# Learning to Extract Events without Human-annotated Text


## Abstract

Existing event extraction systems are supervised and often learned from expert-annotated datasets, such as ACE and ERE event extraction program. However, constructing these high-quality corpora is costly, and manually annotated dataset are limited in size and coverage of event types, which makes models learned on these datasets hard to generalize. Inspired by some Freebase schemas which share similar structures with ACE event templates, we investigate the following problems in this paper: can we generate a feasible dataset for event extraction with Freebase automatically and is it possible to extract events on this dataset. We first propose four hypotheses based on our observation and produce our dataset accordingly. Then, we design a neural network model with ILP-based post inference committing to handling two challenging problems in event extraction: multi-type events and multi-word arguments. Finally, manual evaluation demonstrates that the data we generated are feasible, and experimental results of both manual and automatic evaluation prove the effectiveness of our proposed model.


## 1 Introduction

Automatically extracting events from natural text remains a challenging task in information extraction. Among diverse types of event extraction systems, the event extraction task proposed by Automatic Content Extraction (ACE) [Doddington *et al.*, 2004] program is the most popular corpus for testing new event extraction algorithms. The ACE program defines the two terminologies for event extraction: **trigger** and **argument**. The former one is the word that most clearly expresses the occurrence of an event. The latter one is a named entity that serves as a participant or attribute with a specific role in an event.

However, constructing this corpus is expensive. First, expert linguists are required to summarize a large amount of text to elaborately design templates about potential arguments for each event type. Second, rules should be explicitly stated to guide annotators. In spite of detailed annotation guidelines, there is still some disagreement between human annotators about what should (not) be regarded as triggers. For example, can a prepositional phrase trigger an event, like "in prison" triggers an *arrest* event? Or can a portion of a word be a trigger, like "ex" in "ex-husband" triggers a *divorce* event? Besides, ACE event extraction systems remain two major limitations: first, they focus on simplified scenario of single-token trigger labeling; second, they assume that an event has only one type.

The aforementioned drawbacks of ACE event extraction systems motivate us to (1) investigate the feasibility of automatic data generation, and (2) address the event extraction task in a more realistic scenario where annotated arguments are almost phrases, and some events have more than one types.

First, we draw the inspiration from our observations on knowledge base (KB). A knowledge base stores complex structured information, and some of which shares a highly similar structure with event templates defined in ACE. Thus, any sentence that contains some participants and attributes in a particular KB entry is likely to imply an event in some way. On the other hand, recent studies [Mintz *et al.*, 2009; Zeng *et al.*, 2015] have demonstrated the effectiveness of KB as distant supervision for binary relation extraction, which is another important task in information extraction. However, there is two major challenges when leveraging KB to event extraction: first, relations between an event and its arguments are more complex than binary relations. They can be represented as $\langle event\_type, argument_1, \ldots, argument_n\rangle$, which are n-ary relations with various numbers of entities. Second, there is no trigger information in any existing knowledge base. Therefore, we explore datasets generated based on different hypotheses about arguments to avoid the noise caused by the absence of triggers, and select the one that best fits in both quality and quantity (see Section 2.2). Among these hypotheses, the vital one is that, for a particular event type, there are some relevant arguments, called **key arguments**, which together can trigger an event of the corresponding type and distinguish it with other event types. In this paper, we utilize Freebase as the knowledge base. And we choose Wikipedia articles as texts for data generation, according to Mintz et al. [2009], because a major source of Freebase is the tabular data from Wikipedia, making it a natural fit with Freebase. Figure 1 illustrates two examples of sentences annotated by our algorithm.

Second, unlike previous ACE event extraction systems [Ahn, 2006; Li *et al.*, 2013; Chen *et al.*, 2015; Nguyen *et al.*, 2016], which regard event extraction as a classification problem, we split event extraction into two sequence labeling subtasks, namely event detection and argument detection. Following the fashion in other sequence labeling tasks, like POS tagging and NER [Huang *et al.*, 2015; Lample *et al.*, 2016], we utilize a LSTM-CRF model to label key arguments and non-key arguments in the sentences separately. However, based on the above hypothesis, LSTM-CRF is not a flawless solution, as the structure of an event is not sequential and there are strong dependencies between key arguments with respect to the same event type. Therefore, we reformulate the hypotheses as constraints and apply linear integer programming on the output scores produced by LSTM-CRF to get the global optimization. After this post inference, to tackle with the multiple events and multi-type events issue, we propose a heuristic method to iteratively generate multiple labeling sequences. Finally, we conduct both manual and automatic evaluation for the detected events.

