

Scale Up Event Extraction Learning via Automatic Training Data Generation

Abstract

Event extraction, which underpins many natural language processing applications, has long been investigated in a supervised learning paradigm. The effectiveness of event extractors is bound by the number and the quality of the training instances. Existing training data must be manually generated through a combination of expert domain knowledge and extensive human involvement. However, due to drastic efforts required in annotating text, the resulted datasets are usually small, which severely affects the quality of the learned model, making it hard to generalize. Our work develops an automatic approach for generating training data for event extraction. Our approach allows us to scale up event extraction training instances from thousands to hundreds of thousands, and it does this at a much lower cost than a manual approach. We achieve this by employing distant supervision to automatically create event annotations from unlabelled text using existing structured knowledge bases or tables. We then develop a neural network model with post inference to transfer the knowledge extracted from structured knowledge bases to automatically annotate typed events with corresponding arguments in text. We evaluate our approach by using the knowledge extracted from Freebase to label texts from Wikipedia articles. Experimental results show that our approach can generate a large number of high-quality training instances. We show that this large volume of training data not only leads to a better event extractor, but also allows us to detect multi-typed events – a feature that none of the existing event extractors can offer.

Introduction

Event extraction is a key enabling technique for many natural language processing (NLP) tasks. The goal of event extraction is to detect, from the text, the occurrence of events with specific types, and to extract arguments (i.e. typed participants or attributes) that are associated with an event. Current event extractor systems are typically built through applying supervised learning to learn over labelled datasets. This means that the performance of the learned model is bound by the quality and coverage of the training datasets.

To generate training data for event extraction, existing approaches all require manually identifying – if an event oc-

curs and of what type – by examining the event trigger¹ and arguments from each individual training instance. This process requires the involvement of linguists to design annotation templates and rules for a set of predefined event types, and the employment of annotators to manually label if an individual instance belongs to a predefined event type.

To determine an event type, existing approaches all require *explicitly* identifying the event trigger from the text to associate the text with an event. Because identifying event triggers is non-trivial **FIX()**, human involvement is deemed necessary. However, this thus makes the quality and the number of the generated instances dependent on the skill and available time of the annotators (Aguilar et al. 2014; Song et al. 2015). To scale up event extractor training, we therefore must take human annotators out from the loop of data labeling.

This work develops a novel approach to automatically label training instances for event extraction. Our key insight was that while event triggers are useful, they do not always need to be explicitly captured. Structured knowledge bases (KBs) such as Freebase already provide rich information of arguments, organizing as structured tables, to enable us to automatically infer the event type. For example, given the sentence “*USAir filed for Chapter 11 again in 2004*”, a *bankruptcy* event does not always have to be identified through finding the *bankrupt* trigger, as a *bankrupt_company* argument also implies such an event.

If we can find ways to exploit structured tables or lists, a single entry can then be used to label a large number instances without the involvement of human annotators. Such an approach is known as distant supervision (DS). Recent studies (Mintz et al. 2009; Zeng et al. 2015) have demonstrated its effectiveness in various NLP tasks. The central idea of DS is to use the knowledge extracted from a known knowledge base to *distantly supervise* the process of training data generation using another dataset. By employing DS, our approach is able to produce a large number of high-quality training instances.

Under this new DS paradigm, we develop a novel event extractor architecture based on neural networks and linear

¹In the event extraction task, a trigger is the word or phrase that most clearly expresses the occurrence of an event. For example, *ex* in *ex-husband* may trigger a *divorce* event.

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

company
acquired
acquiring
company
divisions
formed
date

(a) An example sentence from Wikipedia

| id | company_acquired | acquiring_company | date | divisions_formed |
|-----------|------------------|-------------------|------|----------------------------------|
| m.07bh4j7 | Remedy Corp | BMC Software | 2004 | Service Management Business Unit |

(b) Entry of `business.acquisition` in Freebase

Figure 1: The event type of the sentence shown in (a) can be automatically inferred using the structured table given by Freebase in (b).

integer programming. One of the key innovations of our approach is that the model does not rely on explicit triggers. Instead, it uses a set of key arguments to characterize an event type. As a result, it eliminates the need to explicitly identify event triggers, a process that requires heavy human involvement.

