# Neural Chinese Event Extraction

YING ZENG, Peking University
YANSONG FENG, Peking University
HONGHUI YANG, Peking University
ZHENG WANG, Lancaster University
DONGYAN ZHAO, Peking University

Event extraction is an important technology that underpins many text mining applications. It is a task of extracting events of certain types and their corresponding arguments of different roles from unstructured texts such as news articles and tweets. State-of-the-art event extraction approaches rely on high quality feature extraction, which requires careful template design and results of specific semantic or lexical analyzing tools as well as extra thesauruses. As a result, current approaches are hard to generalize, and difficult to adapt to new languages, especially those with limited resources or tools.

Different from English, the unique language structures and syntax of Chinese make prior work on English event extraction inapplicable to the new domain. In this work, we demonstrate, for the first time, it is possible to construct a generalized yet effective Chinese event extraction model without relying on manually designed feature templates or specific lexical or syntactic tools. Our novel, broadly applicable approach employs a convolution bidirectional long short-term memory neural network (`C-BiLSTM`) to automatically learn a set of effective feature representations. We achieve this by exploiting different neural network components, a convolutional component (`CNN`) for prominent lexical features and a bidirectional long short-term memory component (`BiLSTM`) capturing sentence level representations. To overcome the language-specific challenge of Chinese word segmentation, we combine a conditional random field (`CRF`) layer over a character-level neural network component to allow us tag each character beyond word boundary. We evaluate our approach on the ACE 2005 dataset, and experimental results show that our approach can deliver state-of-the-art performance in both trigger labeling and argument labeling without laborious feature engineering.

CCS Concepts: •**Computing methodologies** → **Information extraction; Neural networks;**

Additional Key Words and Phrases: Event Extraction, Chinese Natural Language Processing, Neural Network Models

**ACM Reference Format:**
Ying Zeng, Yansong Feng, Honghui Yang, Zheng Wang, Dongyan Zhao, 2017. Neural Chinese Event Extraction *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* V, N, Article A (January YYYY), 21 pages.
DOI: 0000001.0000001

---

*Extension of Conference Paper:* a preliminary version of this article entitled "A convolution BiLSTM neural network model for Chinese event extraction" by Y. Zeng, Y. S. Feng, H. H. Yang, Z. Wang and D. Y. Zhao appeared in the 24th International Conference on Computer Processing of Oriental Languages, ICCPOL 2016 FIX:[]. The extended version makes the following new contributions over the conference paper: FIX:(1) it provides xxx (Section xx); (2) it describes the selection xx (Section xx); (3) it provides new experimental results xxx (Section xx)...

This work was partially supported by the National High Technology R&D Program of China under Grant No. 2015AA015403, the Natural Science Foundation of China under Grant agreements 61202233, 61272344 and 61370055, the UK Engineering and Physical Sciences Research Council under grants EP/M01567X/1 (SANDeRs) and EP/M015793/1(DIVIDEND), and a joint project with IBM Research. Any correspondence please refer to Yansong Feng.

Author's addresses: Y. Zeng and Y. Feng and H. Yang and D. Zhao, Institute of Computer Science and Technology, Peking University, P.R. China; Z. Wang, School of Computing and Communications, Lancaster University, UK.

## 1. INTRODUCTION

Event extraction is the task of finding specific events and then extracting structured knowledge of the events from unstructured data such as news articles, tweets, or blogs. It is an important technique that underpins many text mining applications such as news recommendation and information filtering. The goal of event extraction is to discover structured information about an even occurrence like "*who did what to whom, when, where through what methods, and why*" [Piskorski and Yangarber 2013]. A concrete example of event extraction is given in Figure 1. Given a text document, an event extraction system should be able to identify all the relevant entities and their relationships.

Finding a set of high-quality features, i.e. feature engineering, is key to event extraction. Most current approaches rely on hand-crafted features to build multi-class classifiers to perform various event-extraction-related tasks [Ahn 2006; Chen and Ji 2009; Li et al. 2012; Chen and NG 2012]. The features used in prior work can be divided into two categories: *lexical features* and *sentence-level features*, where the former are designed to capture the local information of words, such as part-of-speech tags or word lemma, while the latter are used to characterize the topic of the whole sentence. The challenge of finding useful features is that the space of possible features is vast, while their effect depends on the natural language itself, the structure of the data, and the application domain.

Despite much progress has been made in English event extraction, building an effective Chinese event extraction system remains a challenge. There have been efforts to solve this problem using hand-crafted feature templates building upon specific natural language processing (NLP) tools, such as POS tagging, syntactic parsing, semantic role labeling, and so on. While it is possible to manually design a set of task-specific feature templates, current approaches require extensive human involvement and are tightly couple to the information provided by specific NLP tools. This means that when we target a new language, where limited tools or resources are available, despite of the time-consuming process of feature engineering, the system will be heavily affected by the performance of existing NLP tools for that new language. Unfortunately, most low-resourced languages have far-from-perfect performances even in their fundamental semantic or syntactic analysis, e.g., Chinese syntactic parsing performance is around 5% lower than that of English in newswire, which leads to significant performance drop in downstream applications. Therefore, it is essential to have a technique that can automatically discover lexical and sentence-level feature representations for event extraction in a low-resourced language, without extensive human involvement and the heavy reliance on specific NLP tools.

In this paper, we propose to utilize neural networks to alleviate the burden of feature engineering for Chinese event extraction. To do so, we develop a convolution bidirectional long short-term memory (`LSTM`) network (`C-BiLSTM`) to automatically learn lexical and sentence level feature representations without extensive human involvement. We achieve this by first constructing a bidirectional `LSTM` to encode the entire sentence into sentence-level features without performing dependency parsing or semantic role labeling (`SRL`). We then take advantage of a convolutional neural network to capture salient local features to semantically characterize event occurrences, without relying on part-of-speech (`POS`) tagging or named entity recognition (`NER`). To address the word segmentation issue, a fundamental yet challenging step for Chinese language processing, we employ a conditional random field (`CRF`) layer to automatically learn sentence-level labeling constraints and jointly decode the best labeling sequence for the whole sentence.

Putting together these techniques, we are able to build a powerful system for Chinese event extraction. As we show, our approach can automatically learn feature representations that lead to the same level of performance compared to hand-tuned features selected by human experts. This is truly remarkable when one considers that our approach is given no prior knowledge of the syntax or semantics of the target language.

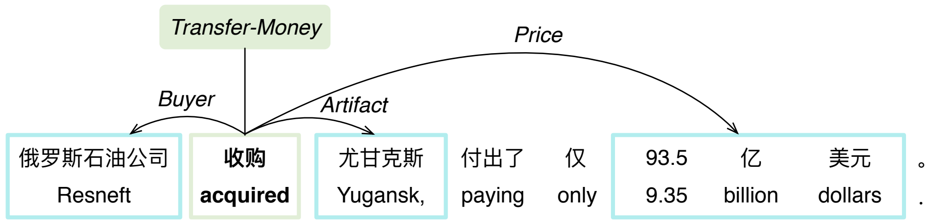


Fig. 1. An example of Chinese event extraction for S1, where English translations are provided besides the Chinese words. The example sentence has one *Transfer-Money* event mention triggered by 收购 *(acquired)*, which have three arguments, 俄罗斯石油公司 *(Rosneft)* as the *Transfer-Money* event's *Buyer*, 尤甘斯克 *(Yugansk)* as its *Artifact*, and *93.5* 亿美元 *(9.35 billion dollars)* as its *Price*.

We evaluate our approach on the ACE 2005 Chinese event extraction corpus which consists of 8 event types and 33 subtypes. We compare our approach against a set of well-established traditional feature-based models available for this heavily-studied data set. The experimental results show that our automatic approach can effectively learn both local semantic and sentence level representations for event extraction and deliver state-of-the-art performance, when compared to the hand-tuned features developed by human experts.

The rest of this paper is organized as follows. Section 2 explains the terminologies and defines the scope of the work. Section 3 discusses related Chinese event extraction systems. Section 4 and Section 5 present the neural network models applied to trigger labeling and argument labeling. Section **??** presents the experimental results and analyses. We conclude our work in Section 7.

## 2. BACKGROUND

This work focuses on the event extraction task proposed by the Automatic Content Extraction (ACE) program [Doddington et al. 2004] which defines the following terminologies for event extraction:

— **Event mention**: a phrase or sentence within which an event is described, including a trigger and its arguments.
— **Event trigger**: the main word or phrase that most clearly expresses the occurrence of an event. An event trigger is typically a verb or a noun in the ACE framework.
— **Event argument**: an entity, temporal expression or value that is involved in an event.
— **Argument role**: the relationship between an argument and the event in which it participates.

