Learning to Extract Events without Human-annotated Text

Abstract

Existing event extraction systems are often supervised and rely on expert-annotated datasets, with limited event types. However, designing and constructing these high-quality corpora, usually with limited size and coverage of event types, is costly, which makes learned extractors hard to generalize. With the essence of distant supervision, we investigate the possibilities of automatic construction of training data for various event types with the help of structured knowledge bases. We further propose a novel neural network with ILP-based post inference committing to handling two challenges in event extraction: multi-type events and multi-word arguments. Both automatic and manual evaluations demonstrate that it is possible to learn to extract various events, according to existing knowledge bases, without human-annotated training data.

1 Introduction

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

However, constructing training data for ACE task is expensive. First, 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 guidelines, there is still disagreement among human annotators about what should (not) be regarded as triggers/arguments. For example, can a prepositional phrase or a portion of a word trigger an event, e.g., *in prison* triggers an *arrest* event, or, *ex* in *ex-husband* triggers a *divorce* event? Besides, ACE event extraction systems remain two major limitations: single-token trigger labeling, and one type for one event.

It would be interesting to see (1) can we automatically build a dataset for event extraction without experts involved? and (2) can we have an event extractor that handles more realistic scenarios, e.g., when trigger annotations are unavailable, or events with more than one type.

First, we observe that structured knowledge bases (KB) often organize complex structured information in tables, which share similar structures with ACE event definitions. A particular entry of such tables usually implies the occurrence of certain events. 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. However, there are two major challenges when leveraging KB to event extraction: first, event structures 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 arguments. Second, there is no explicit trigger information in any existing knowledge base. Therefore, to explore the distant supervision (`DS`) assumption in event extraction, we investigate different hypotheses for better data quality and quantity. Among them, the vital one is that, for a particular event type, there is a group of **key arguments** which together can imply an event instead of explicit triggers. We utilize Freebase as our knowledge base and Wikipedia articles as text 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 examples of sentences annotated by our algorithm.

Second, unlike previous studies focus on tasks defined by ACE evaluation framework [Ahn, 2006; Li *et al.*, 2013; Chen *et al.*, 2015; Nguyen *et al.*, 2016], we propose a novel event extraction paradigm with key arguments to characterize an event type. We consider event extraction as two sequence labeling subtasks, namely event detection and argument detection. Inspired by neural network models in sequence labeling tasks [Huang *et al.*, 2015; Lample *et al.*, 2016], we utilize LSTM-CRF models to label key arguments and non-key arguments in the each sentence separately. However, event structures are not simple sequences and there are strong dependencies among key arguments. We therefore reformulate the hypotheses as constraints, and apply linear integer programming to output multiple optimal label sequences to capture multi-type events.

In this paper, we exploit existing structured knowledge bases, e.g., Freebase, as distant supervision to automatically annotate event structures from plain text without human annotations. We further propose a novel event extraction paradigm that harnesses key arguments to imply certain event types without explicit trigger annotations. We present an LSTM-CRF model with post inference to extract both Freebase-style events as well as multi-type event mentions on the generated dataset, which is demonstrated effective by both manual and automatic evaluations.

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 Dataset Preparation

Here, we will employ the event-related entries in Freebase [Bollacker *et al.*, 2008] to automatically annotate event descriptions in Wikipedia's pages, with the essence of `DS`. We regard a sentence as positive when it mentions the occurrence of an event, or otherwise negative. For example, S1 and S2 are positive examples with their arguments in italics and underlined (also shown in Figure 1), while S3 and S4 are negative.

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 is a structured knowledge base, and there are three basics concepts: *instance*, *type* and *property*. *Instances* are entries in Freebase, and related to real-world entities and their relationships. *Types* are different perspectives of *instances*. ***Compound Value Type*** (CVT) is a special type in Freebase to represent complex structured data where instances are described with multiple ***properties***, usually organized in a table. Some of the CVT schemas indeed imply certain events, e.g., *people.marriage*, *military.military_service* and *business.acquisition*, and closely resemble to event structures, where CVT properties can be treated as event arguments. As shown in Figure 1,

the properties of CVT *business.acquisition* actually can be used to label the participants and attributes of the events mentioned in S1 and S2.

