Event Extraction without Human-annotated Text

Anonymous EMNLP submission

Abstract

Existing event extraction systems are typically investigated in the supervised learning paradigm. The effectiveness of these systems heavily relies on the quality of expert-annotated datasets which still require a costly and time-consuming process to construct. As a result, the built datasets often only cover a limited variety of event types, making the learned event extractor hard to generalize. In this paper, we address the problem of automatically building event extractors for rich event types with little expert involvement. We achieve this by employing distant supervision to automatically create event annotations from unlabelled data using structured knowledge bases. We then propose a novel neural network model with ILP-based inference, committing to detecting events of various types and extracting their corresponding arguments involved. We evaluate our approach by automatically collecting training data for many event types from Freebase, where our proposed extraction model is designed to identify both typed event mentions and typed arguments. Both automatic and manual evaluations demonstrate that it is possible to learn to effectively extract various events without human-annotated training data.

1 Introduction

Event extraction, as represented by the Automatic Content Extraction (ACE) task, is a key enabling technique for many natural language processing (NLP) applications. Current event extractors are typically built through applying supervised learning to learn over labelled datasets. This means that the performance of event extraction systems highly dependent on the quality of the training datasets.

Constructing high-quality training data for event extraction is, however, an expensive and error-prone processFIX:(). This requires the involvement of linguists to design annotation templates and rules, and the employment of annotators to manually label data. To scale up event extraction, we need to reduce expert involvement. In addition to the extensive human involvement, existing event extraction systems have two major drawbacks – they can only handle (1) single-token triggers[1] and (2) and scenarios where each event is merely associated with a single type. To make event extractors practical, we need to support situations where the trigger annotations are unavailable or one event is associated with multiple types.

This paper presents a novel approach for automatic training data generation, specifically targeting event extraction. Our approach advances prior workFIX:() in two aspects: (1) it does not rely on expert-annotated texts (hence it reduces expert involvement) and (2) supports multi-typed events. The former is achieved by employing distant supervision to automatically annotate event structures from plain text, using information extracted from existing structured knowledge bases such as Freebase. The later is achieved by using a novel neural network model with ILP-based inference to extract multiple event types without relying on explicit trigger annotations.

Our first insight is that structured knowledge bases (`KB`) typically organize complex structured information in tables; and these tables often share similar structures with ACE event definitions, i.e. a particular entry of such tables usually implies the occurrence of certain events. Recent studies

[1] In the ACE task, a trigger is the word that most clearly expresses the occurrence of an event.

(Mintz et al., 2009; Zeng et al., 2015) have demonstrated the effectiveness of KB as distant supervision (DS) for binary relation extraction. In this paper, we aim to extend DS to extract events of n-ary relations and multiple arguments (rather than just binary relation). One of the hurdles for doing this is the lacking of explicit trigger information in existing KB. Our solution to the problem is to use a group of **key arguments** rather than explicit triggers to capture a particular event. For example, we can use "*spouse*" as the key argument to identify "*marriage*" events.

Our second innovation is that unlike previous studies which 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.

We evaluate our approach by applying it to automatically collect training data with multiple event types using Freebase. We use the proposed extraction model to identify both typed event mentions and typed arguments. Our experitmental results on both automatic and manual evaluations demonstrate that our approach can effectively extract various events without human-annotated training data. FIX:quantified numbers?

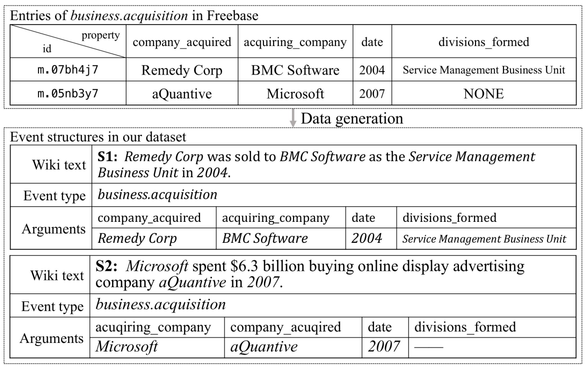
2 The Event Extraction Task

Event extraction aims to detect the occurrence of events with specific types and extract their typed participants or attributes from text. We clarify the following terminologies within our work:

- **Event mention**: a phrase or sentence within which an event is described, including its type and arguments.
- **Argument**: an entity mention, temporal expression or value that is involved in an event, with specific roles.
- **Key argument**: the argument that plays an important role in one event, and helps to distinguish with other events.