Specifically, in the initial 2005 evaluation, there are three languages in the ACE event extraction task, English, Chinese and Arabic. Similar to many other NLP tasks, English event extraction receives relatively more attention in the literature, while Chinese and Arabic are not in the spotlight. In this paper, we aim to improve Chinese event extraction.

Despite of the differences among these three languages, their ACE task setups remain the same. Given a text document, an event extraction system need to identify all event triggers with specific types and their corresponding arguments for each sentence. For example, given an example sentence, S1, in Chinese, the system should produce a structured output, including a typed event trigger and its corresponding arguments with pre-defined roles, as shown in Figure 1.

S1: 俄罗斯石油公司 收购 尤甘斯克 付出 了 仅 93.5 亿 美元。
Rosneft acquired Yugansk, paying only 9.35 billion dollars.

To produce such a structured output, a system usually has to solve the following four subtasks,

(1) **Trigger identification**: recognize the event trigger from the sentence, e.g., *"收购" (acquired)* should be identified as a trigger of an event;
(2) **Trigger classification**: assign an event type for the identified trigger, e.g., *"收购" (acquired)* should be assigned with the type *Transfer-Money*;
(3) **Argument identification**: determine whether a word or phrase plays a certain role in the event mention, e.g., *"俄罗斯石油公司" (Rosneft)*, *"尤甘斯克" (Yugansk)*, and *"93.5亿美元" (9.35 billion dollars)* are the arguments of this *Transfer-Money* event;
(4) **Argument classification**: assign a role to each identified argument, e.g., *"俄罗斯石油公司" (Rosneft)*, *"尤甘斯克" (Yugansk)*, and *"93.5亿美元" (9.35 billion dollars)* play the roles of *Buyer*, *Artifact* and *Price*, respectively, in this *Transfer-Money* event.

However, such a 4-step pipeline system inevitably suffers from the error propagation problems, let alone language specific challenges, e.g., Chinese word segmentation, and insufficient capability of existing NLP tools for those low-resourced languages. As addressed in the literature [Chen and Ji 2009; Li et al. 2012], errors made by an upstream subtask, e.g., Chinese word segmentation or trigger identification, will propagate to the downstream subtasks, e.g., tigger classification or argument identification, and could adversely affect their performance. In most cases, downstream subtasks will have no chance to even alleviate such effect. The work presented in [Chen and NG 2012] addresses the issue by recasting the 4-step framework into two joint learning tasks, namely, **trigger labeling** and **argument labeling**. The former jointly learns trigger identification and classification, while the latter learns argument identification and argument classification simultaneously, which has been shown to be promising in Chinese event extraction [Chen and NG 2012]. In this work, we follow this 2-step approach for Chinese event extraction.

Previous state-of-the-art approaches in Chinese event extraction [Chen and Ji 2009; Li et al. 2012; Chen and NG 2012; Li et al. 2013] usually rely on a variety of elaborate features, which, in general, must be designed by experts, and extracted using lexical or syntactic analysis tools. Those features can be roughly divided into two categories: **lexical features** and **sentence-level features**. The former are designed to capture local features, while the latter characterize global topics of the sentences. Take trigger labeling in the following sentences as an example: the target word *"成立" (found)* indicates a *Business* event in S2, but not in either S3 or S4.

S2: Intel 在 中国 **成立** 了 研究中心。
Intel **founds** a research center in China.

S3: 它 **成立** 于 1994 年 ， 现在 是 一支 深 受 欢迎 的 乐队。
It was **founded** in 1994, and now is a very popular band.

S4: 医院 已 **成立** 救援中心。
The hospital has **founded** rescue centers.

**Lexical features**, often in the form of word lemma or part-of-speech tags (POS), provide semantic information of a candidate word and its surrounding context. In S4, the neighboring word "救援中心"(rescue center) serves as a lexical feature for *"成立" (found)*, indicating a medical institute, thus we can infer that *"成立" (found)* is not a trigger of a *Business* event.

**Sentence-level features** are usually extracted from syntactic structures, and maintain important clues about the whole sentence. S3 talks about "它_是_乐队" (it_is_band) and "它_成立_1994" (it_founded_1994), indicating the verb "成立" (founded) relates to a musical group, thus should not be regarded as a trigger of a *Business* event.

The aim of this work is to investigate ways to automatically find effective features for **trigger labeling** and **argument labeling**. Although event extraction (mainly argument labeling) also depends on name identification and entity mention co-reference, which are challenging by themselves, these techniques fall out of the scope of this work. In comply with the standard practice in prior work [Chen and Ji 2009; Chen and NG 2012], we directly utilize the entity recognition ground-truth provided by ACE for argument labeling. We stress that other work on improving argument labeling is thus orthogonal to our approach.

Traditional event extraction methods [Ahn 2006; Chen and Ji 2009; Li et al. 2012; Chen and NG 2012] rely on those carefully-designed features to build multi-class classifiers for each subtask. They first apply a series of NLP tools to extract lexical features (e.g., `POS` tagging, and named entity recognition (`NER`)) and sentence-level features (e.g., dependency parsing, and semantic role labeling (`SRL`)). Then in the training stage, they learn to weight each feature for each subtask in the pipeline, independently, with a classifiers, e.g., conditional random fields (`CRF`) [Lafferty et al. 2001], support vector machines [Suykens and Vandewalle 1999], maximum entropy [Phillips et al. 2006], etc. Although those methods can achieve satisfactory performances, they still suffer from the hard feature engineering, where human expert involvement is necessary, and many semantic/syntactic tools may not work as well as we expected. This is a particular problem for low-resourced languages, for which the overall performance is often below 70%, resulting in incorrect feature extraction and confusing the classifier.

S5: 犯罪 嫌疑人 都 **落入 法网**。
The suspects were **arrested**.

S6: 警察 **击毙** 了 一名 歹徒。
Polices **shoot** and **kill** a criminal.

S7: 这 是 一件 预谋 的 **凶杀**案。
It is a premeditated **murder** case.

Take Chinese event extraction systems as an example. Current Chinese systems do not perform as well as their English counterparts. One of the major obstacles, as suggested by Li et al. [2012], is caused by the Chinese word segmentation, a unique preprocessing step for Chinese. Unlike English, Chinese does not have delimiters between words, which makes word segmentation the very first and fundamental step in Chinese event detection. However, we find that word segmentation granularity does have a great impact on the trigger labeling performance. For example, triggers in S5∼S7 cannot be recognized accurately if we simply predict whether a word is an event trigger or not. In S5, the word "落入法网" is usually separated into two words, while the gold-standard trigger is "落入法网" as a whole; while in S6, "击毙", in most cases, is treated as one single word, but actually, the characters "击" and "毙" are two triggers of different types, the former an *Attack* event and the latter a *Die* event, respectively. As in such a pipeline framework, incorrect trigger labeling will definitely affect downstream argument labeling, whose inputs are highly related to the predicated triggers.

One of the key innovations of our approach is to recast trigger labeling as a sequence labeling task, and further to combine different neural network components to automatically learn feature representations of different levels capturing both local lexical and sentence level characteristics for event extraction. Doing so allows us to address the Chinese language-specific issues mentioned above, by exploiting the recent breakthrough effectiveness of deep neural networks in natural language modeling, and to alleviate the burden of hard feature engineering and the reliance on external tools or thesauruses. In the reminder of this paper, we will describe how to build such a system in two-step fashion, trigger labeling and argument labeling.

## 3. RELATED WORK

Event extraction is one of the challenging tasks in information extraction. Many approaches have been explored from different views, which can be divided into traditional feature-based methods and neural-network-based methods.

Feature-based methods exploit various feature extraction strategies and evaluate feature contributions for prediction. Ahn [2006] models the event extraction task as a pipeline of four classification tasks and applies lexical features (e.g., full word, word lemma, etc.), syntactic features (e.g., dependency trees, SRL structures, etc.), and features generated from external knowledge resources (e.g., WordNet). Later studies also employ context sentences and higher level information to characterize global features. For example, Ji and Grishman [2008] utilize global evidence from a cluster of topically-related documents to break down the document boundaries for event extraction. And Patwardhan and Riloff [2009] consider both local context and a wider sentential context around a phrase to estimate whether a sentence is discussing an event of interest. Gupta and Ji [2009] employ cross-event features to extract implicit time information. Liao and Grishman [2010] explore document level cross-event information to resolve ambiguities between certain types of events. Hong et al. [2011] leverage cross-entity inference to predict event mentions. Huang and Riloff [2012] propose a bottom-up approach that initially identifies candidate argument independently and then uses that information to model textual cohesion. More recently, some researchers have tried to improve other aspects of event extractions. As opposed to traditional pipelined approaches, Li et al. [2013] attempt to jointly learn all ACE event extraction subtasks using structured perceptron and incorporate global features which explicitly capture the dependencies of multiple triggers and arguments in a transition system.