We use the Freebase copy of 2013-06, containing 1010 CVTs. After filtering out those describing the Freebase structures or irrelevant to events (e.g., *food.recipe_ingredient*), we obtain 24 CVTs with around 280 million instances.

2.2 Data Generation

Now we will describe how to automatically collect training data with quality event annotations, by pushing forward the `DS` framework.

H1: Positive sentences should contain all properties

Our first hypothesis is that *if a sentence contains all properties of an entry in a CVT, this sentence will be considered as a positive instance to indicate an event of such type characterized by this CVT*. We will then label the sentence as a mention of this CVT event, and the words or phrases that match this entry's properties as the involved arguments, with the roles specified by their corresponding property names. For example, S1 contains all the properties of instance $m.07bh4j7$ with a CVT type *business.acquisition*, we thus consider S1 as a positive instance implying an event about *business.acquisition*, and *BMC Software*, *Remedy Corp*, *Service Management Business Unit* and *2004* will be labeled as the arguments that play the role of *acquiring_company*, *company_acquired*, *divisions_formed*, and *date* in this event, respectively.

H2: Positive sentences should contain all key properties

In practice, we find that *H1* is too strict to include many positive instances like S2. We thus relax *H1* by replacing **all properties** with **all key properties**. We define the CVT property that plays an important part in its CVT structure and helps to distinguish with other CVTs as a **key property**. A **key argument** is the word/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*, therefore, S2 should be a positive instance for a *business.acquisition* event, since it contains all key properties. Table 1 lists the key properties of four CVTs.

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

$$degree_{cvt,prop} = log \frac{count(cvt, prop)}{count(cvt) \times count(prop)} \quad (1)$$

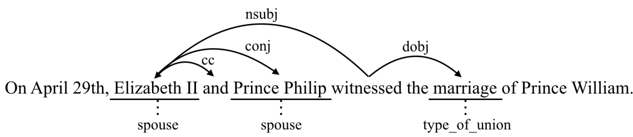
where $count(cvt)$ is the number of all instances of type cvt, $count(prop)$ is the number of times $prop$ appearing in all CVTs, and $count(cvt, prop)$ is the number of cvt instances that contain the property $prop$.

H3: Key properties should include time properties

Although time-related arguments are often missing in the currently imperfect KBs, time-related properties are indeed crucial to indicate an actual event mention, e.g., S3, containing *Microsoft* as *acquiring_company* and *aQuantive* as *company_acquired* but without time-related arguments, should

CVT	Key properties
award.award_honor	award_winner, award, ..., year
film.performance	actor, film, character
education.education	institution, student, end_date
business.employment_tenure	company, title, person, from

Table 1: Examples of key properties of four CVTs.



id \ property	spouse	spouse	type_of_union	location_of_ceremony	from
m.028zms8	Elizabeth II	Prince Philip	marriage	Collegiate Church	1947/11/20

Figure 2: An illustration of dependency tree of S4, which partially matches an entry of *people.marriage*.

not be considered as a negative example for event *business.accquisition*. We thus include time-related properties with highest importance scores as supplementary key properties.

H4: Key properties should be closer on dependency path
Intuitively, two arguments participating in the same event are likely to be closer to each other in syntactic structures, which will help to eliminate negative instances. In Figure 2, both *Prince Philip* and *marriage* can be matched as key properties in a *people.marriage* entry, but are far from each other on the dependency path, thus S4 should be labeled as negative. In our experiments, we set the maximum distance between two key arguments as 2.

We conduct a series of manual evaluations on the quantity and quality of the datasets produced by different hypotheses (see Sec 4.2), and the combination of *H3* and *H4* produces the best dataset, thus serves as our final strategy.

3 Our Approach

Existing works in event extraction rely on explicit trigger identification to detect the occurrence of an event, which is crucial to later decide its event type and label its arguments. In our automatically collected dataset, which human-labeled event triggers are unavailable, we argue that **key arguments** can play the same role as explicit event triggers. We thus treat the event extraction as a pipeline of two subtasks, namely, event detection and argument detection.

Event detection aims to identify key arguments in a sentence. If a sentence contains **all** key arguments of a specific event type, it will be considered to imply an event of the corresponding type. Take S1 as an example, *Remedy Corp*, *BMC Software*, and *2004* could be identified as *company_acquired*, *acquiring_company*, and *date*, respectively, indicating that S1 may mention a *business.acquisition* event.