In this paper, we first intend to employ Freebase as distant supervision to label event structures from text without the need of human annotator. To address the problem caused by the absence of trigger information in Freebase, we propose four hypotheses about the relations between event arguments and Freebase instances. Then we propose a LSTM-CRF model with post inference to extract general events and multi-type events on the generated dataset, which is demonstrated effective by both manual and automatic evaluation.

The rest of the paper is organized as follows. Section 2 presents our data generation methods, and Section 3 introduces our models of extracting events. Section 4 provides the experimental results and analyses. Section 5 discusses related work. Section 6 concludes this paper.

## 2 Dataset Preparation

We employ Freebase to automatically annotate text in Wikipedia. We regard a sentence as a positive one when it suggests an occurrence of event, or otherwise a negative sentence. For example, S1 and S2 are positive sentences and their arguments are in italics and underlined, while S3 and S4 are negative sentences. The event structures of S1 and S2 are illustrated in Figure 1.

**S1**: *Remedy Corp* was sold to *BMC Software* as the *Service Management Business* Unit in *2004*.

**S2**: *Microsoft* spent $6.3 billion buying online display advertising company *aQuantive* in *2007*.

**S3**: Microsoft hopes aQuantive's Brian McAndrews can outfox Google.

**S4**: On April 29th, Elizabeth II and Prince Philip witnessed the marriage of Prince William.

### 2.1 Freebase

Freebase[Bollacker *et al.*, 2008] is a collaborative structured knowledge base which be divided into three layers: *domain*, *type* and *instance*. *Instances* are entries in Freebase, and are related to real-world entities like people, places and things. *Types* are different perspectives of *instances*. ***Compound Value Type*** (CVT) is a special type in Freebase to represent complex structured data where each instance consists of multiple ***properties***. Some of the CVT schemas are highly alike to event structures where CVT properties can be treat as event arguments. For example, in Figure 1, *business.acquisition* is a CVT whose properties are *company_acquired*, *acquiring_company*, *date* and *divisions_formed*. These properties can also be used to represent participants and attributes in events extracted from S1 and S2.

We use the Freebase version of Berant et al. [2013], containing 1010 CVTs. After filtering out CVTs that describe the structures of the Freebase or are irrelevant to event extraction (e.g., *food.recipe_ingredient*), we select 24 types of CVTs with around 280 million instances.

Instances of *business.acquisition* in Freebase

| id \ property | company_acquired | acquiring_company | date | divisions_formed |
|---|---|---|---|---|
| m.07bh4j7 | Remedy Corp | BMC Software | 2004 | Service Management Business Unit |
| m.05nb3y7 | aQuantive | Microsoft | 2007 | NONE |

Data generation

Event structures in our dataset

| Wiki text | **S1:** *Remedy Corp* was sold to *BMC Software* as the *Service Management Business Unit* in *2004*. | | | |
|---|---|---|---|---|
| Event type | *business.acquisition* | | | |
| Arguments | company_acquired | acquiring_company | date | divisions_formed |
| | *Remedy Corp* | *BMC Software* | *2004* | *Service Management Business Unit* |

| Wiki text | **S2:** *Microsoft* spent $6.3 billion buying online display advertising company *aQuantive* in *2007*. | | | |
|---|---|---|---|---|
| Event type | *business.acquisition* | | | |
| Arguments | acuqiring_company | company_acuqired | date | divisions_formed |
| | *Microsoft* | *aQuantive* | *2007* | —— |

Figure 1: Examples of CVT instances in Freebase, and labeled sentences in our dataset. *Company_acuqired*, *acquiring_company* and *date* are key arguments in *business.acquisition*.

### 2.2 Data Generation

The annotation strategy is based on four hypotheses.

**H1: Positive sentences should contain all properties**

This hypothesis indicates that if a sentence has all properties of a CVT, it is more likely to be a positive sentence. We regard the CVT as event type and extract words and phrases that match the properties of a CVT instance as involved arguments. For example, S1 contains all the properties of instance $m.07bh4j7$ whose type is *business.acquisition*, thus we consider S1 as a positive sentence which implies an event about *business.acquisition*, and *BMC Software*, *Remedy Corp*, *Service Management Business Unit* and *2004* should be labeled as the argument that plays the role of *acquiring_company*, *company_acquired*, *divisions_formed*, *date*, respectively.