We evaluate our approach by applying it to perform sentence-level event extraction. To generate training data, we use Freebase to automatically label texts from a subset of Wikipedia articles. We then learn an event extractor using the generated data to extract events from unseen texts.

Using **FIX:24** structured tables from Freebase, we are able to generate over 100 thousands training instances within two hours. This results in a training dataset that is 100x greater than the widely used ACE dataset – which took **FIX:xx man-months** to build **FIX: ()**. We show that the quality of the automatically generated dataset is comparable to ones that were manually constructed by human experts. Experimental results also confirm that this new paradigm of data generation and modeling not only allows us to build a more efficient event extractor, but also unlocks the potential of multi-typed event detection – a feature that none of the existing event extractors offers, but is much needed.

Motivation

As a motivation example, consider a sentence extracted from a Wikipedia article, shown in Figure 1 (a). To detect the event type, traditional approaches require identifying the trigger, *sold*, and associating it with an event type – a business acquisition event in this case.

For this particular sentence, we argue that identifying the trigger is not necessary for determining the event type. Figure 1 (b) shows a Compound Value Type (CVT) entry from Freebase (FB) (Bollacker et al. 2008). Here, a CVT organizes complex structured data with multiple *properties* in a table². Using this CVT schema, one can easily map the three arguments of sentence S1, *Remedy Corp*, *BMC Software*, and *2004*, respectively to their properties, `company_acquired`, `acquiring_company`, and `date`. Here each property essentially describes the role of

²Therefore, we also use the term “argument” to refer to a CVT property for the rest of the paper.

an argument in the sentence; and in combination, they define a `business.acquisition` event.

This example shows that CVT information can be used to infer event type without explicit trigger information. If we can utilize such CVT information, we can then label sentences without needing to explicitly identifying the trigger. As a result, we can free annotators from the labour-intensive manual process. In this work, we develop a simple, yet effective technique to automatically generate training data by exploiting the prior CVT knowledge.

A natural question to ask is that “How useful are CVT-like structured information?”. Later in this paper, we show that structured knowledge bases, tables or lists that are used to sum up certain activities can be a useful source of supervision data generation for event extraction – using 24 CVTs, we managed to automatically generate 100 thousands instances from a subset of the Wikipedia articles.

Training Data Generation

Our approach exploits structural information like FB CVT tables to automatically annotate event mentions³ in order to generate training data to learn an event extractor. We use the arguments of a CVT table entry to infer what event a sentence is likely to express. A CVT table entry can have multiple arguments but not all of them are useful in determining the occurrence of an event. For instance, the `divisions_formed` argument in Figure 1 (b) is not as important as the other three arguments when determining if a sentence expresses a `business.acquisition` event.

Our first step for training data generation is to identify the key arguments from a CVT table entry. A **key argument** is an argument that plays an important role in one event, which helps to distinguish with other events. If a sentence contains all key arguments of an entry in an event table (e.g. a CVT table), it is likely to express the event expressed by the table entry. If a sentence is labelled as an event mention of a CVT event, we also record the words or phrases that match the entry’s properties as the involved arguments, with the roles specified by their corresponding property names.

Determining Key Arguments

We use the following formula to calculate the importance score, $I_{cvt,arg}$, of an argument *arg* (e.g., `date`) to its event type *cvt* (e.g., `business.acquisition`):

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

where $\text{count}(cvt)$ is the number of instances of type *cvt* within a CVT table, $\text{count}(arg)$ is the number of times *arg* appearing in all CVT types within a CVT table, and $\text{count}(cvt, arg)$ is the number of *cvt* instances that contain *arg* across all CVT tables. **FIX:explain how do you come up with this formula.**

Our strategy for selecting key arguments of a given event type is described as follows:

³An event mention is a phrase or sentence within which an event is described, including its type and arguments.