2.1 Indirect Supervision for Event

We first utilize Freebase (Bollacker et al., 2008) as a source of supervision to guide our data construction, where ***Compound Value Type*** (CVT) is a special type to represent complex structured data with multiple *properties*, usually organized in a table. Some CVT schemas indeed imply certain events, e.g., *business.acquisition*, and closely resemble to event structures, where CVT properties can be treated as event arguments[2]. As shown in Figure 1, the properties of CVT *business.acquisition* actually can be used to label arguments of the events mentioned in S1 and S2. We use the Freebase copy of 2013-06, containing 1010 CVTs. After manually filtering out those describing Freebase structure or irrelevant to events, we obtain 24 CVTs with around 280 million instances.


Entries 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 a CVT table in Freebase, and labeled sentences in our dataset. *Company_acquired*, *acquiring_company* and *date* are key arguments in *business.acquisition*.

Besides structured knowledge base, tables or lists that are used to sum up certain activities or occasions could be considered as a source of supervision for event extraction, as well. We thus investigate tables collected from Wikipedia pages, potentially referring to three types: winning of the Olympics, music and film awards, mergers and acquisitions[3].

[2] Therefore, we also use the term "argument" to refer to CVT property in the rest of paper.

[3]For example, https://en.wikipedia.org/

2.2 Dataset Construction

Here, we employ the event-related entries of Freebase CVT tables to illustrate how to automatically annotate event mentions in Wikipedia's articles, with the essence of distant supervision (`DS`): *A sentence that contains* ***all key arguments*** *of an entry in an event table (e.g., CVT table) is likely to express that event*. We will then label this 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.

We regard a sentence as *positive* when it mentions the occurrence of an event, or *negative* otherwise. 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.

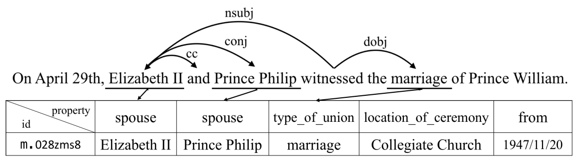
The selection strategy for key arguments for a given event type is based on two criteria: (1) *Key arguments should have high importance value*; (2) *Key arguments should include time-related arguments*.

The *Importance value* of an argument *arg* (e.g., *date*) to its event type *cvt* (e.g., *business.acquisition*) can be defined as:

$$I_{cvt,arg} = log\frac{count(cvt, arg)}{count(cvt) \times count(arg)} \quad (1)$$

where $count(cvt)$ is the number of instances of type cvt, $count(arg)$ is the number of times arg appearing in all CVT types, and $count(cvt, arg)$ is the number of cvt instances that contain arg.

Although time-related arguments are often missing in the currently imperfect KBs, they are indeed crucial to indicate the actual occurrence of an event, e.g., S3, containing *Microsoft* as *acquiring_company* and *aQuantive* as *company_acquired* but without time-related arguments, will be mistakenly considered as a positive sample for event *business.accquisition*.



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

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: The dependency tree of S4, which partially matches an entry of *people.marriage*.

Intuitively, two arguments involving in the same event mention are likely to be closer within the syntactic structure. In Figure 2, both *Prince Philip* and *marriage* can be matched as key arguments in a *people.marriage* entry, but are far from each other on the dependency parse tree, thus S4 should be labeled as negative.

We conduct a series of manual evaluations on the quantity and quality of the datasets produced by different strategies (see Sec 4.1), and our final strategy is: for each CVT, we first sort all its arguments in descending order by their importance values, and select the top half arguments as key arguments. We then include the time-related argument with highest importance value as a supplementary key argument. Finally, we eliminate sentences in which the dependency distances between any two key arguments are greater than 2.

3 Our Approach

Previous event extraction systems mainly rely on explicit trigger identification to detect the occurrence of an event, which is then used to decide its event type and label its arguments. In our automatically collected dataset, where 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 the following two steps:

- **Event detection**: 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 mention of this specified type.
- **Argument detection**: to identify other non-key arguments for each event in the sentence.