However, in practice, we realize that *H1* is too strict that excludes a great many positive sentences like S2. So we put forward the second hypothesis.

**H2: Positive sentences should contain all key properties**

This hypothesis is an extension of *H1*, which relaxes "all properties" constraint to "key properties". We define the CVT property that plays an important part in its CVT structure that helps to distinguish with other CVT as **key property**.

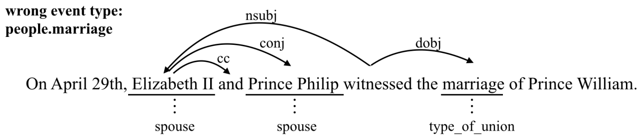


Figure 2: An illustration of dependency parse tree of S4.

And **key argument** is the word or phrase that matches a key property of a CVT instance. For example, *company_acquired* and *acquiring_company* are the key properties of CVT *business.acquisition*, and with this relaxation, positive sentences like S2 need not contain all properties, but only key properties instead.

The importance of a property $pro$ (e.g., *date*) to its CVT $cvt$ (e.g., *business.acquisition*) can be defined as follows:

$$degree_{cvt,pro} = log\frac{count(cvt, pro)}{count(cvt) \times count(pro)} \qquad (1)$$

where $count(cvt)$ is the number of all $cvt$ instances, $count(pro)$ is the number of $pro$ among all CVTs, and $count(cvt, pro)$ is the number of $cvt$ instances that contain the property $pro$.

**H3: Key properties should include time property**
This hypothesis strengthens *H2* by counting time property in key properties. We discover that for many CVTs, their key properties do not take into account time property. However, in fact, ignoring time property will produce a large number of negative sentences like S3. This sentence does not express an explicit event about *business.accquisition* while contain all key properties of an instance, resulting in mistaking *Microsoft* for *acquiring_company*, and *aQuantive* for *company_acquired*. By adding *date* to the set of key properties, S3 will be filtered. Therefore, in *H3*, we choose the property which achieves highest relevance degree among all time properties as a supplementary key property.

**H4: Positive sentences should contain key properties with close syntactic distance**
We introduce another factor, syntactic distance, to annotate positive sentences. Intuitively, two arguments take participant in the same event are likely to be close in syntactic structures. This factor is feasible to eliminate negative sentences, such as S4, which satisfies *H2*. The syntactic distance can be measured by the distance of two words in dependency parsing tree. We set the maximum distance between two key arguments as 2, denoting that, for a candidate sentence, if a pair of key arguments within it violates this constraint, it is supposed to be negative. Given the dependency parsing tree in Figure 2, S4 is negative because the distance between *Prince Philip* and *marriage* is 3.

We conduct a manual evaluation on the quantity and quality of datasets generated by different hypotheses (see Section 4.2), and utilize the combinations of hypothesis *H3* and *H4* as the final strategy to data generation.

# 3 Model

Unlike existing event extraction work in which triggers are the key evidence to identify event and classify different event types, in the absence of human-labeled triggers, we argue that **key arguments** can play the same role as triggers. Consequently, we can treat event extraction as a pipeline of two primary subtasks, namely event detection and argument detection.

**Event detection**: to identify key arguments in the sentence. And if a sentence contains **all** key arguments of a specific event type, it is considered to imply an event of the corresponding type. For example, in S1, *Remedy Corp*, *BMC Software*, and *2004* should be identified as *business.acquisition.company_acquired*, *business.acquisition.acquiring_company*, and *business.acquisition.date* respectively. As a result, S1 should be labeled as expressing an event about *business.acquisition*.

**Argument detection**: to identify other non-key arguments for each event in the sentence. For the *business.acquisition* event in S1, *Service Management Business Unit* should be identified as *business.acquisition.divisions_formed*.

## 3.1 Event Detection

Before presenting our model, we need to come up with the solution to multi-words arguments. Then we introduce the components in our LSTM-CRF-ILP$_{multi}$ model one-by-one from bottom to top.

**Tagging scheme** 68 percent of arguments in our dataset consist of more than one words. To address this issue, we model each subtask as a sequence labeling task rather than a word classification task. Each word in the given sentence is tagged in the `BIO` scheme, where each token is labeled as `B-role` if it is the beginning of an event argument with its corresponding role `role`, or `I-role` if it is inside an argument, or `O` otherwise.