Table 1: CVT entry of `business.acquisition` in FB.

| id | company_acquired | acquiring_company | date | divisions_formed |
|-----------|------------------|-------------------|------|------------------|
| m.05nb3y7 | aQuantive | Microsoft | 2007 | N/A |

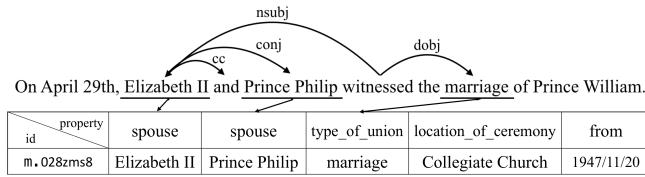


Figure 2: The dependency tree of S3, which partially matches a CVT entry of `people.marriage` from FB.

P1 For a CVT table with n arguments, we first calculate the important score of each argument. We then consider the top half $\lceil n/2 \rceil$ (rounding up) arguments that have the highest importance scores as key arguments.

P2 We find that time-related arguments are useful in determining the event type, so we always include time-related arguments (such as date) in the key argument set.

P3 We also remove sentences from the generated dataset in which the dependency distances between any two key arguments are greater than 2. The distance between two arguments is the minimal number of hops it takes from one argument to the other on the directed dependency tree (see also Figure 2).

Using this strategy, the first three arguments of the CVT entry are considered to be key arguments for event type `business.acquisition`.

Key Argument Selection Parameters

To determine how many arguments should be chosen (step P1 in our strategy), we have conducted a series of evaluations on the quantity and quality of the datasets using different policies. We found that choosing the top half arguments (sorted by their importance scores) gives the best accuracy for event labeling.

We use three example sentences from the Wiki text dataset to explain steps P2 and P3 in our key argument selection strategy described above. The three sentences are:

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.

Although time-related arguments are often missing in the currently imperfect KBs, they are crucial to identify the actual occurrence of an event. As an example, suppose we want to use the CVT entry shown in Table 1 to determine if sentences S2 and S3 are a `business.acquisition` event. While this strategy works for S2, it gives a false prediction for S3. If we add the time-related argument (i.e., date in this

case), we can then correctly label both sentences. We have conducted some initial experiments on our training data during the design phase, and found time-related arguments to be useful. Therefore, we always consider time-related arguments as key arguments (step P2) they are available.

Finally, step P3 is based on our intuitions that two arguments involved in the same event mention are likely to be closer within the syntactic structure. As an example, consider S4. The dependency parse tree of this sentence is given in Figure 2. As can be seen from the diagram, although both *Prince Philip* and *marriage* can be matched as key arguments in a `people.marriage` entry, but with a distance of 3 (i.e., far from each other under our criterion) on the dependency parse tree, thus S4 will be labeled as negative.

Data Generation

To generate training data, we follow a number of steps.

Our approach takes in existing structured tables or lists that are organized in a way similar to the FB CVT tables. The structured tables can be obtained from an existing knowledge base, created by experts, or generated through a combination of both approaches. For each event type, we determine the key arguments for each entry within that type used the key argument selection strategy described above. This step produces a set of rules to be used for data labeling, where each rule contains the event type, key arguments and non-key arguments given by a structured table entry. We also use alias information (such as Wikipedia redirect) to match two arguments that have different literal names but refer to the same entity (e.g. Microsoft and MS).

Next, we label each individual sentence from the target dataset. The labeling process is straightforward. We enumerate all rules from the generate rule set, and check if the target sentence contains all the key arguments specified by a rule. We regard a sentence as a *positive* sample if it contains all the key arguments of a rule, or *negative* otherwise. For instances, S1 in Figure 1 (a) and S2 (with its arguments in italics and underlined) are positive examples, while S3 and S4 are negative.

Because 68% of arguments in our dataset consist of more than one word, we formulate each step in a sequence labeling paradigm rather than word-level classifications. We tag each word of the sentence using the standard begin-inside-outside (BIO) scheme **FIX:()**, 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 call this a **labeling sequence**. For instance, Figure 3 illustrates the labeling sequence for sentence S1 given in Figure 1 (a). As an output, we record the raw text of each training sentence, plus its labeling sequence, and key/non-key arguments.