Argument detection aims 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 *divisions_formed*.

3.1 Event Detection

Next, we first present our solution for multi-words arguments, and then introduce each component in our model.

Tagging scheme There 68% of arguments in our dataset consisting of more than one word. To address this issue, we model each subtask in a sequence labeling paradigm rather than word-level classifications. Each word in the given sentence is tagged with 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 fit for sequence labeling, which maintains a memory based on historical contextual information. Formally, given a sentence $\boldsymbol{w} = \{w_1, w_2, \dots, w_n\}$ of length n, we use $\mathbf{x}_t$ to represent feature vector, e.g., word embeddings, corresponding to the t-th word w_t. At each time step t, an LSTM unit takes $\mathbf{x}_t$ as input and computes the output vector $\mathbf{h}_t$ through several multiplicative gates. The output vector is fed into a softmax layer to estimate a probability distribution over all possible labels.

CRF A straightforward way to find the label sequence for given sentence is to choose the best label for each word individually according to LSTM output. However, this greedy strategy ignores the dependencies between labels, thus can not guarantee the best sequence. Therefore, we introduce a CRF layer over the LSTM output, which is admittedly effective in various sequence labeling tasks [Collobert *et al.*, 2011; Huang *et al.*, 2015].

We consider $\mathbf{P}$ to be a matrix of confidence scores output by LSTM, 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 takes a transition 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 $\boldsymbol{y} = \{y_1, y_2, \dots, y_n\}$ is measured by the sum of LSTM 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 Basically, event detection is a structure prediction problem, while the output sequences of LSTM-CRF do not necessarily satisfy the structural constraints. For instance, regardless of how many key arguments are correctly identified by LSTM-CRF, if there is one key argument missing, this detection should be considered as failed.

We thus propose to apply Integer Linear Programming (ILP) to further globally optimize the LSTM-CRF output to produce the best label 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 as:

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

where we consider the following four constraints:

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

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

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

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

C3: If the 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, all its key arguments should co-occur in the sentence, or none of them appears 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:

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

where n is the length of the sentence.

In order to address the multi-type event mention issue, we allow our ILP solver to output multiple optimal sequences. Specifically, after our model outputs the best sequence $\boldsymbol{s}^t$ at time t, we remove the previously best solutions $\{\boldsymbol{s}^1, \dots, \boldsymbol{s}^t\}$ from the solution space, and re-run our solver to obtain the next optimal sequences $\boldsymbol{s}^{t+1}$. We repeat the optimization procedure until the difference between the scores of $\boldsymbol{s}^1$ and $\boldsymbol{s}^T$ is greater than a threshold λ, and consider all solutions $\{\boldsymbol{s}^1, \boldsymbol{s}^2, \dots, \boldsymbol{s}^{T-1}\}$ as the optimal label sequences. We use Gurobi [Gurobi Optimization, 2016] as our ILP solver and set $\lambda = 0.05 \times n$, which averagely produce 1.04 optimal sequences for each sentence.

3.2 Argument Detection

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

We adopt the same LSTM-CRF architecture (in Sec 3.1) for argument detection, where we encode the event label (output of event detection) of each word into a key-argument feature vector through a look-up table, and concatenate it with the original word embedding as the input to the new LSTM-CRF. Note that we do not need post inference here.

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 dataset. We then randomly select 4800 sentences for training and 1180 sentences as test set, and the rest 1200 sentences for validation. We conduct both automatic evaluation and manual evaluation in the experiments. We first manually evaluate the quality of our test set. Next, we regard the noisy 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 differences 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. 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. Word embeddings are pretrained using skip-gram word2vec model [Mikolov *et al.*, 2013] over the whole Wikipedia dump and fine tuned during training. To mitigate overfitting, we apply a dropout rate of 0.5 on both the input and output layers.

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 2: 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 2, the strictest hypothesis, *H1*, guarantees the quality and confidence of generated data, while we can merely obtain 203 sentences and cover 9 types of events, which is quite insufficient for further training. *H2* is looser than *H1*, though expands the resulting dataset, it produces 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 3, 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.

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 3, 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. We further investigate the effect of LSTM-CRF-ILP$_{multi}$. Evaluated on the sentences containing multi-type event mentions, the F1 scores of LSTM-CRF-ILP$_{multi}$ in event classification, argument detection and event detection are 70.7%, 26.9% and 58.4%, respectively. As we can see from Table 3, this strategy can detect multi-type event mentions for a sentence, contributing to the increase of recalls, and F1 scores with a little drop of precisions.