**LSTM** Long Short-Term Memory Network (LSTM) [Hochreiter and Schmidhuber, 1997] is a natural model for sequence labeling task, which maintains a memory based on historical contextual information. Formally, given a sentence $\boldsymbol{w} = \{w_1, w_2, \ldots, w_n\}$ of length $n$, we use $\mathbf{x}_t$ to represent feature vector (e.g. word embedding) corresponding to the $t$-th word $w_t$. At each time step $t$, a LSTM unit takes $\mathbf{x}_t$ as input and computes the output vector $\mathbf{h}_t$ through several multiplicative gates. Then output vector is fed into a softmax layer to estimate a probability distribution over all possible labels.

**CRF** A straightforward way is to choose the label which obtains maximum probability by LSTM as the prediction for each word. However, this independent labeling strategy is limited especially when there are strong dependencies and constraints between arguments. To model the correlations between labels, we introduce a CRF layer into the output of LSTM, which is widely used and proved to be effective in various sequence labeling tasks, such as POS tagging and NER [Collobert *et al.*, 2011; Huang *et al.*, 2015; Lample *et al.*, 2016].

We consider $\mathbf{P}$ to be a matrix of confidence scores output by LSTM network, and the element $\mathbf{P}_{i,j}$ of the matrix denotes the probability of the label $j$ for the $i$-th word in a sentence.

The CRF layer has a transition score matrix $\mathbf{A}$ as parameter, where $\mathbf{A}_{i,j}$ represents the score of a transition from label $i$ to label $j$. The score of a sentence $\boldsymbol{w}$ along with a path of labels $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ is measured by the sum of neural network outputs and transition scores:

$$score(\boldsymbol{w}, \boldsymbol{y}) = \sum_{i=0}^{n} \mathbf{P}_{i,y_i} + \sum_{i=1}^{n} \mathbf{A}_{y_i,y_{i+1}}, \tag{2}$$

During test, given a sentence $\boldsymbol{w}$, we adopt the Viterbi algorithm [Rabiner, 1989] to find the optimal label sequence with the maximum score among all possible label sequences.

**ILP-based Post Inference** Event detection is a structure prediction problem, while the output sequences of LSTM-CRF not necessarily satisfy the structural constraints. Specifically, regardless of how many key arguments are identified correctly by LSTM-CRF, if there is one key argument missing, detection of its corresponding event is failed. To amend this flaw, we apply Integer Linear Programming (ILP) with respect to the scores given by the above LSTM-CRF model to generate the final labeling sequence.

Formally, let $\mathcal{L}$ be the set of possible argument labels. For each word $w_i$ in the sentence $\boldsymbol{w}$ and a pair of labels $\langle l, l' \rangle \in \mathcal{L} \times \mathcal{L}$, we create a binary variable $v_{i,l,l'} \in \{0, 1\}$, denoting whether or not the $i$-th word $w_i$ is tagged as label $l$ and its following word $w_{i+1}$ is tagged as label $l'$ at the same time. The objective of ILP is to maximize the overall score of the variables,

$$\sum\nolimits_{i,l,l'} v_{i,l,l'} * (\mathbf{P}_{i,l} + \mathbf{A}_{l,l'}).$$

We consider the following four constraints:

**C1**: Each word should be and only be labeled with one label, i.e.:

$$\sum\nolimits_{l,l'} v_{i,l,l'} = 1 \tag{3}$$

**C2**: If the value of $v_{i,l,l'}$ is 1, then there has to be a label $l^*$ which makes $v_{i+1,l',l^*}$ equal to 1, i.e.:

$$v_{i,l,l'} = \sum\nolimits_{l^*} v_{i+1,l',l^*} \tag{4}$$

**C3**: If current label is `I-arg`, then its previous label must be `B-arg`, i.e.:

$$v_{i,\texttt{I-arg},l'} = v_{i-1,\texttt{B-arg},\texttt{I-arg}} \tag{5}$$

**C4**: For a specific event type, its key arguments should be co-occurred, or none of them should appear in the resulting sequence. For any pair of key arguments $arg_1$ and $arg_2$ with respect to the same event type, the variables related to them are subject to:

$$\sum\nolimits_{i,l'} v_{i,\texttt{B-arg}_1,l'} \leq n * \sum\nolimits_{j,l^*} v_{j,\texttt{B-arg}_2,l^*} \tag{6}$$

where $n$ is the length of the sentence.

