Applied Intelligence (2019) 49:206–219
https://doi.org/10.1007/s10489-018-1269-0

Utilizing structured knowledge bases in open IE based event template extraction

Ade Romadhony[1] · **Dwi H. Widyantoro**[1] · **Ayu Purwarianti**[1]

Published online: 22 August 2018

Abstract
Automatic template extraction including event template has been studied intensively in recent years. Researchers study the topic in order to solve the problem of manually defining a template that is required in most information extraction systems. Several studies of event template extraction rely on the documents characteristics to discover the pattern. Although there exist some structured knowledge bases, such as: FrameNet, Predicate Matrix, ACE (Automatic Content Extraction) event type keywords seeds, and FrameNet-ACE event type mapping, no previous researchers have studied combining this information for event template extraction. This paper presents an event template extraction approach that incorporates structured knowledge bases. We propose event template extraction from Open Information Extraction (Open IE) results (relation tuples) in two stages: relation tuple clustering and relation tuple filtering. Both processes utilize structured knowledge bases, as constraint sources in the clustering process and as the basis for the filtering process. The filtering process employs the word embedding representation to capture the semantic relatedness between words. We argue that by involving structured knowledge bases, the relation tuple semantic information can be enriched. Therefore, we can get groups of relation tuples with a similar event sense that represent event templates. The empirical experiment was based on an event argument extraction task and showed that our proposed approach outperforms similar methods that do not use structured knowledge bases. We also compare our proposed system performance to the performance of state-of-the-art systems. The comparison result shows that our proposed system outperforms other state-of-the-art systems, in terms of precision.

Keywords Event template extraction · Open Information Extraction (Open IE) · Structured knowledge base · Constrained clustering · Word embedding

1 Introduction

Automatic event template extraction is the task of discovering event templates and using them to extract event arguments [9]. There are some variations in how to represent an event, ranging from role-based representation, and narrative schemes, to narrative-like knowledge. Chambers and Jurafsky made use of narrative-like knowledge representation by defining a set of related events (event triggers) and semantic roles as can be seen in the following example [9]:

Bombing template
Event trigger: detonate, blow up, plant, explode, defuse, destroy
Event roles
Perpetrator: Person who detonates, plants, blows up
Instrument: Object that is planted, detonated, defused
Target: Object that is destroyed, is blown up

We adopt event, event trigger and event argument definitions from LDC entities and event annotation guidelines [22, 23]. An event is usually described in a span of text (most commonly used: sentence) that contains an event anchor/trigger. Event triggers are one or more words in a sentence that most clearly express an event. Event arguments are entity mentions, time expressions, or values that filled the event semantic roles (slots). In recent years,

✉ Ade Romadhony
ade_romadhony@students.itb.ac.id

Dwi H. Widyantoro
dwi@stei.itb.ac.id

Ayu Purwarianti
ayu@stei.itb.ac.id

[1] School of Electrical Engineering and Informatics Institut Teknologi Bandung, Jl. Ganesha no. 10 Bandung, Indonesia

extracting event templates has been studied intensively because of the increasing variety of document types, especially of those circulating on the Internet. With extensive format and topic variations, defining a template manually is a daunting task.

To solve event template extraction, one commonly used approach is the clustering-based method. Chambers and Jurafsky performed an early study regarding the discovering of event templates in a two-stage clustering: event trigger clustering and slot syntactic function clustering [9]. The experimental result of their work showed that the method successfully extracted event templates. They performed an extra information retrieval (IR) process to build better document representation. However, there were no constraints employed in the clustering process, which could lead to event triggers being placed in the wrong group.

Besides using unstructured knowledge bases such as documents, as an external knowledge base, the use of structured knowledge bases, when they are available, could improve system performance even more. There are several structured knowledge bases that we can use in solving an event argument extraction task: WordNet [29], Predicate Matrix [10] and FrameNet [38]. Previous studies on event-related tasks have employed structured knowledge bases, including [24], who used FrameNet to generalize event triggers, and [40], who used Predicate Matrix as a resource for building an event ontology. However, we have not found an example of structured knowledge base utilization for event template extraction. Our hypothesis is that by using structured knowledge bases that have semantic information, we can improve template quality, especially when the size of the dataset is small.

Although we can determine an event mostly through its trigger, there are several cases where we can not identify its sense based on its trigger, but only based on its arguments. For example, an event could be described by the word *get* as its trigger and *5 years* and *prison* as its arguments. In that case, we need to identify the event by processing its arguments. To our knowledge, there are no previous studies that have taken this issue into account.