Recently, neural network models have been employed to achieved competitive performances against traditional models in many NLP tasks. Chen et al. [2015] propose a convolutional neural network (CNN) to capture lexical-level clues, with a dynamic multi-pooling layer to capture sentence-level features. Recurrent neural network (RNN) is also a popular and effective choice for classification tasks. Ghaeini et al. [2016] effort to detect triggers that can be either words or phrases by a forward-backward RNN. Nguyen et al. [2016] propose a bidirectional RNN with various memory matrices to jointly learn triggers and arguments, which benefits from the advantages of both joint models and neural network models. Feng et al. [2016] introduce a language-independent hybrid neural network model that incorporates both bidirectional LSTMs and convolutional neural networks, which yields competitive performances on both English and Chinese event extraction.

Compared to the amount of work on English event extraction, there are a few studies focusing on Chinese event extraction, and most of them are feature-based methods. Chen and Ji [2009] are the first to report the language specific issues in Chinese trigger labeling, and apply various kinds of lexical, syntactic and semantic features to the modularized pipeline system. One of their simple yet effective strategies is to build an errata table to deal with the inconsistency between word segmentation results and trigger annotations. The table includes the correspondence between words and their frequent labeling results, e.g., the word "打死" is often labeled as "打"$-$*Attack* and "死"$-$*Die*, and during testing, if the word "打死" has been identified as a trigger candidate, we will directly label each character with their specified event types in the errata table, respectively. However, this strategy is incapable to handle the "落入法网" case, where "落入" and "法网" are usually treated as two words, since the candidate search space is vast. Li et al. [2012] also explore language phenomena in Chinese event extraction, and alleviate the word segmentation errors via compositional semantics inside Chinese triggers and discourse consistency between Chinese triggers. The state-of-art feature-based model investigates two extensions to Li et al. [2012], namely joint-learning method and knowledge-rich method. As for the first extension, they divide the original four-step pipeline system into two joint learning tasks where they jointly

Table I. Numbers of triggers inconsistent with the words.

| NLP Tool | Cross-word Trigger | Inside-word Trigger | Total |
|---|---|---|---|
| Stanford NLP[1] | 14.6% | 5.0% | 19.6% |
| Jieba[2] | 16.6% | 2.6% | 19.2% |
| NLPIR[3] | 8.9% | 5.2% | 14.1% |

learn trigger identification and trigger classification, and then jointly learn argument identification and argument classification. In the second extension, they apply six groups of features motivated by deep investigations on the dataset: zero pronoun features and trigger type consistency features extracted by well-designed rules, syntactic and semantic features extracted using a Chinese semantic role labeling system, and also help from external linguistic resources, e.g., a Chinese synonym dictionary for character-based features. As we can see, this method heavily relies on elaborate feature templates and external tools or thesauruses, thus may suffers from the errors propagated from other NLP tools.

In this paper, we follow the two-step joint-learning style that alleviates the error propagation problems of traditional pipeline systems, and propose two neural network models to avoid heavy feature engineering and extra resources. Specifically, the CNN component learns lexical features while the bidirectional LSTM component extracts sentence-level features automatically. We further combine a CRF layer to learn sequence labeling preferences to jointly decode the best labels. To further deal with specific issues with respect to Chinese, we, instead of labeling word by word, we introduce the character-based models to avoid the reliance on carefully designed resources for Chinese event extraction.

## 4. TRIGGER LABELING

Trigger labeling, also called event detection, aims to discover event triggers and assign them a predefined event type. We jointly learn trigger identification and type classification by one network to reduce the error propagation problem in a pipeline model. Before we present our solution, we first discuss the language specific issues in Chinese trigger labeling.

### 4.1. Language Specific Issues

We summarize the inconsistency problems between words and triggers as the following two types.

(1) Cross-word triggers: While many events anchor on a single word, multiple words could reasonably be called a trigger. In S4, the *arrest_jail* event should be triggered by neither “落入” nor “法网”, but “落入法网” (arrested).
(2) Inside-word triggers: Almost all Chinese characters have their own meanings, and some of which can be triggers themselves. There may be greater than one trigger in a word like “击毙” (shoot and kill). Continuous characters of a word can also form a trigger such as “凶杀” (murder) in “凶杀案” (murder case).

Table I summarizes the number of problematic triggers we found in ACE 2005 Chinese corpus using different Chinese word segmentation tools. Even the minimum inconsistency rate is as high as 14%.

To address the language specific issues, we treat event detection as a sequence labeling task. Formally, we use $\boldsymbol{w} = \{w_1, w_2, \ldots, w_n\}$ to represent an input sentence of length $n$, where $w_i$ is the $i$-th word. And $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ represents a generic sequence of labels for $\boldsymbol{w}$. Each word is tagged in the `BIO` scheme, where each token is labeled as `B-type` if it is the beginning of an event trigger with event type `type`, or `I-type` if it is inside a trigger, or `O`

[1]`http://nlp.stanford.edu/software/segmenter.shtml`
[2]`https://github.com/fxsjy/jieba`
[3]`https://github.com/NLPIR-team/NLPIR`

otherwise. Our first labeling model is a word-based bidirectional LSTM network (BiLSTM) with a CNN layer as shown in Figure 2.

## 4.2. Word-based Convolution BiLSTM Model

In this section, we introduce the components (layers) in our Convolution BiLSTM (C-BiLSTM) network one-by-one from bottom to top.

*LSTM Network.* Recurrent neural networks (RNNs) maintain a memory based on historical contextual information, which makes them a natural choice for processing sequential data. Unfortunately, it is difficult for standard RNNs to capture long range dependencies due to vanishing/exploding gradients [Bengio et al. 1994]. Long Short-Term Memory Network [Hochreiter and Schmidhuber 1997] is explicitly designed to solve the long-term dependency problem through purpose-built memory cells. They consist of several multiplicative gates that control the proportion of information to forget and to store in the cell states. In this paper, we apply a variation of LSTM units, Gated Recurrent Unit (GRU) [Cho et al. 2014], which is found to be superior to LSTM on a suit of tasks by Chung et al. [2014]. The GRU is implemented as the following formulas:

$$\begin{aligned}
\mathrm{r}_t &= \sigma(\mathrm{W}_{xr}\mathrm{x}_t + \mathrm{W}_{hr}\mathrm{h}_{t-1} + \mathrm{b}_r) \\
\mathrm{z}_t &= \sigma(\mathrm{W}_{xz}\mathrm{x}_t + \mathrm{W}_{hz}\mathrm{h}_{t-1} + \mathrm{b}_z) \\
\tilde{\mathrm{h}}_t &= \tanh(\mathrm{W}_{xh}\mathrm{x}_t + \mathrm{W}_{hh}(\mathrm{r}_t \odot \mathrm{h}_{t-1}) + \mathrm{b}_h) \\
\mathrm{h}_t &= \mathrm{z}_t \odot \mathrm{h}_{t-1} + (1 - \mathrm{z}_t) \odot \tilde{\mathrm{h}}_t
\end{aligned}$$

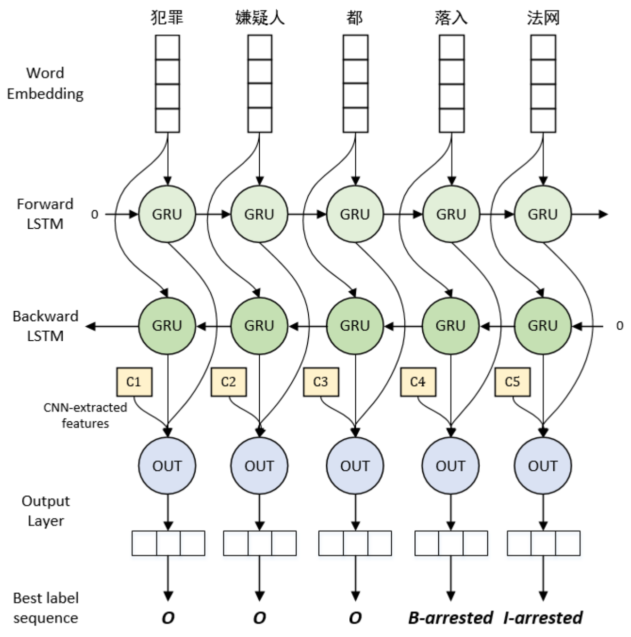
In these formulas, the $\mathrm{W}_*$ variables are the weight matrices and the $\mathrm{b}_*$ variables are the biases. $\sigma(\cdot)$ is the element-wise sigmoid function and $\odot$ is the element-wise product.