Limitations

Our approach relies on structured tables or lists to automatically label text. The table can be obtained from an existing knowledge base or hand-crafted by developers. We stress that providing a table incurs much less overhead than manually tagging each training sentence, as a single table entry

| | | | | | |
|--------------|-------------|-----|------------|------------|-------------|
| Remedy | Corp | was | sold | to | BMC |
| B-com..ac. | I-comp..ac. | O | O | O | B-ac..comp. |
| Software | as | the | Service | Management | |
| I-acq..comp. | O | O | B-div..fo. | I-div..fo. | |
| Business | Unit | in | 2004 | . | |
| I-div..fo. | I-div..fo. | O | B-date | | |

Figure 3: The labeling sequence for sentence S1 shown in Figure 1 (a). Each token of the sentence is tagged using the BIO scheme.

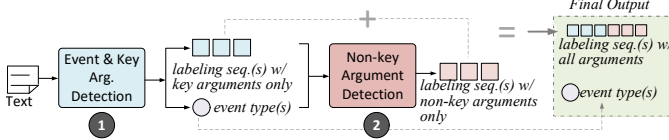


Figure 4: Our 2-stage event extraction pipeline.

can be automatically applied to an unbounded number of sentences.

Like all current event extractors, our approach does not support detecting event types that are not seen in the training data. However, our approach does offer an easy way to extend the structured table to support new types, i.e., by adding new entries to the table.

Our current implementation does not disambiguate pronouns. There are co-reference methods **FIX: ()** that can be used to find out which word a pronoun refers to. These works are therefore complementary to our approach. Integrating our work with co-reference methods is our future work. Finally, while this work targets sentence-level event detection, we believe it is general applicable. There are methods like **FIX:()** can be used to extend our approach to document-level event extraction.

Event Extraction

Unlike prior works, our approach does not rely on explicit trigger information. Instead, it uses key arguments to detect the occurrence of an event. Therefore, we represent problem as a 2-stage pipeline. This is depicted in Figure 4. The input to the pipeline is the raw text. The outputs are the labeling sequence(s) and event type(s) – if any event is detected.

The first stage of the pipeline identifies the 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. This stage outputs a labeling sequence and the corresponding event type (if an event is detected). Since at this stage we do not concern non-key arguments, all non-key argument tokens are tagged as O. Furthermore, multiple labeling sequences may be produced by this stage, each corresponds to an event type. The second stage of the pipeline takes in the outputs of the first stage. It then detects all the non-key arguments of the sentence and tags the non-key argument tokens.

Stage 1: Key Argument and Event Detection

The model used in stage 1 consists of a Bidirectional Long Short-Term Memory (BLSTM) network with a conditional random field (CRF) layer and an Integer Linear Programming (ILP) based post inference. The BLSTM-CRF layer finds the optimal labeling sequence which will then be post-processed by an ILP solver. Using the labeling sequence, an event detector then checks if the sentence mentions a specific event.

BLSTM At each time t , a forward LSTM layer takes \mathbf{x}_t as input and computes the output vector $\vec{\mathbf{h}}_t$ 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 a 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 a given sentence is to choose the best label for each word individually according to the BLSTM output. However, this greedy strategy ignores the dependencies between labels, thus cannot guarantee the best sequence. Therefore, we introduce a CRF layer over the BLSTM output, which is shown to be effective in various sequence labeling tasks (Collobert et al. 2011; Huang, Xu, and Yu 2015).

We consider \mathbf{P} to be a matrix of confidence scores output by BLSTM, 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 \mathbf{w} along with a path of labels $\mathbf{y} = \{y_1, y_2, \dots, y_n\}$ is measured by the sum of BLSTM outputs and transition scores:

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

During test, given a sentence \mathbf{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 The output sequences of BLSTM-CRF do not necessarily satisfy the structural constraints of event extraction. 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 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 \mathbf{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.:

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

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.:

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

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

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

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:

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

where n is the length of the sentence.

Event Detection This step simply maps the input sentence to a specific event type, checking if it contains all the key arguments of that event type by examining the labeling sequence produced by the ILP post-inference layer.