In order to address the multi-type event issue, we allow ILP solver to output multiple sequences iteratively. Formally, let $\boldsymbol{s}^t = \{l_1^t, l_2^t, \ldots, l_n^t\}$ be the sequence produced at iteration $t$. During each iteration $t$, we eliminate $\{\boldsymbol{s}^1, \ldots, \boldsymbol{s}^{t-1}\}$ from the solution space to obtain the next optimal sequences $\boldsymbol{s}^t$. We repeat the above procedure with these constraints through ILP, until the difference between objective value of $\boldsymbol{s}^1$ and $\boldsymbol{s}^T$ is greater than a threshold $\lambda$, and consider all sequences $\{\boldsymbol{s}^1, \boldsymbol{s}^2, \ldots, \boldsymbol{s}^{T-1}\}$ as the optimized set of label sequences. In our experiment, Gurobi [Gurobi Optimization, 2016] is chosen as our ILP problem solver and $\lambda = 0.05 \times n$.

### 3.2 Argument Detection

After event detection, a sentence will be classified into different event types, and labeled with its corresponding key arguments. Next step is argument detection which aims to identify the remaining arguments (non-key arguments) in the sentences.

We simply adapt the same architecture as the LSTM-CRF model (see Section 3.1) for argument detection, where we encode the label (output of event detection) of each word into a key-argument feature vector througth a look-up table, and concatenate it with the origin word embedding as the input vector to LSTM. We do not need post inference in this task.

## 4 Experiments

### 4.1 Experimental Setup

**Dataset and Evaluation Methodology** We use the November 20th, 2016 English Wikipedia dump, and generate 7180 sentences, containing 7376 events and 25840 arguments as corpus. We then randomly select 4800 sentences for training and 1180 sentences as test set, and the remained 1200 sentences for validation. We conducted both automatic evaluation and manual evaluation in the experiments. Specifically, we first manually evaluate the reliability of our test set. Next, we regard the noisy rule-generated data as gold standard and evaluate our model automatically. Finally, we manually evaluate a subset of events detected by our model and analysis the difference with results in automatic evaluation.

**Evaluation Measures** 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 event type classification, an event is correctly classified if its reference sentence contains all key arguments of this event type.
- For argument detection, an argument is correctly detected if its offsets, role, and related event type exactly match any reference argument within the same sentence.
- For event detection, an event is correctly detected if its type and all its key arguments match a reference event within the same sentence.

**Training** In our experiments, all hyperparameters are tuned on the development set. In event detection, we set the size of word embedding to 200, the size of LSTM layer to 100. In argument detection, we use the same size of word embedding, while the size of LSTM layer is 150, and the size of key argument embedding is 50. During training, we apply the generic stochastic gradient descent [Bottou, 2010] with a dropout rate as 0.5 on both the input and output layers to mitigate overfitting. Word embeddings are pretrained using skip-gram word2vec model [Mikolov *et al.*, 2013] over the whole Wikipedia dump and fine tuned during training.

## 4.2 Dataset Evaluation

For comparison, we evaluate five datasets that utilize different hypotheses to generate positive sentences from Wikipedia. We randomly select 100 sentences in each dataset, and annotators are asked to determine whether these sentences imply events.

| Hypothesis | H1 | H2 | H2+H4 | H3 | H3+H4 |
|---|---|---|---|---|---|
| Instances | 0.3M | 3.6M | 3.6M | 1.3M | 1.3M |
| Dataset | 203 | 108K | 12K | 9241 | 7180 |
| Event type | 9 | 24 | 24 | 24 | 24 |
| Correct (%) | 98 | 22 | 37 | 81 | 89 |

Table 1: Statistic of generated dataset with different hypotheses. Instances denotes the number of CVT instances that can be used for each hypothesis. Dataset is the number of generated sentences. Event type indicates the number of different CVT types in each dataset. Correct represents the percentage of sentences which account as stating events explicitly.

As shown in Table 1, the strictest hypothesis, *H1*, guarantees the quality and confidence of generated data, while there are merely 30K CVT instances that contain all properties of their corresponding CVT types. And by utilizing these instances, we can only obtain 203 sentences and cover 9 types of events, which is quite insufficient for further training. *H2* is looser than *H1*, though it expands the resulting dataset, it produce a large number of noisy sentences. This side effect demonstrates that *H2* is inappropriate to be used as a soft constraint. Compared with *H2*, the significant improvement in the quality of sentences generated by *H3* proves that CVT properties referring time information are critical to data generation. Among all hypotheses, finally, data obtained by a combination of *H3* and *H4* achieves highest precision, which demonstrates that our hypothesis *H3* and *H4* are feasible and it is an effective way to generate reliable data automatically.