Formally, we use $\mathrm{x}_t$ to represent the feature vector (e.g. word embedding) corresponding to the $t$-th word $w_t$. At each time step $t$, a GRU takes $\mathrm{x}_t$ as input and computes the hidden state (also called output vector) $\mathrm{h}_t$ by reset gate and update gate. The reset gate $\mathrm{r}_t$ controls how much and what information from the previous hidden state should be reset, and the update gate $\mathrm{z}_t$ determines how much the unit updates its previous hidden state.

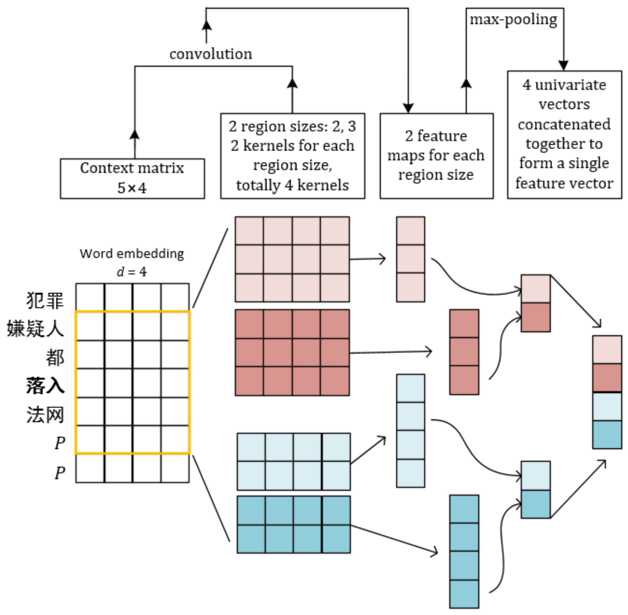
*BiLSTM Network.* In the event extraction task, if we can access to both past and future contexts for a candidate trigger, we can explore richer sentence-level information and make better prediction. This can be done by bidirectional LSTM networks [Graves and Schmidhuber 2005; Graves et al. 2013]. Figure 1(a) shows the layers of a BiLSTM trigger identification model.

At time $t$, a forward LSTM network computes the hidden state $\overrightarrow{\mathrm{h}_t}$ of the past (left) context of the sentence at word $w_t$, while a backward LSTM network reads the same sentence in reverse and outputs $\overleftarrow{\mathrm{h}_t}$ given the future (right) context. We concatenate these two vectors to form the output vector of a BiLSTM network, i.e. $\mathrm{B}_t = [\overrightarrow{\mathrm{h}_t}; \overleftarrow{\mathrm{h}_t}]$.

*CNN Layer.* Convolutional neural networks (CNNs) are originally applied to computer vision to capture salient local features [LeCun et al. 1998]. Previous studies on event extraction [Nguyen and Grishman 2015; Chen et al. 2015; Feng et al. 2016] have gradually shown that CNN architectures are effective to capture semantic features similar to n-grams, but represent them in a more compact way. We employ a convolutional neural network as illustrated in Figure 1(b) to extract local contextual information for each word in a sentence. Specifically, for every word in the sentence, we want to extract local contextual information to help predict whether the current word is an event trigger. The current word $w_i$ along with its context constitutes the input of CNN. Let $w_{i:i+j}$ be the window of words from $w_i$ to $w_{i+j}$, and $2k + 1$ be the fixed context size. So the context window $c_i$ (padded when



(a) Convolution BiLSTM network


convolution
max-pooling
Context matrix 5×4
2 region sizes: 2, 3 2 kernels for each region size, totally 4 kernels
2 feature maps for each region size
4 univariate vectors concatenated together to form a single feature vector
Word embedding d = 4
犯罪
嫌疑人
都
落入
法网
P
P

(b) Details of CNN in (a)

Fig. 2. The main architecture of our word-based convolution bidirectional LSTM model. The local contextual feature $c_i$ (yellow rectangle) in (a) for each word $w_i$ is computed by the CNN as (b) illustrated. Our CNN layer learns a representation of local context information about the center word “落入”. Here the context size is 5 (2 words to the left and to the right of a center word), and we depict two kernel region sizes: 2 and 3, each of which has 2 kernels. The symbol $P$ in sentence of (b) represents a padding word.

necessary) where the current word $w_i$ is in the middle can be written as

$$c_i = w_{i-k:i+k} = [w_{i-k}, \dots, w_i, \dots, w_{i+k}]. \tag{1}$$

Before being fed into a convolution layer, each word $w_i$ is transformed into a d-dimensional word vector $\mathrm{x}_i$ by looking up the embedding table. As a result, the original context $c_i$ is transformed into a matrix $\boldsymbol{c}_i = [\mathrm{x}_{i-k}, \dots, \mathrm{x}_i, \dots, \mathrm{x}_{x+k}]$ of size $(2k+1) \times d$. The matrices $\boldsymbol{c}_1, \dots, \boldsymbol{c}_n$ are then passed through a convolution layer and a max pooling layer. In the convolution layer, we utilize a set of kernels $\{\mathrm{w}_1, \mathrm{w}_2, \dots, \mathrm{w}_m\}$ with various widths to extract semantic features like n-grams of different granularities. For every context matrix $\boldsymbol{c}_i$, a kernel $\mathrm{w}_j$ of width $l$ is applied to all possible windows of $l$ words within the context (i.e., $w_{i-k:i-k+l-1}, \dots, w_{i+k-l+1:i+k}$). And $\mathrm{w}_j$ can be essentially seen as a weight matrix of size $l \times d$. For example, the convolution operation involves kernel $\mathrm{w}_j$ over the window $w_{t:t+l-1}$ can be express as:

$$s_{jt} = f(\mathrm{w}_j \cdot \mathrm{x}_{t:t+l-1} + b_j), 1 \le j \le m, i-k \le t \le i+k-l+1 \tag{2}$$

where $b$ is a bias term and $f$ is a non-linear function such as hyperbolic tangent. The convolution result is a feature map $\mathrm{s}_j \in \mathbb{R}^{2k-l+2}$. We then perform a max-over-time pooling operation [Collobert et al. 2011] over each feature map,

$$\tilde{s}_j = \max\{\mathrm{s}_j\} = \max\{s_{j1}, s_{j2}, \dots, s_{jm}\} \tag{3}$$

so that only the largest number is recorded. One property of pooling is that it produces a fixed size output vector, which enables us to apply variable kernel sizes. And by performing the max operation, we are keeping the most salient information. Finally, we take the fixed length output vector $\mathrm{C} = [\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_m]$ as a representation of local contextual information about the current word.

In our implementation, the context window size is 7 (3 words to the left and to the right of a center word), and kernel sizes from 2 to 7 to encode the semantics of n-grams with various granularities. Each kernel generates 32 feature maps.

*Output Layer.* For each word $w_i$ in the sentence, we concatenate the bidirectional sentence-level features $\mathrm{B}_i$ learned by BiLSTM, and the contextual semantic features $\mathrm{C}_i$ extracted by CNN, into a single vector $\mathrm{F}_i = [\mathrm{B}_i; \mathrm{C}_i]$. To compute the confidence of each label, the final feature vector $\mathrm{F}_i \in \mathbb{R}^{2d_{gru}+d_{cnn}}$, where $d_{gru}$ is the dimension of the GRU unit and $d_{cnn}$ is the number of feature maps in CNN layer, is fed into a fully connected linear layer.

$$\mathrm{O}_i = \mathrm{W}_s \mathrm{F}_i + \mathrm{b}_s \tag{4}$$

where $\mathrm{W}_s \in \mathbb{R}^{n_e \times (2d_{gru}+d_{cnn})}$ is the transformation matrix and $\mathrm{O}_i = [O_{i,1}, \dots, O_{i,n_e}]$ is the final output vector, $n_e$ is the number of distinct labels, and the $e$-th element $O_{i,e}$ indicates the score for label $e$.