Multi-typed Events There are scenarios where one sentence expresses multiple events which share some key arguments, but most current event extractors only map a sentence to one event. For example, in S5, *Obama* is the officeholder of a government_position_held event and nominee of an award_nomination event.

S5: *Obama* was inaugurated as *president* on *January, 2009*, and was named the *Nobel Peace Prize* laureate nine months later.

One of the advantages of our approach is that it can be easily extended to support multi-typed events. To do so, we allow our ILP solver to output multiple optimal sequences. Specifically, after our model outputs the best sequence s^t at time t , we remove the previously best solutions $\{s^1, \dots, s^t\}$ from the solution space, and re-run our solver to obtain the next optimal sequences s^{t+1} . We repeat the optimization procedure until the difference between the scores of s^1 and s^T is greater than a threshold λ , and consider all solutions $\{s^1, s^2, \dots, s^{T-1}\}$ as the optimal label sequences. We use Gurobi (Gurobi Optimization 2016) as our ILP solver and set $\lambda = 0.5 \times n$, which averagely produce 1.07 optimal sequences for each sentence.

Stage 2: Non-key Argument Detection

After event detection, a sentence will be classified into a or more than one 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 (that is used for event detection) 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, **FIX:because....**

Experiments

Our experiments are designed to answer the following questions, 1) whether it is possible to automatically collect training data for event extraction, 2) whether extractors trained on such data can detect events of interest and identify their corresponding arguments, and 3) whether our solution can work with other knowledge resources for more types of event.

Dataset Evaluation

Let us first evaluate **different data-collecting strategies discussed in Section XXXX**: (1) *ALL*: all arguments are regarded as key arguments; (2) *IMP*: we select the top half arguments with highest importance values as key arguments; (3) *IMP&TIME* means adding a time-related argument with the highest importance value to *IMP*; (4) *DIS*: we eliminate sentences where the dependency distances between any two key arguments are greater than 2. We follow the above methods to collect datasets using Freebase and the English Wikipedia dump of 2016-11-20, randomly select 100 sentences from each dataset, and ask two annotators to decide whether each sentence implies a given type of event.

| No. | Strategy | Sent. | Type | Post. |
|-----|-------------------------|-------|------|-------|
| T1 | <i>ALL</i> | 203 | 9 | 98% |
| T2 | <i>IMP</i> | 318K | 24 | 22% |
| T3 | <i>IMP+DIS</i> | 170K | 24 | 37% |
| T4 | <i>IMP&TIME</i> | 112K | 24 | 83% |
| T5 | <i>IMP&TIME+DIS</i> | 46K | 24 | 91% |

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. *Post.* is the percentage of sentences mentioning the given events explicitly.

As shown in Table 2, it is not surprising that *T1*, as the most strict, guarantees the quality of the collected data, but only contributes 203 sentences covering 9 event types, which is far from sufficient for further applications. *T2* relaxes *T1* by allowing the absence of non-key arguments, which expands the resulting dataset, but introduces more noise. We can also see that the dependency constraint (*DIS*) improves the data quality (*T3*). 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, the dataset by *T5* achieves the best quality, while still accounting for 46735 sentences with 50109 events, almost 8 times more than the ACE dataset, showing that it is feasible to automatically collect quality training data for event extraction without either human-designed event schemas or **extra** human annotations.

| Event types | Trigger candidates | Prop. |
|-------------------|------------------------------------------|-------|
| film_performance | play, appear, star, cast, portray | 0.72 |
| award_honor | win, receive, award, share, earn | 0.91 |
| education | graduate, receive, attend, obtain, study | 0.83 |
| acquisition | acquire, purchase, buy, merge, sell | 0.81 |
| employment_tenure | be, join, become, serve, appoint | 0.79 |

Table 3: Top 5 most frequent trigger candidates and the proportion of their appearance over all positive instances of five event types.

Our final dataset, **FBWiki**, using *T5*, contains 46,735 positive sentences and 79,536 negative ones⁴ a random split of 101,019 for training and 25,252 for testing. Note that there are averagely 4.8 arguments per event, and in total, 5.5% instances labeled with more than two types of events.