`wiki/List_of_mergers_and_acquisitions_by_IBM`

Take S1 as an example, in event detection, *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. During argument detection, *Service Management Business Unit* should be identified as *divisions_formed*, which, together with the detected key arguments, form a full mention for a *business.acquisition* event.

3.1 Event Detection

68% of arguments in our dataset consist of more than one word. We thus formulate 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. We accordingly propose a novel BLSTM-CRF model with ILP-based inference to detect and organize key arguments.

BLSTM 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 $\boldsymbol{w} = \{w_1, w_2, \ldots, w_n\}$ 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 t, a forward LSTM layer takes $\mathbf{x}_t$ as input and computes the output vector $\overrightarrow{\mathbf{h}}_t$ of the past context, while a backward LSTM layer reads the same sentence in reverse and outputs $\overleftarrow{\mathbf{h}}_t$ given the future context. We concatenate these two vectors to form the output vector of BLSTM, which 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 the 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 $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$ is measured by the sum of LSTM outputs and transition scores:

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

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 BLSTM-CRF do not necessarily satisfy the structural constraints. For instance, regardless of how many key arguments are correctly identified by BLSTM-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 BLSTM-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:

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

where we consider the following four constraints:

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

$$\sum\nolimits_{l,l'} v_{i,l,l'} = 1 \quad (3)$$

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.:

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

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

$$v_{i,\texttt{I-arg},l'} = v_{i-1,\texttt{B-arg},\texttt{I-arg}} + v_{i-1,\texttt{I-arg},\texttt{I-arg}} \quad (5)$$

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:

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

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. Next, we will identify the remaining non-key arguments in the sentence.

We adopt the same BLSTM-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 BLSTM-CRF. Note that we do not need post inference here.

4 Experiments

We use Freebase and the English Wikipedia dump of 2016-11-20, to construct our dataset. Statistics for the generated dataset from Freebase is shown in Table 1. Note that we only include positive sentences in this dataset. We first manually evaluate the quality of our test set and then regard the automatically generated data as gold standard and evaluate our model accordingly. Next, we manually evaluate a subset of events detected by our model and analyze the differences with regards to the automatic evaluation. Finally, we conduct evaluation on a smaller dataset constructed according to Wikipedia tables and articles.

	Train	Dev	Test
#Sent.	4800	1200	1180
#Eve.	4918	1247	1229
#Arg.	17274	4318	4248
%Multi_Eve.	3.0	3.2	3.1

Table 1: Statistics for the generated dataset. *#Sent.* is the number of sentences, *#Eve.* is the number of event mentions, and *#Arg.* is the number of event arguments. *%Multi_Eve.* is the ratio of multi-type events.

Evaluation Metrics: We evaluate our models in terms of precision (P), recall (R), and F-measure (F) for each subtask. These 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 *event detection*, an event is correctly detected if its type and all its key arguments match a reference event within the same sentence. For *argument detection*, an argument is correctly detected if its offsets, role, and related event type exactly match the reference argument within the same sentence.

Training: All hyper-parameters are tuned on the development set. During event detection, we set the size of word embeddings 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 (Mikolov et al., 2013) on English Wikipedia and fine tuned during training. We apply dropout (0.5) on both input and output layers.

4.1 Dataset Evaluation

To investigate the possibility of automatically constructing training data for event extraction, we evaluate five datasets that utilize following strategies to determine key arguments and collect positive instances: (1) *ALL* means regarding all arguments as key arguments; (2) *IMP* means selecting the top half arguments with high importance value as key arguments; (3) *IMP&TIME* means adding a time-related argument with highest importance value to the set of key arguments defined by *IMP*; (4) *DIS* means eliminating sentence where dependency distances between any two key arguments are greater than 2. We randomly select 100 sentences from each dataset, and annotators are asked to determine whether each sentence implies a given event.

No.	Strategy	Sent.	Type	Post.
T1	*ALL*	203	9	98%
T2	*IMP*	108K	24	22%
T3	*IMP+DIS*	12K	24	37%
T4	*IMP&TIME*	9241	24	81%
T5	*IMP&TIME+DIS*	7180	24	89%

Table 2: Statistics of the datasets built with different strategies. *Sent.* is the number of sentences found. *Type* the number of different CVT types found within each dataset. *Post.* is the percentage of sentences mentioning the given events explicitly.