Then $\mathrm{O}_i \in \mathbb{R}^{n_e}$ is fed into a softmax layer to estimate a probability distribution over all possible labels. In the end, we choose the label that obtains maximum probability as the prediction for $y_i$.

$$P(e \mid \mathrm{O}_i) = \frac{\exp(O_{i,e})}{\sum_{j=1}^{n_e} \exp(O_{i,j})} \tag{5}$$

$$y_i = \operatorname*{argmax}_e P(e \mid \mathrm{O}_i) \tag{6}$$

*Errata Table.* However, this word-base model still suffer from the inconsistency problem caused by inside-word triggers. Inspired by Chen and Ji [2009], we construct a global errata table to record some frequent appearances of tokens and triggers in the training set. In our

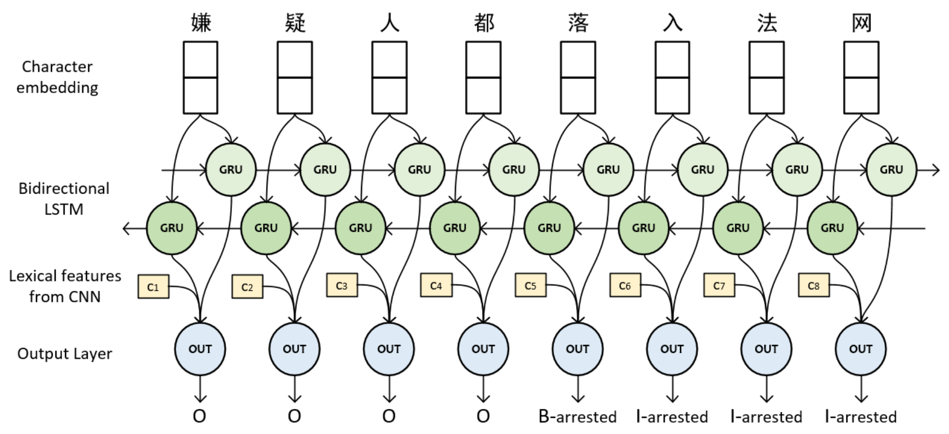


Fig. 3. Character-based Convolution BiLSTM network.

Table II. Difference of output sequences between character-based method and word-based method

| | |
|---|---|
| Sentence: | 犯罪 嫌疑人 都 **落入 法网**。<br>The suspects were **arrested**. |
| C-BiLSTM$_{word}$: | `O O O B-arrested I-arrested` |
| C-BiLSTM$_{char}$: | `O O O O O O B-arrested I-arrested I-arrested I-arrested` |
| Sentence: | 警察 **击毙** 了 一名 歹徒。<br>Polices **shoot** and **kill** a criminal. |
| C-BiLSTM$_{word}$: | Can not label this sentence correctly. |
| C-BiLSTM$_{char}$: | `O O B-attack B-die O O O O O` |

experiments, if 80% occurrences of a token inside a word should be labeled as triggers with the same event type, we then add this "word−token−type" triple into the table. During testing, if a word has an entry in the errata table, we regard its token as a trigger with the event type according to the corresponding triple directly. For example, if "击毙−击−*Attack*" and "击毙−毙−*Die*" are two triples in the errata table, word-based C-BiLSTM model can identify all inside-word triggers in S6 correctly.

## 4.3. Character-based Convolution BiLSTM Model

Despite of the effectiveness of errata table, word-based method is not a perfect solution to language specific issues, because it only recognizes triggers in a word level or frequent inside-word triggers appearing in training data. When it comes to an unknown inside-word trigger which never occurs during training, there is no way for an errata table to label it correctly.

Ideally, character-based methods may solve both inconsistency problems, which use the same {`BIO`}`-type` tagging scheme as word-based model, however, to label each character rather than each word. For better understanding, resulting sequences of example sentences S4 and S5 labeled by two models are listed in Table II.

As shown in Figure 3, character-based C-BiLSTM has a similar network architecture as word-based C-BiLSTM. The main difference is that character-based model tags a sentence character by character, while word-based model tags a sentence word by word. They also differ in their input layers: character-based C-BiLSTM uses character embeddings as input feature vectors, while word-based C-BiLSTM utilizes word embeddings.

### 4.4. Character-based Convolution BiLSTM-CRF Model

As mentioned in Section 4.2, regarding the final feature vector F as an input to a softmax classifier is a straightforward but effective way to make independent labeling decisions. However, the independent classification decisions are limited when there are strong dependencies between tags in a sentence. For example, in trigger labeling, `B-attack` is more likely to be followed by `B-die`, while `I-die` cannot follow `B-attack`. **Output layers in C-BiLSTM models cannot exploit this sentence level labeling information.**

Therefore, we propose a character-based convolution BiLSTM-CRF model (C-BiLSTM-CRF) that considers the correlations between labels in neighborhoods and jointly decodes the best sequence of labels via a CRF layer. This kind of CRF architecture is similar to the ones presented in other sequence labeling tasks, such as chunking and NER [Collobert et al. 2011; Huang et al. 2015; Lample et al. 2016].

Given an input sentence

$$\boldsymbol{X} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n),$$

we consider $\boldsymbol{P}$ to be a matrix of confidence scores output by C-BiLSTM network.

$$\boldsymbol{P} = [\mathrm{O}_1, \mathrm{O}_2, \ldots, \mathrm{O}_n] \tag{7}$$

$\boldsymbol{P}$ is of size $n \times e$, where $e$ is the number of distinct tags, and $P_{i,j}$ correspond to the confidence of the $j$-th tag for the $i$-th word in a sentence.

We introduce a state transition matrix $\boldsymbol{A}$ in CRF layer such that $A_{i,j}$ represents the score of a transition from label $i$ to label $j$. We take into account neural network outputs and transition scores, and score a sentence $\boldsymbol{X}$ along with a path of labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ to be

$$score(\boldsymbol{X}, \boldsymbol{y}) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}, \tag{8}$$

where $y_0$ and $y_{n+1}$ are the special labels, `start` and `end`, that we add to the set of possible labels. $\boldsymbol{A}$ is therefore a square matrix of size $(e+2) \times (e+2)$.

We normalize this score over all possible label sequences $\tilde{\boldsymbol{y}}$ using a softmax, and we interpreted the resulting ratio as a conditional label sequence probability:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \frac{\exp(score(\boldsymbol{X}, \boldsymbol{y}))}{\sum_{\tilde{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}} \exp(score(\boldsymbol{X}, \tilde{\boldsymbol{y}}))}. \tag{9}$$

where $\boldsymbol{\mathcal{Y}}$ represents all possible label sequences, including those invalid `BIO` scheme.

During training, for a dataset $\mathcal{T}$, we want to maximize the following logarithm of likelihood with respect to parameters $\tilde{\theta}$, which contains all neural network parameter $\theta$ and transition score matrix $\boldsymbol{A}$ in the CRF layer.

$$L(\tilde{\theta}) = \sum\nolimits_{(\boldsymbol{X}, \boldsymbol{y}) \in \mathcal{T}} \log\left(p(\boldsymbol{y}|\boldsymbol{X})\right) \tag{10}$$

$$\begin{aligned} \log\left(p(\boldsymbol{y}|\boldsymbol{X})\right) &= score(\boldsymbol{X}, \boldsymbol{y}) - \log\left(\sum\nolimits_{\tilde{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}} \exp(score(\boldsymbol{X}, \tilde{\boldsymbol{y}}))\right) \\ &= score(\boldsymbol{X}, \boldsymbol{y}) - \underset{\tilde{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}}{\mathrm{logadd}}\, score(\boldsymbol{X}, \tilde{\boldsymbol{y}}) \end{aligned} \tag{11}$$

Although the number of terms in the logadd operation in Equation (11) is equal to the size of $\boldsymbol{\mathcal{Y}}$, which grows exponentially with the length of the sentence, we can compute it in linear time, proved by Collobert et al.[2011].

At test time, given a sentence $\boldsymbol{X}$ to label, we have to find the best label sequence with the maximum score given by:

$$\boldsymbol{y}^* = \underset{\tilde{\boldsymbol{y}} \in \boldsymbol{\mathcal{Y}}}{\mathrm{argmax}}\, score(\boldsymbol{X}, \tilde{\boldsymbol{y}}) \tag{12}$$

Naturally, this inference can be solved efficiently by adopting the Viterbi algorithm [Rabiner 1989].

## 5. ARGUMENT LABELING

After trigger labeling, a sentence will be labeled with an event trigger, and its corresponding event type. Next step is argument labeling, including two subtasks: argument identification and argument classification. Given the related event trigger, argument identification aims to determine whether a candidate argument serves as a participant or an attribute with a specific role or not, and argument classification assigns a role to each identified argument. Similar to trigger labeling, we propose a joint argument labeling model that jointly performs argument identification and argument classification.

However, it is worth mentioning that unlike trigger labeling, argument labeling is no longer a sequence tagging task, **but a classification task: given a pair of trigger and candidate argument, determine the relationship between them.** ACE dataset provides ground truth about entity mention, value, time expression recognition, and it guarantees that gold standard arguments are annotated exactly from those particular candidates. As a result, for each trigger, the list of its candidate arguments that we used include all and only those entity mentions, values, and time expressions within the same sentence. Training instances are created by pairing each trigger with each of its argument candidates. For instance, there are three triggers (bold words), and three entities (underlined words) in S8, which make up nine pairs of trigger and argument candidate to be classified, such as (**murders**, <u>France</u>) and (**assassination**, <u>Joe</u>).