## 4.3 Baselines

To investigate the effectiveness of our proposed model, we develop three baseline extraction systems for comparison, including traditional feature-based methods and neural network models. For neural network method, we train a long short-term memory network that takes word embeddings as the input, and simply learns a probability distribution over all possible labels. For feature-based methods, we apply Conditional Random Field [Lafferty *et al.*, 2001] and Maximum Entropy [Berger *et al.*, 1996] to explore a variety of elaborate features (lexical, syntactic, entity information features) modified from state-of-art feature-based ACE event extraction system [Li *et al.*, 2013]. It is worth mentioning that during argument detection, we add the label of each word output by event detection as a supplementary feature.

We derive these features using Stanford CoreNLP [Manning *et al.*, 2014], and apply the implementation from the CRF++ toolkit [Kudo, 2005] and Le Zhang [1] to train CRF and max entropy classifiers, respectively.

## 4.4 Automatic Evaluations

As we can summarize from Table 2, traditional feature-based models are inefficient in both event detection and argument detection. Some of features they utilized, such as dependency features, suffer much from the absence of trigger. Although they can achieve high precisions, they can only extract a limited number of events, resulting in low recalls. Neural-network-based methods performs much better than feature-based models, because they can make better use of word semantic features, especially, LSTM can capture longer range dependencies and richer contextual information instead of neighborly word features. Moreover, neural-network-based methods can avoid errors propagating from other NLP preprocessing tools like POS tagging and NER.

**Effect of CRF Layer** Every model which has a CRF layer over its LSTM output layer is superior to the one with a simple LSTM layer. Compared with LSTM model, LSTM-CRF achieves higher precisions and recalls in all subtasks by significantly reducing the invalid labeling sequences (e.g., `I-arg` appears right after `O`). During prediction, instead of tagging each token independently, LSTM-CRF takes into account the constraints between neighbor labels, and increases the cooccurrences of key arguments with regard to the same event type in some way.

**Effect of Post Inference** As shown in Table 2, post inference based on ILP considerably improve the overall system performance, especially in event classification. With the help of constraint **C4**, some dubious key arguments can be inferred through other key arguments from their contexts. Compared with LSTM-CRF, LSTM-CRF-ILP$_1$ produces a gain of 7.4 in event classification, 1.8 in event detection, and 4.6 in argument detection, with respect to the F1 score.

We further investigate the effect of our heuristic method, LSTM-CRF-ILP$_{multi}$ which deals with the multi-event sentence issue. Compared with other models, LSTM-CRF-ILP$_{multi}$ selects several labeling sequences according to their objective value, and extract a number of events with comparable confidences from a sentence. As we can see from Table 2, this strategy may detect multiple events for a sentence, contributing to the increase of recalls, and F1 scores at the spent of a little drop of precisions.

## 4.5 Manual Evaluations

**Manual Annotations** We randomly sample 150 unlabeled sentences from test data set. Annotators are asked to fully annotate the events and arguments to each sentence following steps. First, determine whether a given sentence is positive or negative, and assign event types to positive sentences. Next, label all related arguments and their roles according to the types of events in the positive sentences. To make the annotation more credible, each sentence is independently annotated

[1] `https://github.com/lzhang10/maxent`

| Model | Event Classification | | | Argument Detection | | | Event Detection | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| CRF | 96.8 | 9.93 | 18.0 | 64.8 | 6.54 | 11.9 | 29.8 | 3.06 | 5.55 |
| MaxEnt | **97.9** | 11.4 | 20.3 | 64.5 | 7.28 | 13.1 | 29.3 | 3.40 | 6.08 |
| LSTM | 97.2 | 62.4 | 75.1 | 77.1 | 53.9 | 63.5 | 51.0 | 32.8 | 39.9 |
| LSTM-CRF | 97.3 | 67.2 | 79.5 | **78.0** | 60.2 | 68.0 | **54.4** | 37.6 | 44.4 |
| LSTM-CRF-ILP$_1$ | 93.4 | 81.4 | 86.9 | 74.1 | 71.1 | 72.6 | 49.6 | 43.3 | 46.2 |
| LSTM-CRF-ILP$_{multi}$ | 93.2 | **81.9** | **87.2** | 74.0 | **71.5** | **72.7** | 49.5 | **43.5** | **46.3** |

