prediction (an improvement of 1.9% over the probabilistic soft logic based model in (Liu et al., 2016)). Finally, comparing *NC-CNN* and the CNN model in (Nguyen and Grishman, 2015b), we see that the non-consecutive mechanism significantly improves the performance of the traditional CNN model for ED (up to 2.3% in absolute F-measures with $p < 0.05$).

3.3 The Domain Adaptation Experiments

Previous studies have shown that the NLP models would suffer from a significant performance loss when domains shift (Blitzer et al., 2006; Daume III, 2007; Plank and Moschitti, 2013; Nguyen et al., 2015c). In particular, if a model is trained on some *source domain* and applied to a different domain (*the target domain*), its performance would degrade significantly. The domain adaptation (DA) studies aim to overcome this issue by developing robust techniques across domains.

The best reported system in the DA setting for ED is (Nguyen and Grishman, 2015b), which demonstrated that the CNN model outperformed the feature-based models in the cross-domain setting. In this section, we compare *NC-CNN* with the CNN model in (Nguyen and Grishman, 2015b) (as well as the other models above) in the DA setting to further investigate their effectiveness.

3.3.1 Dataset

This section also uses the ACE 2005 dataset but focuses more on the difference between domains. The ACE 2005 corpus includes 6 different domains: broadcast conversation (*bc*), broadcast news (*bn*), telephone conversation (*cts*), newswire (*nw*), usenet (*un*) and webblogs (*wl*). Following (Nguyen and Grishman, 2015b), we use ***news*** (the union of ***bn*** and ***nw***) as the source domain and ***bc***, ***cts***, ***wl*** and ***un*** as four different target domains[3]. We take half of *bc* as the development set and use the remaining data for testing. Our data split is the same as that in (Nguyen and Grishman, 2015b).

[3]Note that (Nguyen and Grishman, 2015b) does not report the performance on *un* but we include it here for completeness.

System	In-domain(bn+nw)			bc			cts			wl			un		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
MaxEnt	74.5	59.4	66.0	70.1	54.5	61.3	66.4	49.9	56.9	59.4	34.9	43.9	-	-	-
Joint+Local	73.5	62.7	67.7	70.3	57.2	63.1	64.9	50.8	57.0	59.5	38.4	46.7	-	-	-
Joint+Local+Global	72.9	63.2	67.7	68.8	57.5	62.6	64.5	52.3	57.7	56.4	38.5	45.7	-	-	-
B-RNN	71.4	63.5	67.1	70.7	62.1	66.1	70.0	54.4	61.0	52.7	38.3	44.2	66.2	46.0	54.1
DM-CNN	75.9	62.7	68.7	75.3	59.3	66.4	74.8	52.3	61.5	59.2	37.4	45.8	72.2	44.5	55.0
CNN	69.2	67.0	68.0	70.2	65.2	67.6	68.3	58.2	62.8	54.8	42.0	**47.5**	64.6	49.9	56.2
NC-CNN	74.9	66.5	**70.4**†	73.6	64.7	**68.8**†	71.7	57.3	**63.6**	57.8	40.3	47.4	71.7	49.0	**58.1**†

Table 2: Performance on the source domain and on the target domains. Cells marked with †designates that NC-CNN significantly outperforms ($p < 0.05$) all the compared methods on the specified domain.

3.3.2 Performance

Table 2 reports the performance of the systems with 5-fold cross validation. Note that we focus on the systems exploiting only the sentence level information in this section. For each system, we train a model on the training data of the source domain and evaluate this model on the test set of the source domain (in-domain performance) as well as on the four target domains *bc*, *cts*, *wl* and *un*.

We emphasize that the performance of the systems *MaxEnt*, *Joint+Local*, *Joint+Local+Global*, *B-RNN*, and *CNN* is obtained from the actual systems in the original work (Li et al., 2013; Nguyen and Grishman, 2015b; Nguyen et al., 2016a). The performance of *DM-CNN*, on the other hand, is from our re-implementation of the system in (Chen et al., 2015) using the same hyper-parameters and resources as *CNN* and *NC-CNN* for a fair comparison.