S8: 六起 **谋杀案** 发生 在 <u>法国</u> ， 包括 <u>Bob</u> 的 **暗杀** 和 <u>Joe</u> 的 **杀害** 。
Six **murders** occurred in <u>France</u>, including the **assassination** of <u>Bob</u> and the **killing** of <u>Joe</u>.

The idea of convolution bidirectional LSTM model in trigger labeling is also suitable for argument labeling: given a typed trigger recognized by trigger labeling, together with a candidate argument, we utilize a RNN to obtain a sentence-level feature, concatenated with a CNN-extracted local lexical feature, to predict what role the candidate argument plays in the detected event. A special role `NONE` indicates the candidate does not play any role in the event.

Even though all candidate arguments are words, we need to encode trigger information into input features (see Section 5.1 for details), where language specific issues still have an impact on our prediction, therefore, the model we proposed here is a character-based model. We detail the differences between the convolution bidirectional LSTM models used in trigger labeling and argument labeling in the following sections.

### 5.1. Input

Given a training instance $(t, e)$ where $t$ is a typed trigger and $e$ is an ACE-annotated entity regarded as argument candidate, the input feature vector of each character includes not only character embeddings, but also information extracted from upstream trigger labeling subtask. Therefore, we propose four additional types of features to capture these important clues:

— Trigger position feature (TPF): as a trigger may be composed of several characters, we calculate the relative distance of a character to the nearest character in $t$.
— Trigger type feature (TTF): classified trigger type of a character, and a special type `NONE` for characters not in $t$.
— Candidate argument position feature (APF): the relative distance of the current character to the nearest character in $e$.

— Entity type feature (ETF): the entity type of the current character, and a special type `NONE` for characters not in an entity.

For example, in S8, if we want to predict the relationship between “暗杀” (assassination) and “Bob”, the TPF, TTF, APF, ETF of character “法” in the entity “法国” (France) is 8[4] , `NONE`, 4[5], `GPE`, respectively.

We transform these feature values into vectors by their corresponding lookup tables, and concatenate them with the original character vector, as the final input vector fed into BiLSTM and CNN layer. All additional feature embeddings are randomly initialized and optimized through back propagation.

### 5.2. BiLSTM

We continue to make use of GRU units to memory long sequences, and adapt the output vector of LSTM layer for argument labeling. Specifically, in the forward LSTM layer, we regard the hidden state of the last word $\overrightarrow{\mathrm{h}_n}$ as the sentence vector. As for the backward LSTM layer, hidden state of the first word $\overleftarrow{\mathrm{h}_1}$ is selected to be the reverse sentence vector. We concatenate these two vectors to form the output vector of a BiLSTM network, i.e. $\boldsymbol{B} = [\overrightarrow{\mathrm{h}_n}; \overleftarrow{\mathrm{h}_1}]$.

### 5.3. CNN

For the CNN layer, we take all words of the entire sentence as the context, rather than a shallow context window for each word under prediction. Formally, the matrix $\boldsymbol{c}$ fed into the convolution operator is given by:

$$\boldsymbol{c} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$$

where $\mathrm{x}_i$ is the input vector described in Section 5.1. After convolution and max pooling operation, we obtain a output vector $\boldsymbol{C}$ which is the representation of salient lexical features about the current pair of trigger and candidate argument.

Finally, we feed the concatenation of output vectors from two networks, $\boldsymbol{F} = [\boldsymbol{B}; \boldsymbol{C}]$, into a softmax classifier to estimate a probability distribution over all possible roles.

## 6. EVALUATION

**We evaluate our models on the ACE 2005 corpus to investigate the following questions: (1) can our neural network models achieve satisfactory performances without feature engineering; (2) can our models alleviate the language issue in Chinese event extraction?**

### 6.1. Experimental Setup

*Dataset and evaluation methodology.* We used the standard ACE 2005 corpus in our evaluation, which contains 633 Chinese documents. Unlike English, the ACE Chinese corpus does not have a recognized partition of documents for evaluation. Most of the previous work [Chen and Ji 2009; Feng et al. 2016] randomly select about 10% of the 633 documents as the test set. However, as reported by Chen and NG [2012] via 10-fold cross validation, performances achieved on different partitions vary considerably due to the small size of test sets. Therefore, cross validation may be a better choice when conducting experiments on the ACE 2005 corpus. In order to make accurate evaluation and save time, we perform 5-fold cross-validation and report our performances averaged over five folds.

[4]The relative distance between character “法” and character “暗” in the trigger word “暗杀” (assassination)
[5]The relative distance between character “法” and character “B” in the candidate argument “Bob”

Table III. Hyper-parameters for all experiments.

| Layer | Hyper-parameter | Trigger identification and classification | Argument identification and classification |
|---|---|---|---|
| Input | word embedding | 200 | – |
| | character embedding | 100 | 100 |
| | feature embedding[1] | – | 32 |
| Dropout | dropout rate | 0.5 | 0.5 |
| GRU | state size | 100 | 100 |
| CNN | context size | 7 | sentence length |
| | kernel sizes | [2, 4, 6] | [1, 2, 3, 4, 5] |
| | number of filters | [32, 32, 32] | [16, 16, 16, 16, 16] |

[1]Feature embeddings, including trigger position embedding, trigger type embedding, argument position embedding and entity type embedding, have the same size.

*Evaluation measures.* Similar to previous work, we evaluated our models in terms of $precision(P)$, $recall(R)$, and $F\!-\!measure(F)$ for each subtask. These performance metrics are computed according to the following standards of correctness for four subtasks:

— For trigger identification, a trigger is correctly identified if its offsets exactly match a reference trigger;
— For trigger classification, a trigger is correctly classified if its trigger type and offsets exactly match a reference trigger;
— For argument identification, an argument is correctly identified if its offset, related trigger type and trigger's offsets exactly match a reference argument;
— For argument classification, an argument is correctly classified if its offsets, role, related trigger type and trigger's offsets exactly match a reference argument.

### 6.2. Network Training

*Parameter initialization.* All matrix parameters (including weight matrices in neural network layers and transition matrix in the CRF layer) are randomly initialized with uniform samples from range $[-\sqrt{\frac{6}{row+col}}, +\sqrt{\frac{6}{row+col}}]$, where $row$ and $col$ are the number of rows and columns in the matrix [Glorot and Bengio 2010]. Bias vectors are initialized to zero.

*Pre-trained embeddings.* Initializing word vectors with those obtained from a neural language model is a popular method to improve performance in the absence of a large training set. [Collobert et al. 2011]. Both word and character embeddings are trained on over 261 thousand articles crawled from Chinese news website, using skip-gram word2vec model [Mikolov et al. 2013]. All embeddings are fine tuned during training.

*Optimization algorithm.* All models share a generic stochastic gradient descent (SGD) forward and backward training procedure. For all models presented, parameter optimization is performed using Adam [Kingma and Ba 2014] with gradient clipping [Pascanu et al. 2013]. We also apply the dropout method [Srivastava et al. 2014] on both the input and output vectors of all models to mitigate overfitting.

*Hyper-parameters.* In different stages of event extraction, we adopted different parameters. Table III summarizes the chosen hyper-parameters for all experiments.

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

Table IV shows the overall performance of all methods on the ACE2005 Chinese corpus. We select the following state-of-art methods for comparison.

Table IV. Overall system performance (%)

| Model | Trigger Identification | | | Trigger Classification | | | Argument Identification | | | Argument Classification | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F | P | R | F |
| Char-MEMM | **82.4** | 50.6 | 62.7 | **78.8** | 48.3 | 59.9 | **64.4** | 36.4 | 46.5 | **60.6** | 34.3 | 43.8 |
| Rich-L | 62.2 | **71.9** | 66.7 | 58.9 | **68.1** | 63.2 | 43.6 | **57.3** | 49.5 | 39.2 | **51.6** | 44.6 |
| HNN | 74.2 | 63.1 | **68.2** | 77.1 | 53.1 | 63.0 | – | | | | | |
| C-BiLSTM$_{word}$ | 65.2 | 62.7 | 63.9 | 61.7 | 59.4 | 60.5 | 50.3 | 50.2 | 50.2 | 43.4 | 43.3 | 43.3 |
| + Errata table | 68.3 | 65.8 | **66.9** | 62.7 | 60.9 | 61.6 | 49.2 | 52.1 | 50.6 | 42.4 | 44.8 | 43.5 |
| C-BiLSTM$_{char}$ | 63.7 | 63.2 | 63.4 | 60.1 | 59.6 | 59.9 | 49.8 | 52.8 | 51.2 | 42.4 | 45.0 | 43.6 |
| + Errata table | 67.0 | 66.4 | 66.7 | 60.7 | 62.6 | 61.5 | 49.8 | 53.0 | 51.2 | 42.8 | 45.5 | 44.1 |
| C-BiLSTM-CRF$_{char}$ | 66.1 | 67.9 | **66.9** | 62.5 | 64.2 | **63.3** | 50.8 | 54.1 | **52.4** | 44.0 | 46.8 | **45.3** |

— **Char-MEMM** [Chen and Ji 2009] is the first character-based method to handle the language specific issues, which trains a Maximum Entropy Markov Model to label each character with BIO tagging scheme.
— **Rich-L** [Chen and NG 2012] is a joint-learning, knowledge-rich approach that extends the union of the features employed by Char-MEMM and Li et al. [2012] with six groups of linguistic features, including character-based features and discourse consistency features, which is the feature-based state-of-art system.
— **HNN** [Feng et al. 2016] is a hybrid neural network model, which also incorporates both bidirectional LSTMs and convolutional neural networks to capture sentence and structure semantic information, and it achieves state-of-the-art performance in Chinese event detection.