Table 2: Overall system performance of automatic evaluations. (%)

| Model | EC | AD | ED |
|---|---|---|---|
| CRF | 21.2 | 13.3 | 5.30 |
| MaxEnt | 17.7 | 11.7 | 5.44 |
| LSTM | 80.2 | 65.1 | 42.2 |
| LSTM-CRF | 81.6 | 68.6 | 44.1 |
| LSTM-CRF-ILP$_1$ | 85.4 | 70.2 | 44.2 |
| LSTM-CRF-ILP$_{multi}$ | **85.5** | **70.4** | **44.6** |

Table 3: Average F1 scores of overall system performance of manual evaluations. (%)

| Event type | P | R | F |
|---|---|---|---|
| olympics.medal_honor [2] | ↓ 25.0% | ↓ 5.0% | ↓ 13.8% |
| film.performance | ↓ 21.4% | ↑ 3.1% | ↓10.3% |
| business.acquisition | → | ↓ 7.1% | ↓ 5.4% |
| tv.appearance [3] | ↓ 9.5% | ↑ 3.0% | ↓ 3.1% |
| film.release [4] | ↓ 7.7% | ↑ 5.6% | ↓ 0.55% |

Table 4: Top 5 event types whose performances on event classification differ most from automatic evaluation. The model we evaluated is LSTM-CRF-ILP$_{multi}$

by two annotators, and the inter-annotator agreement is 87% for event types and 79% for arguments.

**Results** Table 3 presents the average results of manual evaluations where we measure precision, recall and F1 by the same standards of correctness as automatic evaluation.

| **S5**: That night, in an apparent bid to kill Amos, the car instead runs over the sheriff, leaving Chief Deputy Wade Parent (played by James Brolin) in charge. | | | |
|---|---|---|---|
| **Event Type** | film.performance (Wrong labeled in data generation) | | |
| **Arguments** | actor | character | film |
| | *James Brolin* | *Wade Parent* | *the car* |
| **S6**: Nicholas Hammond (born May 15, 1950) is an American actor and writer who is perhaps best known for his roles as Friedrich von Trapp in the film The Sound of Music, and as Peter Parker/Spider-Man on the CBS television series The Amazing Spider-Man. | | | |
| **Event Type** | film.performance | | |
| **Arguments** | actor | character | film |
| | *Nicholas Hammond* | *Friedrich von Trapp* | *The Sound of Music* |
| **Event Type** | tv.regular_tv_appearance (Missing in generated data) | | |
| **Arguments** | actor | character | series |
| | *Nicholas Hammond* | *Peter Parker/Spider-Man* | *The Amazing Spider-Man* |

Figure 3: Example outputs of LSTM-CRF-ILP$_{multi}$.

We can draw similar conclusions about the comparison of performances between different models as automatic evaluation. We demonstrate that LSTM-CRF-ILP$_{multi}$ is the most effective model in event extraction as it attains the highest F1 score in both manual and automatic evaluation.

Moreover, manual evaluation helps us to gain a deep insight of our generated data and proposed models. We further conduct automatic evaluation on this manual annotated dataset and list the top 5 event types whose F1 scores of LSTM-CRF-ILP$_{multi}$ differ greatly from automatic evaluation in Table 4.

[2] The full name is olympics.olympic_medal_honor in Freebase.

[3] The full name is tv.regular_tv_appearance in Freebase.

[4] The full name is film.film_regional_release_date in Freebase.

Most of the performance differences blame on the stage of data generation. Figure 3 examples two types of errors in data generation. Some of the sentences automatic generated test set are noisy, in other words, they do not imply any event while still match all key properties of certain instances. Take S5 as an example, though the phrases *the car* matches a film name, it does not indicate this film, and there is no explicit evidence expressing that an actor starring in a film. This is a bottleneck of our data generation strategy. During manual evaluation, we find 16 negative sentences and all of them are mistakenly labeled due to the same reason. Unfortunately, our model fails to identify some of these negative sentences.

Remarkably, our LSTM-CRF-ILP$_{multi}$ model can help find more CVT instances that not referenced in Freebase. There are two events mentioned in S6, while the arguments of the second event do not match any CVT instances in Freebase, leading to an omitting event in data generation. This phenomenon suggests that learning from distant supervision provided by Freebase, our model can help complete and update properties of Freebase instances in return.