As shown in Table 2, it is not surprising that the most strict strategy, *T1*, guarantees the quality of the generated data, while we can merely obtain 203 sentences covering 9 types of events, which is insufficient for further applications. *T2* relaxes *T1* by allowing the absence of non-key arguments, which expands the resulting dataset, but introduces more noise, indicating that *T2* is inappropriate to be used as a soft constraint. Compared with *T2*, the significant quality improvement by *T4* proves that time-related arguments within CVT schemas are critical to imply an event occurrence. Among all strategies, data obtained by *T5* achieves the highest precision, while still accounting for 7,180 sentences, showing that it is feasible to automatically collect quality training data for event extraction without either human-designed event schemas or **extra** human annotations.

4.2 Baselines

We compare our proposed models with three baselines, including traditional feature-based methods and neural network models. All baselines are trained following the two-step pipeline, i.e., event detection and argument detection. For neural network methods, we train a BLSTM model that takes word embeddings as input, and outputs the label with the maximum probability among all possible labels. For feature-based methods, we apply Conditional Random Field (Lafferty et al., 2001) (with the CRF++ toolkit (Kudo, 2005)) and Maximum Entropy (Berger et al., 1996) (Le Zhang's MaxEnt toolkit) to explore a variety of elaborate features, such as lexical, syntactic and entity-related features, according to the state-of-art feature-based ACE event extraction system (Li et al., 2013). Note that during argument detection stage, we add the label of each word output by event detection as a supplementary feature.

4.3 Automatic Evaluation

As shown in Table 3, traditional feature-based models perform worst in both event detection and argument detection. One of the main reasons is the absence of explicit event trigger annotations in our dataset, which makes it impossible to include trigger-related features, e.g., trigger-related dependency features and positions of triggers. Although traditional models can achieve higher precisions, they only extract a limited number of events, resulting in low recalls. Neural-network methods perform much better than feature-based models, especially recalls, since they can make better use of word semantic features. Specifically, BLSTM can capture longer range dependencies and richer contextual information, instead of neighboring word features only. And the CRF component brings an averagely 4% improvement in all metrics, and by adding the ILP-based post inference module, our full model, BLSTM-CRF-ILP$_{multi}$, achieves the best performance among all models.

Effect of CRF Layer Every model with a CRF layer over its BLSTM output layer is superior to the one with a BLSTM layer only. Compared with the BLSTM, BLSTM-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, BLSTM-CRF takes into account the constraints between neighboring labels, and potentially increases the cooccurrences of key arguments regarding the same event type.

Effect of Post Inference As shown in Table 3, ILP-based post inference considerably improves the overall system performance, especially in event type classification. With the help of constraint **C4**, some dubious key arguments can be inferred through other key arguments from their contexts. Compared with BLSTM-CRF, BLSTM-CRF-ILP$_1$ produces an F1 gain of 7.4% in event type classification, 1.8% in event detection, and 4.6% in argument detection.

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
BLSTM	97.2	62.4	75.1	77.1	53.9	63.5	51.0	32.8	39.9
BLSTM-CRF	97.3	67.2	79.5	**78.0**	60.2	68.0	**54.4**	37.6	44.4
BLSTM-CRF-ILP$_1$	93.4	81.4	86.9	74.1	71.1	72.6	49.6	43.3	46.2
BLSTM-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 (%).

Multi-type Event Extraction We further investigate the effect of BLSTM-CRF-ILP$_{multi}$, which is the only model that can deal with the mulit-type event mentions. As shown in Table 3, the proposed strategy in BLSTM-CRF-ILP$_{multi}$ helps detect more event mentions, contributing to the increase of recalls, and F1 scores with a little drop of precisions. Evaluated on 38 sentences containing multi-type event mentions, the F1 scores of BLSTM-CRF-ILP$_{multi}$ in event type classification, event detection and argument detection are 70.7%, 58.4% and 26.9%, respectively.