Our reported results are averaged over five folds. We directly list the results of previous work from their paper. Although they all evaluated on ACE 2005 corpus, only Rich-L performs cross validation, while Char-MEMM and HNN select their test set randomly, therefore it is inappropriate to compare their performances directly.

*Comparison with Char-MEMM.* Char-MEMM concludes that *neighborly word features* are fairly applicable. They utilize the left word and right word of an entity to reduce spurious argument, which is similar to the motivation of our CNN-extracted lexical features. They obtain the highest precisions but lowest recalls on all subtasks. However, all neural network models outperforms in F1 on the first three subtasks. Because neural network methods can avoid the errors propagating from other NLP tools like dependency parsing and POS tagging, while capturing not only neighborly word features but also sentence-level information in the absence of feature engineering.

*Comparison with Rich-L.* Our best model is a character-based C-BiLSTM-CRF model, and it outperforms Rich-L on both trigger labeling and argument labeling. Note that in argument labeling, some of the arguments are not in the same sentence with their triggers. It is a bottleneck of our C-BiLSTM model, while Rich-L uses discourse-level features to deal with this problem. Under this unfavorable circumstance, our C-BiLSTM can still achieve a better performance against Rich-L, which depends on sophisticated human designed features.

*Comparison with HNN.* It is not appropriate for us to directly compare our results with this model, as it evaluates on randomly selected test documents. Our C-BiLSTM model assembles HNN model in the choice of neural networks, since both of them concatenate the output vector of BiLSTM and CNN, but their CNN parts and final outputs are different in two aspects. CNN in C-BiLSTM learns a representation over shallow windows for every word, rather than the entire sentence as in HNN model. We argue that C-BiLSTM can obtain more accurate contextual information, because different words may have different context, and, as a result, should have different contextual feature representations instead

Table V. Performance of different types of triggers by different models on trigger labeling.[2]

| Stage | Model | Regular Triggers[1] | | | Inside-word Triggers | | | Cross-word Triggers | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F | P | R | F |
| Trigger Identification | C-BiLSTM$_{word}$ | 64.1 | 72.4 | 68.0 | – | 0 | – | 63.0 | 44.6 | **52.3** |
| | C-BiLSTM$_{char}$ | 67.6 | 68.5 | 68.0 | 20.3 | 30.8 | 37.6 | 57.8 | 31.2 | 40.5 |
| | C-BiLSTM-CRF$_{char}$ | 67.6 | 72.4 | **70.0** | 51.1 | 40.7 | **45.3** | 51.3 | 38.9 | 44.3 |
| Trigger Classification | C-BiLSTM$_{word}$ | 60.6 | 68.4 | 64.2 | – | 0 | – | 57.8 | 40.9 | **47.9** |
| | C-BiLSTM$_{char}$ | 64.1 | 64.9 | 64.5 | 18.0 | 27.3 | 21.7 | 49.7 | 26.8 | 34.9 |
| | C-BiLSTM-CRF$_{char}$ | 64.0 | 68.5 | **66.2** | 46.7 | 37.2 | **41.4** | 47.8 | 36.2 | 41.2 |

[1]We regard regular triggers as triggers composed of exactly one word.
[2]We use NLPIR to perform word segmentation, and the number of regular triggers, inside-word triggers and cross-word triggers are 2863, 172 and 298, respectively.

Table VI. Error analysis: examples of triggers mislabeled by character-based C-BiLSTM.

| Sentence | Triggers | Labels |
|---|---|---|
| S9: **走访** 相关 人员 以后，... <br> After **visiting** to relevant staff, ... | 走访 <br> visiting | expect: `BI` <br> output: `BO` |
| S10: **贺电** 全文 如下: <br> There is the full **congratulatory message** | 贺电 <br> congratulatory message | expect: `BI` <br> output: `OB` |
| S11: 小偷 被 逼 进 **死 胡同**。 <br> The thief was chased into a **dead end**. | 死 胡同 <br> dead end | expect: `O OO` <br> output: `B OO` |

of sharing one sentence level representation. Moreover, HNN treats event detection as a classification task that determine whether each word is an event trigger, so it can not identify neither inside-word nor cross-word triggers.

## 6.4. Word-based C-BiLSTM vs. Character-based C-BiLSTM

As shown in Table IV, when applying the same network architecture to the same subtask, word-based methods always have higher precisions while character-based methods always have higher recalls.

We then take a further step to investigate their impacts on different kinds of triggers. As shown in Table V that:

(1) Both character-based and word-based methods achieve similar performances in regular trigger identification, and character-based method performs slightly better in trigger classification, 0.3% higher than word-based method in F-measure;
(2) Word-based methods can not label inside-word triggers, while character-based methods can handle this issue, which brings higher overall recall;
(3) It is more difficult for character-based method to correctly identify cross-word triggers. As there are more cross-word triggers than inside-word triggers in dataset, the overall F-measure of word-based methods is slightly higher.

We find several reasons that cause the lower precision of character-based method:

(1) Character-based method needs to learn the basic patterns of word segmentation by itself. 8.7% triggers identified by character-based method are partially mislabeled into inside-word triggers, like triggers in S9 and S10, which leads to a low level precision for inside-word triggers labeling.
(2) Word embedding brings richer semantic information than character embedding. Take S11 in Table VI as an example, characters "胡" and "同" do not have any meaning related to the formed word "胡同" (alleyway), while this word strongly suggests that "死" (dead) is not a trigger within the context "死胡同" (dead end). Given the more accurate embedding of surrounding context, word-based networks can understand the meaning of the center word thus lead to better disambiguation.

Table VII. Comparison of performances of BiLSTM, CNN and C-BiLSTM model. Performances of argument labeling are evaluated on gold standard event triggers.

| Model | Trigger Identification | | | Trigger Classification | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| BiLSTM$_{word}$ | 66.0 | 59.7 | 62.7 | 59.3 | 60.1 | 59.5 |
| CNN$_{word}$ | 67.6 | 58.2 | 62.5 | 63.8 | 54.9 | 59.0 |
| C-BiLSTM$_{word}$ | 65.2 | 62.8 | 63.9 | 61.7 | 59.4 | 60.5 |

| Model | Argument Identification | | | Argument Classification | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| BiLSTM$_{char}$ | 68.7 | 73.4 | 71.0 | 60.4 | 64.5 | 62.4 |
| CNN$_{char}$ | 67.1 | 66.9 | 66.9 | 60.4 | 60.2 | 60.3 |
| C-BiLSTM$_{char}$ | 69.5 | 74.4 | 71.9 | 61.3 | 65.6 | 63.4 |

Table VIII. Percentages of ambiguous words classified correctly. All networks listed in this table are word-based.

| | BiLSTM | CNN | C-BiLSTM |
|---|---|---|---|
| ambiguous word classification (%) | 61.4 | 61.7 | 64.1 |

(3) GRU units in character-based method needs to maintain clues for longer sequences, as 1.6 times longer than the average length of word sequences. In trigger classification, when we evaluate on sentences containing more than 100 characters, F-measure of character-based method is 57.3%, while word-based method can achieve 59.8%, suggesting that GRU units lose some information for longer sequences.

### 6.5. Neural Network Architectures

To compare the effectiveness of different neural network components of C-BiLSTM, we detect events by using BiLSTM and CNN separately. As Table VII shows, BiLSTM is more efficient than CNN, especially in argument labeling. And the combined C-BiLSTM model outperforms all other models on all subtasks. This observation demonstrates that **both of the BiLSTM and CNN models are important for event extraction, and have certain complementarity with each other.**

Another challenge here is that some words can trigger different types of events, given different contexts. Take S12 and S13 for example, the word “释放” (released) in S12 means releasing gas and expresses an *Attack* event, while in S13, it means setting somebody free. These ambiguous words account for 36.8 percent of all triggers. We further evaluate the capacity of each network on ambiguous triggers.