Trigger Inference: To further explore the relations between key arguments and triggers, we regard the least common ancestor of all key arguments in the dependency parse tree as a trigger candidate. As listed in Table 3, these trigger candidates share similar lexical meanings and are highly informative to infer their corresponding event types. Though missing explicit information from KB, most triggers can be inferred from the dependencies between a group of key arguments and its context.

On ACE: We also test our strategy on the ACE dataset. We first collect all annotated events, without triggers, as the knowledge base to compute the importance values for all arguments, and select the key arguments for each ACE event type accordingly. We follow *T5* to examine every sentence whether it can be selected as an annotated instance within the ACE event types. Eventually, we correctly obtain 3448 sentences as positive instances, covering 64.7% of the original ACE dataset. We find that the main reason for the missing 35.3% is that many arguments in the ACE dataset are pronouns, where our strategy is currently unable to treat a pronoun as an key argument. However, if a high-precision coreference resolution tool is available to preprocess the document, our solution would be able to automatically label more instances as training data.

Extraction Setup

Next, we evaluate our event extractor on the collected datasets, in terms of popular metrics in event extraction, i.e., precision (P), recall (R), and F-measure (F) for each sub-task. These metrics are computed according to the following standards of correctness: For *event classification*, an event is correctly classified if its reference sentence contains **all key arguments** of this event type. For *key argument detection*, an event is correctly detected if its type and all its key arguments match a reference event within the same sentence.

⁴To train a better system, besides trivial negative samples that have no arguments, we then randomly sample 34,837 negative instances that contain several but not all key arguments, and 21,866 sentences whose key arguments violate the dependency constraint.

For *all argument detection*, an event is correctly extracted if its type and all its arguments match a reference event within the same sentence.

Training: All hyper-parameters are tuned on a development split in the training 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 pre-trained 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.

Baselines: We compare our proposed model with three baselines. The first is a BLSTM model that takes word embeddings as input, and outputs the label for each word with the maximum probability. For feature-based methods, we apply Conditional Random Field (Lafferty, McCallum, and Pereira 2001) (with the CRF++ toolkit (Kudo 2005)) and Maximum Entropy (Berger, Pietra, and Pietra 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, Ji, and Huang 2013).

All models are first required to predict each word with a label indicating whether **it is in an key argument, and add the resulting label of each word as a supplementary feature to predict whether it is in a non-key argument.**

Compare with Automatic Annotations

Firstly, we compare the model output against with the automatically obtained event annotations. As shown in Table 4, 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 position features. 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 in recall, since they can make better use of word semantic features. Specifically, BLSTM can capture longer dependencies and richer contextual information, instead of neighbouring word features only. And the CRF component brings an averagely 2% 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.

The CRF Layer: It is not surprising that every model with a CRF layer over its BLSTM layer is superior to the one with a BLSTM layer only. Compared with vanilla BLSTM, BLSTM-CRF achieves higher precisions and recalls in all subtasks by significantly reducing the invalid labelling sequences (e.g., I-arg appears right after O). During prediction, instead of tagging each token independently, BLSTM-

| Model | Event Classification | | | Key Argument Detection | | | All Argument Detection | | |
|--------------------------------|----------------------|-------------|-------------|------------------------|-------------|-------------|------------------------|-------------|-------------|
| | P | R | F | P | R | F | P | R | F |
| CRF | 88.9 | 11.0 | 19.6 | 36.1 | 4.47 | 7.96 | 19.9 | 3.06 | 5.30 |
| MaxEnt | 95.2 | 12.4 | 21.9 | 41.6 | 5.40 | 9.56 | 22.5 | 3.40 | 5.91 |
| BLSTM | 89.8 | 63.0 | 74.1 | 64.9 | 45.5 | 53.5 | 42.9 | 27.7 | 33.7 |
| BLSTM-CRF | 86.4 | 67.4 | 75.7 | 63.6 | 49.6 | 55.8 | 44.4 | 31.0 | 36.5 |
| BLSTM-CRF-ILP ₁ | 84.4 | 74.1 | 78.9 | 62.3 | 53.8 | 57.3 | 42.7 | 33.8 | 37.7 |
| BLSTM-CRF-ILP _{multi} | 85.3 | 79.9 | 82.5 | 60.4 | 55.3 | 57.7 | 41.9 | 34.6 | 37.9 |

Table 4: System performance when compared with automatic annotations (%).