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

4.5 Manual Evaluations

We randomly sample 150 unlabeled sentences from test data set. Annotators are asked to annotate the events and arguments to each sentence following two 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. Each sentence is independently annotated by two annotators, and the inter-annotator agreement is 87% for event types and 79% for arguments.

Table 4 presents the average F1 score of manual evaluations. 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 achieves the highest F1 score in both manual and 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$_{multi}$.

Moreover, manual evaluation helps us to gain a deep insight of our data and models. We further conduct automatic evaluation on the 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 5.

Most of the performance differences are caused by 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 which are mistakenly labeled due to this reason. Unfortunately, our model fails to rectify 10 of them.

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 3: 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 4: Average F1 scores of overall system performance of manual evaluations. (%) EC, AD, ED denote the event classification, argument detection and event detection, respectively.

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.

Event type	P	R	F
olympics.medal_honor [2]	$\downarrow$ 25.0%	$\downarrow$ 5.0%	$\downarrow$ 13.8%
film.performance	$\downarrow$ 21.4%	$\uparrow$ 3.1%	$\downarrow$10.3%
business.acquisition	$\rightarrow$	$\downarrow$ 7.1%	$\downarrow$ 5.4%
tv.appearance [3]	$\downarrow$ 9.5%	$\uparrow$ 3.0%	$\downarrow$ 3.1%
film.release [4]	$\downarrow$ 7.7%	$\uparrow$ 5.6%	$\downarrow$ 0.55%

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

5 Related Work

Most event extraction works are within the tasks defined by several evaluation frameworks (e.g., MUC [Grishman and Sundheim, 1996], ACE [Doddington *et al.*, 2004], ERE [Song *et al.*, 2015] and TAC-KBP [Mitamura *et al.*, 2015]), all of which can be considered as a template-filling-based extraction task. These frameworks focus on limited number of event types, which are designed and annotated by human experts and hard to generalize to other domains. Furthermore, existing extraction systems, which usually adopt a supervised learning paradigm, have to rely on those high-quality training data within those frameworks, thus hard to move to more domains in practice, regardless of feature-based [Gupta and Ji, 2009; Hong *et al.*, 2011; Li *et al.*, 2013] or neural-network-based methods [Chen *et al.*, 2015; Nguyen *et al.*, 2016].

Besides the works focusing 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. In contrast, we propose to exploit existing structured knowledge bases, e.g., Freebase, to automatically discover types of events as well as their corresponding argument settings, without expert annotation, and further automatically construct training data, with the essence of distant supervision [Mintz *et al.*, 2009].

Distant supervision (`DS`) has been widely used in binary relation extraction, where the key assumption is that sentences containing both the subject and object of a $<subj, rel, obj>$ triple can be seen as its support, and further used to train a classifier to identify the relation rel. However, this assumption does not fit to our event extraction scenario, where an event usually involves several arguments and it is hard to collect enough training sentences with all arguments appearing in, as indicated by the low coverage of **H1**. We therefore investigate different hypotheses for event extraction within the `DS` paradigm and propose to utilize time and syntactic clues to refine the `DS` assumption for better data quality. We further relieve the reliance on event trigger annotations by previous event extractors, and define a novel event extraction paradigm with key arguments to characterize an event type.

6 Conclusions and Future Work

In this paper, we propose a new event extraction paradigm without expert-designed event templates by leveraging structured knowledge bases to automatically acquire event schema and corresponding training data. We propose an LSTM-CRF model with ILP-based post inference to extract events without explicit trigger annotations. Experimental results on both manual and automatic evaluations show that it is possible to learn to extract KB-style events on automatically constructed training data. Furthermore, our model can extract information not covered by Freebase which indicates the possibility to extend this work to knowledge base population.

[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.

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.

[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.

[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.

[Gupta and Ji, 2009] Prashant Gupta and Heng Ji. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP 2009*, pages 369–372. Association for Computational Linguistics, 2009.

[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.

[Hong *et al.*, 2011] Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. Using cross-entity inference to improve event extraction. In *ACL 2011*, pages 1127–1136. Association for Computational Linguistics, 2011.

[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.