In this paper, our main contribution is to propose an approach that utilizes structured knowledge bases for extracting event templates from Open Information Extraction (Open IE) extraction result (relation tuples). Our proposed approach is a two-stage process that is used to identify event patterns, which consists of identification based on an event trigger and based on event arguments. Through our two-stage approach, we perform relation tuple constrained clustering and relation tuple filtering based on relation tuple arguments. The details of our contribution to structure knowledge based utilization in two-stage event template extraction is as follows: 1) creating constraints for clustering Open IE relation tuples based on structured knowledge bases, and 2) proposing a relation tuple filtering process for identifying an event semantic based on its arguments. We employ the knowledge bases from WordNet, Predicate Matrix, FrameNet, Automatic Content Extraction (ACE) event type keyword seeds [6], and FrameNet-ACE event type mapping [26] as the resources for generating the constraints in relation tuple clustering. We defined a simple mapping procedure of these structured knowledge bases, which to the best of our knowledge no previous study on event identification has done, and the experimental results show a positive improvement. Regarding the relation tuple filtering, we perform a process on event arguments, which has rarely been addressed in previous studies, and we propose a new idea for representing the event slots/arguments at the word surface level. We employ FrameNet exemplars as the basis for generating a classifier model for relation tuple filtering, to filter out relation tuples without event sense.

The input of our system is an Open IE extraction result, the relation tuples. Open IE is an information extraction paradigm that applies no restriction to the information type to be extracted [5]. An Open IE system extracts the relation tuple from a sentence, in which a binary relation tuple t is commonly described in the form of $t = (e_i, \texttt{trigger}, e_j)$, where the `trigger` denotes the relation between entity e_i and e_j. For example, from the sentence *"We give everything, we have to be in the final in Accra"*, ReVerb [12], an Open IE system, will extract two relation tuples: *(we, give, everything)* and *(we, have to be in, the final)*. The Open IE result structure is similar to the event structure, which consists of a trigger and arguments and the extraction process in a particular Open IE system (Exemplar [28]) is similar to the event extraction process in several event template extraction works [8, 9, 32]. Until now, Open IE results have been successfully used for a question answering system [14], a search engine [12], fact knowledge bases [7], a thesaurus [31] and an intermediate structure for semantic tasks [43]. Our work is similar to [4], who presented a study on utilizing Open IE results for event schema generation. However, they did not employ structured knowledge bases and only focused on schema coherence in their evaluation.

The rest of this paper is organized as follows. Section 2 presents related work on event template extraction. Section 3 presents an overview of our proposed approach. Section 4 explains the details of the components of our proposed approach. Section 5 describes the experiment. This is followed by a discussion of the result in Section 6. Finally, the conclusion is given in Section 7.

2 Related work

There are two approaches for solving event template extraction: an event trigger clustering-based approach and an event argument-based approach. The clustering-based approach models an event based on an event trigger, while the event argument based models an event based on the entity chain. A previous study by Chambers and Jurafsky employed the clustering-based approach [9], which needs a sufficiently large dataset to generate the event trigger cluster. Chambers and Jurafsky employed an additional IR process to gather a larger dataset.

In the entity chain-based approach, an additional dataset is not needed. Chambers defined the entity chain model and employed an LDA-based method to group similar events [8]. A similar approach was used by Nguyen et al., with an extension in entity processing [32]. In addition to the works by Chamber and Nguyen et al., a work by Sha et al. also used entity-based modeling, but using a normalized cut approach for clustering the templates and slots [41]. Although event argument-based approaches perform better in argument extraction, it is difficult to characterize the event type, because an event type is mostly associated with an event trigger [11].

Since an event type is mostly associated with an event trigger, most studies on event identification and event template extraction give more attention to the event trigger in their methods. However, in several cases, event arguments can also play a significant role in describing the event type. A study by Liu et al. exploited the argument information to improve the event detection task using the supervised attention mechanism [27]. Their proposed method was able to tackle the problem of incorrect event detection caused by ignored event arguments. A similar approach to biomedical event trigger identification was also performed by Li and Liu, and the experimental results show an improvement on F1 [27]. Our proposed approach also exploits argument information on the event detection process, which we found to be especially important in a case where the event trigger has multiple senses, similar to the motivation of Nguyen and Grishman's work, which proposed an argument-aware pooling method on graph convolutional networks [34].

The work of Sha et al. models an event as a joint model of an event trigger and arguments [41]. A similar approach was used by Huang et.al in their proposed IE system, Liberal IE [18]. Liberal IE is performed by defining a model to describe an event, attaching the sense information to the model, and clustering the representations [18]. The results are clusters of vector representations that contain information regarding event triggers and arguments. Huang's study compared the use of several representations: Stanford dependency types, FrameNet, and Abstract Meaning Representation (AMR). The results showed that the best system performance was achieved by using AMR.

Besides using document collections as inputs, several previous studies have used other intermediate structures, including SRL and Open IE relation tuples, for tasks related to event template extraction [4, 13, 36, 47]. However, we have only found one study employing Open IE results for event template extraction, RelGrams [4], which extracts coherent event schemes.