CRF takes into account the constraints between neighbouring labels, and potentially increases the co-occurrences of key arguments regarding the same event type.

The ILP Post Inference: As shown in Table 4, ILP-based post inference considerably improves the overall system performance, especially in *event type classification*. With the help of constraint **C4**, dubious key arguments can be correctly inferred through other key arguments from their context. Compared with BLSTM-CRF, BLSTM-CRF-ILP₁ produces an F1 gain of 3.2% in event classification, 1.5% in key argument detection, and 1.2% in all argument detection.

Multi-type Events: Among all methods, BLSTM-CRF-ILP_{multi} is the only model that can deal with multi-type event mentions. The proposed strategy in BLSTM-CRF-ILP_{multi} helps detect more event mentions for a sentence, contributing to the increase of recalls, and F1 scores with a little drop of precisions. BLSTM-CRF-ILP_{multi} can correctly identify 132 sentences with multi-type events, with an accuracy of 95.6%, and for each involved event, our model maintains a high performance in identifying its corresponding arguments, achieving the F1 scores of 45.5%, and 29.1% for key argument detection and all argument detection, respectively.

Manual Evaluation

To provide a deep investigation about our dataset and models, we randomly sample 150 sentences from the test set. Two annotators are asked to annotate all arguments to each sentence following two steps. First, determine whether a given sentence is positive or negative, and assign **all existing** event types to positive ones. Next, label all related arguments and their roles according to the event types for all positive instances. Two annotators will independently annotate each sentence, and discuss to reach an agreement. The inter-annotator agreement is 87% for event types and 79% for arguments.

By comparing the automatic and manual annotations on the 150 sentences, we find that the main issue for the automatic annotation is that some automatically labeled sentences do not imply any event while still matching all key properties of certain CVT entries in Freebase. We find 16 such instances that are mistakenly labeled as positive. For example in Figure 5, although the phrase *the car* in S5 matches a film name, it does not indicate that film. This is

mainly because that we currently do not have a strong entity linker to verify those entities, which we leave for future work. But, interestingly, during manual investigation, we find that BLSTM-CRF-ILP_{multi} manages to detect 6 of them as negative.

| Model | EC | KAD | AAD |
|--------------------------------|-------------|-------------|-------------|
| BLSTM-CRF | 75.4 | 54.7 | 35.5 |
| BLSTM-CRF-ILP ₁ | 77.2 | 56.1 | 35.7 |
| BLSTM-CRF-ILP _{multi} | 80.7 | 56.4 | 36.3 |

Table 5: The F1 scores of different systems on the manually annotated data. EC, KAD, AAD denote the event type classification, key argument detection and all key argument detection, respectively.

Table 5 summarizes the performances of different models on the manually annotated dataset, where we can observe similar trends with Table 4, that both CRF and ILP_{multi} greatly improve the overall performance and BLSTM-CRF-ILP_{multi} remains the most effective model.

| | | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|-------------------------|------------------------|
| 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 5: Example outputs of BLSTM-CRF-ILP_{multi}.

Remarkably, our BLSTM-CRF-ILP_{multi} model can find more CVT instances that are currently not referenced in Freebase. Our model detects two events in S6, while the arguments of the second event do not match any existing CVT instances in Freebase, which do not receive any credit during automatic evaluation, but should be populated into Freebase. This phenomenon suggests that by learning from distant supervision provided by Freebase, our model can be used to populate or update Freebase instances in return.

On BBC News: We further apply our event extractor, trained on the FBWiki, to 397 BBC News articles (from 2017/04/18 to 2017/05/18 in Politics, Business and TV sections), and manually examine the extraction results. We find that our model is able to identify 117 events, and 53 events, almost half of which are not covered in the currently used Freebase.

Tables as Indirect Supervision

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

| Event type | Entr. | Sent. | EC | KAD | AAD |
|-------------|-------|-------|------|------|------|
| 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 TBWiki dataset and the performance of our model on TBWiki. *Entr.* is the number of table entries. *Sent.* is number of positive instances.