## 5 Related Work

Event extraction is one of the fundamental research topics in information extraction and natural language understanding. There are diverse descriptions of event extraction tasks defined by different programs (MUC [Grishman and Sundheim, 1996], ACE [Doddington *et al.*, 2004], ERE [Song *et al.*, 2015] and TACKBP [Mitamura *et al.*, 2015]), all of which can be summarized as template-filling-based event extraction. Nearly all approaches on these corpora use a supervised paradigm, and highly rely on the human-annotated data. We typically divide them into feature-based methods and neural-network-based methods.

Feature-based methods usually rely on a variety of elaborately features. They aim to exploit different feature extraction strategies and evaluate feature contributions to the classification. Li et al. [2013] jointly learn trigger labeling and argument labeling using structured perceptron to capture both local and global features of triggers and arguments.

Neural-network-based methods are free of hard feature engineering and error propagation from external NLP tools. Two types of neural works have been employed. Chen et al. [2015] propose a convolutional neural network (CNN) with a dynamic multi-pooling layer to capture sentence-level features better. Nguyen et al. [2016] propose a bidirectional RNN with various memory matrices to jointly learn triggers and arguments, which benefits from both joint models and neural network models. However, to our knowledge, LSTM-CRF models have not been applied in earlier studies.

In contrast to these prior systems focused on small human-labeled corpus, Huang et al. [2016] propose a novel Liberal Event Extraction paradigm which automatically discovers event schemas and extract events simultaneously from any unlabeled corpus.

Freebase is a typical resource for distant supervision in binary relation extraction [Mintz *et al.*, 2009; Zeng *et al.*, 2015]. However, we can not simply apply their data labeling strategies to event extraction, as event structures is much more like a n-ary relation extraction.

## 6 Conclusions and Future Work

Motivated by the data labeling problem in ACE event extraction system, we explore a method to generate training data automatically based on four hypotheses between arguments and events. The manual evaluation result demonstrates that our hypotheses are feasible, and the produced dataset is clean and high-quality.

In addition, we propose a LSTM-CRF model with a ILP-based post inference approach to perform event extraction on the auto-generated data. Experimental results of both manual and automatic evaluation show that our model can not only extract simple events, but also multi-type events and multiple events within one sentence.

Furthermore, we find that our model can extract information not collected by Freebase. In the future, we intend to explore an approach to extend this work to knowledge base population.

## References


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

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *EMNLP 2013*, volume 2, page 6, 2013.

[Berger *et al.*, 1996] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD 2008*, pages 1247–1250. ACM, 2008.

[Bottou, 2010] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Chen *et al.*, 2015] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL 2015*, pages 167–176, 2015.

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

[Doddington *et al.*, 2004] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC 2004*, volume 2, page 1, 2004.

[Grishman and Sundheim, 1996] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *COLING 1996*, volume 96, pages 466–471, 1996.

[Gurobi Optimization, 2016] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2016.

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

[Huang *et al.*, 2015] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[Huang *et al.*, 2016] Lifu Huang, T Cassidy, X Feng, H Ji, CR Voss, J Han, and A Sil. Liberal event extraction and event schema induction. In *ACL 2016*, 2016.

[Kudo, 2005] Taku Kudo. Crf++: Yet another crf toolkit. *Software available at http://crfpp. sourceforge. net*, 2005.

[Lafferty *et al.*, 2001] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, volume 1, pages 282–289, 2001.

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

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

[Manning *et al.*, 2014] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL 2014*, pages 55–60, 2014.

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

[Mintz *et al.*, 2009] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP 2009*, pages 1003–1011. Association for Computational Linguistics, 2009.

[Mitamura *et al.*, 2015] Teruko Mitamura, Yukari Yamakawa, Susan Holm, Zhiyi Song, Ann Bies, Seth Kulick, and Stephanie Strassel. Event nugget annotation: Processes and issues. In *Proc. of the Workshop on EVENTS at the NAACL-HLT*, pages 66–76, 2015.

[Nguyen *et al.*, 2016] Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. Joint event extraction via recurrent neural networks. In *NAACL-HLT 2016*, pages 300–309, 2016.

[Rabiner, 1989] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77(2):257–286, 1989.

[Song *et al.*, 2015] Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. From light to rich ere: annotation of entities, relations, and events. In *Proc. of the Workshop on EVENTS at the NAACL-HLT*, pages 89–98, 2015.

[Zeng *et al.*, 2015] Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP 2015*, pages 1753–1762, 2015.