From the table, we see that *NC-CNN* is significantly better than the other models on the source domain. This is consistent with the conclusions in Section 3.2 and further confirms the effectiveness of *NC-CNN*. More importantly, *NC-CNN* outperforms *CNN* and the other models on the target domains *bc*, *cts* and *un*, and performs comparably with *CNN* on *wl*. The performance improvement is significant on *bc* and *un* ($p < 0.05$), thereby verifying the robustness of *NC-CNN* for ED across domains.

4 Related Work

There have been three major approaches to event detection in the literature. First, the pattern-based approach explores the application of patterns to identify the instances of events, in which the patterns are formed by predicates, event triggers and constraints on the syntactic context (Grishman et al., 2005; Cao et al., 2015a; Cao et al., 2015b).

Second, the feature-based approach relies on linguistic intuition to design effective feature sets for statistical models for ED, ranging from the local sentence-level representations (Ahn, 2006; Li et al., 2013), to the higher level structures such as the cross-sentence or cross-event information (Ji and Grishman, 2008; Gupta and Ji, 2009; Patwardhan and Riloff, 2009; Liao and Grishman, 2011; Hong et al., 2011; McClosky et al., 2011; Li et al., 2015). Some recent work on the feature-based approach has also investigated event trigger detection in the joint inference with event argument prediction (Riedel et al., 2009; Poon and Vanderwende, 2010; Li et al., 2013; Venugopal et al., 2014) to benefit from their inter-dependencies.

Finally, neural networks have been introduced into ED very recently with the early work on convolutional neural networks (Nguyen and Grishman, 2015b; Chen et al., 2015). The other work includes: (Nguyen et al., 2016a) who employ bidirectional recurrent neural networks to perform event trigger and argument labeling jointly, (Jagannatha and Yu, 2016) who extract event instances from health records with recurrent neural networks and (Nguyen et al., 2016b) who propose a two-stage training algorithm for event extension with neural networks.

5 Conclusion

We present a new CNN architecture for ED that exploits the non-consecutive convolution for sentences. Our evaluation of the proposed model on the general setting and the DA setting demonstrates the effectiveness of the non-consecutive mechanism. We achieve the state-of-the-art performance for ED in both settings. In the future, we plan to investigate the non-consecutive architecture on other problems such as relation extraction or slot filling.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Kai Cao, Xiang Li, and Ralph Grishman. 2015a. Improving event detection with dependency regularization. In *RANLP*.

Kai Cao, Xiang Li, Miao Fan, and Ralph Grishman. 2015b. Improving event detection with active learning. In *RANLP*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL-IJCNLP*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. In *ACE 2005 Evaluation Workshop*.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP*.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *ACL*.

Abhyuday N Jagannatha and Hong Yu. 2016. Bidirectional rnn for medical event detection in electronic health records. In *NAACL*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2015. Molding cnns for text: non-linear, non-consecutive convolutions. In *EMNLP*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *ACL*.

Xiang Li, Thien Huu Nguyen, Kai Cao, and Ralph Grishman. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS)*.

Shasha Liao and Ralph Grishman. 2011. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*.

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

David McClosky, Mihai Surdeanu, and Christopher Manning. 2011. Event extraction as dependency parsing. In *BioNLP Shared Task Workshop*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Thien Huu Nguyen and Ralph Grishman. 2015a. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the 1st NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Thien Huu Nguyen and Ralph Grishman. 2015b. Event detection and domain adaptation with convolutional neural networks. In *ACL-IJCNLP*.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015c. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *NAACL*.

Thien Huu Nguyen, Lisheng Fu, Kyunghyun Cho, and Ralph Grishman. 2016b. A two-stage approach for extending event detection to new types via neural networks. In *Proceedings of the 1st ACL Workshop on Representation Learning for NLP (RepL4NLP)*.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*.

Barbara Plank and Alessandro Moschitti. 2013. Embedding semantic similarity in tree kernels for domain adaptation of relation extraction. In *ACL*.

Hoifung Poon and Lucy Vanderwende. 2010. Joint inference for knowledge extraction from biomedical literature. In *NAACL-HLT*.

Sebastian Riedel, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *BioNLP 2009 Workshop*.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*.

Deepak Venugopal, Chen Chen, Vibhav Gogate, and Vincent Ng. 2014. Relieving the computational bottleneck: Joint inference for event extraction with high-dimensional features. In *EMNLP*.