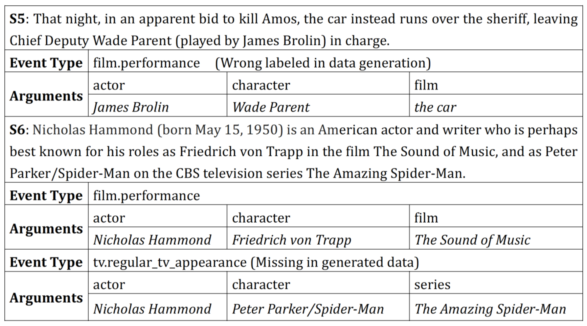
4.4 Manual Evaluation

We randomly sample 150 unlabeled sentences from the test 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 event types for positive sentences. Two annotators will independently annotate each sentence, and discuss to achieve an agreement. The inter-annotator agreement is 87% for event types and 79% for arguments.

Model	EC	AD	ED
CRF	21.2	13.3	5.30
MaxEnt	17.7	11.7	5.44
BLSTM	79.8	64.3	41.2
BLSTM-CRF	81.3	67.9	43.2
BLSTM-CRF-ILP$_1$	85.0	70.4	43.8
BLSTM-CRF-ILP$_{multi}$	**85.3**	**70.8**	**44.3**

Table 4: The F1 scores of different systems on the manually annotated data. EC, AD, ED denote the event type classification, argument detection and event detection, respectively.

Table 4 shows the system performances in the manual evaluation. We can draw similar conclusions about the comparison of performances between different models as automatic evaluation. It is clear that BLSTM-CRF-ILP$_{multi}$ is the most effective model in event extraction as it achieves the highest F1 scores 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 BLSTM-CRF-ILP$_{multi}$.

Moreover, we manually check the top 5 event types whose EC F1 scores by BLSTM-CRF-ILP$_{multi}$ differ greatly between automatic evaluation and manual evaluation, summarized in Table 5.

We find that most of the performance differences are caused by data generation. Figure 3 examples two types of errors in data construction. Several automatically labeled sentences do not imply any event while still matching all key properties of some CVT entries. For an example, although the phrase *the car* in S5 matches a film name, it does not indicate this film, and there is no explicit evidence indicating that an actor starred in this film. This is a bottleneck of our data generation strategy. During manual evaluation, we find 16 negative sentences which are mistakenly labeled as positive, and our model manages to rectify 6 of them.

Remarkably, our BLSTM-CRF-ILP$_{multi}$ model can help find more CVT instances that are not referenced in Freebase. There are two events mentioned in S6, while the arguments of the second event do not match any existing CVT instances in Freebase, which should be populated into Freebase. 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	$\downarrow$ 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	$\downarrow$ 9.5%	$\uparrow$ 3.0%	$\downarrow$ 3.1%
film.release	$\downarrow$ 7.7%	$\uparrow$ 5.6%	$\downarrow$ 0.55%

Table 5: The difference of EC F1 scores (by BLSTM-CRF-ILP$_{multi}$) between automatic and manual evaluation for top 5 event types.

4.5 Tables as Indirect Supervision

To investigate the applicability of our approach to other structured tables besides Freebase CVT tables, we automatically build a new dataset with the supervision provided by Wikipedia tables, which may characterize events about acquisition, winning of the Olympics games, and winning prestigious awards in entertainment (Table 6).

Event type	Entr.	Sent.	EC	AD	ED
Acquisition	690	414	87.0	72.0	69.6
Olympics	2503	1460	77.2	64.5	38.6
Awards	3039	2217	95.0	82.8	58.6

Table 6: Statistics of the Table dataset and performance of our model. *Entr.* is the number of table entries. *Sent.* is the size of training set.

We train our BLSTM-CRF-ILP$_{multi}$ on this dataset and evaluate it on 100 manually annotated sentences. We can see that without extra human annotations, our model can learn to extract events from the training data weakly supervised by Wikipedia tables. Given a specific event type, as long as we can acquire tables implying events of such type, it is possible to automatically collect training data from such tables, and learn to extract structured event representations of that type.

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 *T1*. 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 KBstyle 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.

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, pages 1–8.

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

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

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

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):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 2004*. volume 2, page 1.

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

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

Inc. Gurobi Optimization. 2016. Gurobi optimizer reference manual. http://www.gurobi.com.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):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 *ACL 2011*. Association for Computational Linguistics, pages 1127–1136.

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

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

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

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*. volume 1, pages 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* .

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

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

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

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

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

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

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. 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.

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