S12: 部队 向 抗议者 **释放** 了 催泪弹 。
The troops **released** tear gas to the protesters.
[type: *attack*]

S13: 他 在 服刑 五 年 后 从 狱中 **释放** 出来 。
He was **released** from prison after serving a sentence of five years.
[type: *release-parole*]

Table VIII provides evidences that, benefited from the salient local contextual features, CNN can performs better than BiLSTM on ambiguous triggers, while BiLSTM-extracted sentence-level information also help reduce some errors caused by ambiguous triggers. C-BiLSTM model that combines both levels of features, achieve highest precision as we expected.

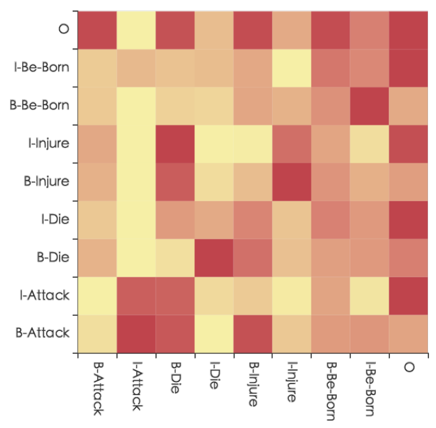


Fig. 4. A visualization of a submatrix of transition score in CRF layer. The horizontal axis represents label of current word and the vertical axis represents label of previous word. The darker the color, the higher the score.

## 6.6. C-BiLSTM vs. C-BiLSTM-CRF

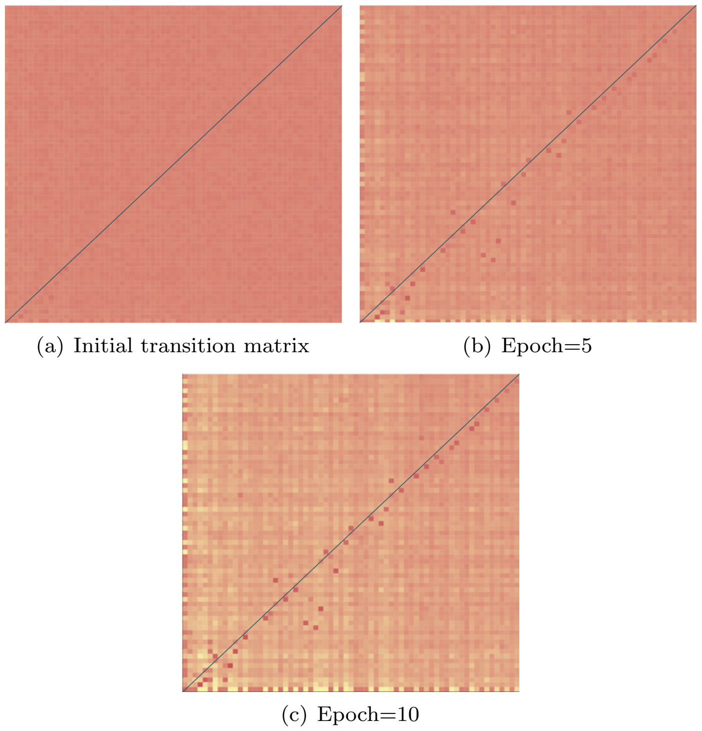
In Table IV, we can see that our character-based convolution BiLSTM-CRF model significantly outperforms other state-of-the-art methods. We list the performance on different types of triggers by C-BiLSTM-CRF and C-BiLSTM model in Table V. Compared with C-BiLSTM models, instead of modeling tagging decisions independently, C-BiLSTM-CRF model takes into account the correlations between labels, which contributes to a 2% increase of F1 in regular trigger identification, and a 7.7% increase of F1 in inside-word trigger identification. We select a subset of labels and generate a heat map in Figure 4 from the transition scores between each label. We find that the better performance of C-BiLSTM-CRF model can be further explained by the following two reasons:

(1) A CRF layer characterizes the constraints nicely: for a specific event type `Type`, `I-Type` can only follow `B-Type` or `I-Type`. In other words, if the current label is `I-Type`, the transition scores for previous labels, except `B-Type` and `I-Type`, are invalid and supposed to be very low. For example, as the forth column shows, going from `B-Die` or `I-Die` to `I-Die` is encouraged, while other transitions, like `B-Attack` to `I-Die` or `I-Injure` to `I-Die` are discouraged.
(2) A CRF layer models the co-occurrences of several event types. For example, an *Attack* event is usually followed by an *Injure* event or a *Die* event. As a result, in the first row, the transition scores of going from `B-Attack` to `B-Injure` or `B-Die` are much higher than other labels start with `B-`, such as `B-Be-born`.

Figure 5(c) shows the whole transition matrix after training. The darkest color grids always appear right below the diagonal, which suggests that for a given label `B-type`, its most likely following label is `I-Type`. As we can conclude from Figure 5, during training, CRF layer gradually learns a score for going from one label to another label and encourages valid paths of labels, while discouraging other invalid or unusual paths.

## 7. CONCLUSION

In this paper, we propose a novel convolution bidirectional LSTM-CRF model on Chinese event extraction task. Our model departs from the inherent characteristic of Chinese language, formulates the event detection task as a sequence labeling fashion, and features both bidirectional LSTM and CNN to capture both sentence-level and lexical features from raw text. Experimental results show that without human-designed features and external


(a) Initial transition matrix (b) Epoch=5

(c) Epoch=10

Fig. 5. Transition score matrix in CRF layer during training, visualized with heatmaps. Axes have the same meaning as Figure 4: the horizontal axis represents label of current word while the vertical axis represents label of previous word. Labels in both axes are arranged by their types in the same order as Figure 4, i.e., `O, B-type1, I-type1, B-type2, I-type2, B-type3, ...`

resources, our neural network method can achieve state-of-the-art performances on ACE 2005 datasets. In the future, we will explore more global evidences, from both discourse level and cross document level, to further alleviate the language specific issues, such as the zero pronoun phenomenon in Chinese. Currently, many event extraction methods utilize the type consistency between triggers or arguments as a post-processing step or a set of consistency-check features to prune the extraction results, which should be treated jointly in the extraction models, hopefully in a neural network fashion. And, we will also attempt to model the whole event extraction task in an end-to-end paradigm to further jointly deal with trigger and argument labeling, avoiding feature engineering or other resources.

## REFERENCES

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*. Association for Computational Linguistics, 1–8.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

Chen Chen and V Incent NG. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *In COLING*. Citeseer.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Vol. 1. 167–176.

Zheng Chen and Heng Ji. 2009. Language specific issue and feature exploration in Chinese event extraction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North Amer-*

*ican Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers.* Association for Computational Linguistics, 209–212.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

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

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The Automatic Content Extraction (ACE) Program-Tasks, Data, and Evaluation.. In *LREC*, Vol. 2. 1.

Xiaocheng Feng, Lifu Huang, Duyu Tang, Bing Qin, Heng Ji, and Ting Liu. 2016. A Language-Independent Neural Network for Event Detection. In *The 54th Annual Meeting of the Association for Computational Linguistics.* 66.

Reza Ghaeini, Xiaoli Z Fern, Liang Huang, and Prasad Tadepalli. 2016. Event nugget detection with forward-backward recurrent neural networks. In *The 54th Annual Meeting of the Association for Computational Linguistics.* 369.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks.. In *Aistats*, Vol. 9. 249–256.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE, 6645–6649.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18, 5 (2005), 602–610.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers.* Association for Computational Linguistics, 369–372.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1.* Association for Computational Linguistics, 1127–1136.

Ruihong Huang and Ellen Riloff. 2012. Modeling Textual Cohesion for Event Extraction.. In *AAAI*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* (2015).

Heng Ji and Ralph Grishman. 2008. Refining Event Extraction through Cross-Document Inference.. In *ACL*. 254–262.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, Vol. 1. 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* (2016).

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. 2012. Employing compositional semantics and discourse consistency in Chinese event extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.* Association for Computational Linguistics, 1006–1016.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint Event Extraction via Structured Prediction with Global Features.. In *ACL (1)*. 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.* Association for Computational Linguistics, 789–797.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, Vol. 2. 365–371.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28 (2013), 1310–1318.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, 151–160.

Steven J Phillips, Robert P Anderson, and Robert E Schapire. 2006. Maximum entropy modeling of species geographic distributions. *Ecological modelling* 190, 3 (2006), 231–259.

Jakub Piskorski and Roman Yangarber. 2013. *Information Extraction: Past, Present and Future.*

Lawrence R Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77, 2 (1989), 257–286.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

Johan AK Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural processing letters* 9, 3 (1999), 293–300.