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.

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, Ji, and Huang 2013) or neural-network-based methods (Chen et al. 2015; Nguyen, Cho, and Grishman 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 auto-

matically 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 $\langle subj, rel, obj \rangle$ 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 *TI*. 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.

Conclusions and Future Work

In this paper, we propose a novel event extraction paradigm without expert-designed event templates by leveraging structured knowledge bases or tables to automatically acquire event schema and corresponding training data. We propose a BLSTM-CRF model with ILP-based post inference to extract multi-typed event mentions without explicit trigger annotations. Experimental results on both manual and automatic evaluations show that it is possible to learn to identify both typed events and typed roles with indirect supervision from Freebase or Wikipedia tables. In the future, we will first further investigate how to better characterize key arguments for a certain event type, and extend our work to update a structured knowledge base according to news events.

References

- Aguilar, J.; Beller, C.; McNamee, P.; Van Durme, B.; Strassel, S.; Song, Z.; and Ellis, J. 2014. A comparison of the events and relations across ace, ere, tac-kbp, and framenet annotation standards. In *Proceedings of the Second Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, 45–53.
- Ahn, D. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, 1–8. Association for Computational Linguistics.
- Berger, A. L.; Pietra, V. J. D.; and Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Computational linguistics* 22(1):39–71.
- Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD 2008*, 1247–1250. ACM.
- Chen, Y.; Xu, L.; Liu, K.; Zeng, D.; and Zhao, J. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL 2015*, volume 1, 167–176.
- Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language pro-

- cessing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.
- Doddington, G. R.; Mitchell, A.; Przybocki, M. A.; Ramshaw, L. A.; Strassel, S.; and Weischedel, R. M. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC 2004*, volume 2, 1.
- Grishman, R., and Sundheim, B. 1996. Message understanding conference-6: A brief history. In *COLING 1996*, volume 96, 466–471.
- Gupta, P., and Ji, H. 2009. Predicting unknown time arguments based on cross-event propagation. In *ACL-IJCNLP 2009*, 369–372. Association for Computational Linguistics.
- Gurobi Optimization, I. 2016. Gurobi optimizer reference manual.
- Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.
- Hong, Y.; Zhang, J.; Ma, B.; Yao, J.; Zhou, G.; and Zhu, Q. 2011. Using cross-entity inference to improve event extraction. In *ACL 2011*, 1127–1136. Association for Computational Linguistics.
- Huang, L.; Cassidy, T.; Feng, X.; Ji, H.; Voss, C.; Han, J.; and Sil, A. 2016. Liberal event extraction and event schema induction. In *ACL 2016*.
- Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Kudo, T. 2005. Crf++: Yet another crf toolkit. *Software available at <http://crfpp.sourceforge.net>*.
- Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, volume 1, 282–289.
- Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.
- Li, Q.; Ji, H.; and Huang, L. 2013. Joint event extraction via structured prediction with global features. In *ACL 2013*, 73–82.
- Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*, 3111–3119.
- Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP 2009*, 1003–1011. Association for Computational Linguistics.
- Mitamura, T.; Yamakawa, Y.; Holm, S.; Song, Z.; Bies, A.; Kulick, S.; and Strassel, S. 2015. Event nugget annotation: Processes and issues. In *Proc. of the Workshop on EVENTS at the NAACL-HLT*, 66–76.
- Nguyen, T. H.; Cho, K.; and Grishman, R. 2016. Joint event extraction via recurrent neural networks. In *NAACL-HLT 2016*, 300–309.
- Rabiner, L. R. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proc. of the IEEE* 77(2):257–286.
- Song, Z.; Bies, A.; Strassel, S.; Riese, T.; Mott, J.; Ellis, J.; Wright, J.; Kulick, S.; Ryant, N.; and Ma, X. 2015. From light to rich ere: annotation of entities, relations, and events. In *Proc. of the Workshop on EVENTS at the NAACL-HLT*, 89–98.
- Zeng, D.; Liu, K.; Chen, Y.; and Zhao, J. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP 2015*, 1753–1762.