In terms of external knowledge base utilization, most of the previous works we have mentioned rely in their method on information redundancy (word co-occurrence in documents) [4, 8, 9, 18, 20, 32, 41]. A method based on information redundancy has disadvantages for small dataset collection. Our approach was tested on a small dataset, which made obtaining semantic relatedness characteristics challenging. FrameNet, as a structured knowledge base has been studied in relation to improving an event detection task by Kokkinakis and Liu et al. [21, 26]. We incorporate their result (FrameNet - ACE event type mapping) and other structure knowledge bases (WordNet and Predicate Matrix) to generate the constraints in a relation tuple constrained clustering process, and we employ FrameNet sentence examples (exemplars) as the base source for generating a model for relation tuple filtering. By utilizing structured knowledge bases, the semantic relatedness information can be significantly improved.

3 Overview of proposed approach

In this section we provide an overview of our proposed approach. Figure 1 shows the event template extraction process using structured knowledge bases. Our proposed approach uses Open IE relation tuples as the input. A relation tuple consists of two parts: a relation trigger and relation arguments. According to the ACE guidelines [11] and previous studies on event identification, an event consists of an event trigger and event arguments. Since the Open IE relation tuple structure is similar to the event structure, we map relation triggers into event triggers and relation arguments to event arguments.

Our proposed approach for extracting event templates consists of two stages. The first stage is clustering and the second stage is filtering. Both stages involve the process of identifying an event pattern, which is the main goal of event template extraction. For the relation tuple clustering stage, we apply a constrained clustering method. We generate the constraints by defining the event type of a relation tuple. We use the ACE event type as the basis for constraint generation. To obtain the relation tuple event type information, we employ a mapping procedure across several knowledge bases. For each relation tuple,

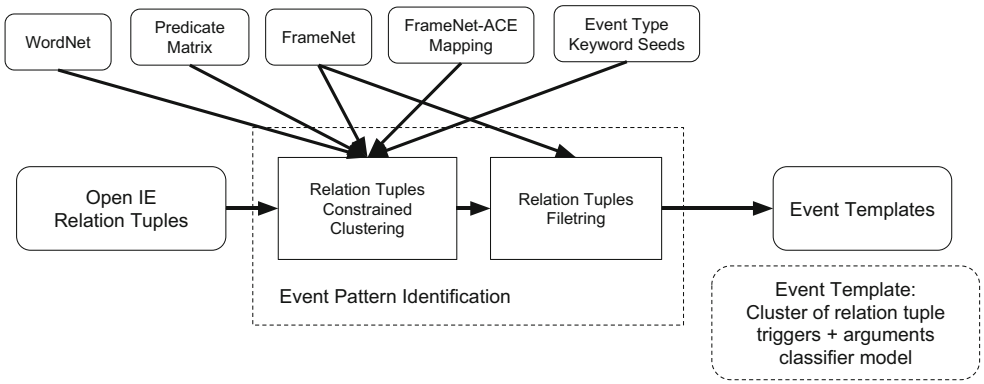

Fig. 1 Overview of utilizing knowledge bases for event template extraction

we assign the corresponding WordNet sense key based on a relation trigger. After we have the WordNet sense key, we look up the corresponding FrameNet frame from the Predicate Matrix. The Predicate Matrix contains mapping information across several knowledge bases, including WordNet and FrameNet. Since we use ACE event types for constraint generation, we need to obtain the information from FrameNet-ACE event type mapping.

Based on each relation tuple event type information, we generate must-link and cannot-link constraints. The must-link constraints consist of relation tuple triggers with the same ACE event type plus the event type keyword seeds from the ACE guidelines. We use the constraints for the clustering of the relation tuples based on their trigger. The clustering results may contain event trigger keywords and multi-sense triggers. Clustering result members that are not listed in the must-link constraint (multi-sense triggers) need to be processed further. In contrast with most previous studies on event identification, which focus on event triggers only [1, 15, 37], we also take event argument mention into account. For this purpose, we need to retain the relation tuples with multi-sense triggers that have arguments describing an event sense. Therefore, we need to filter out relation tuples that have no event sense in their trigger and arguments.

We perform the relation tuple filtering stage by treating it as a classification process. We classify the relation tuple arguments into two classes: with event-sense and without event-sense. To generate the classifier model, we create a training dataset based on sentence examples form FrameNet (Exemplar). The results of our proposed approach are event templates, where each template is represented by a cluster of relation tuples and an argument classifier model for filtering the relation tuples.

4 Event template extraction

In this section we describe in more detail the definition of the event template extraction task and the main components of our proposed approach for event template extraction. The components include inputs, knowledge bases involved, and the two-stage process of event template extraction. The first stage is relation tuple constrained clustering, which involves the following processes: pre-clustering, event-type assignment and constraint generation. The second step is relation tuple filtering, whose end goal is building a classifier model through additional positive and negative training dataset generation.

4.1 Task definition

According to Chambers and Jurafsky, an event template defines a specific type of event and a set of semantic roles that are involved in the event [9]. In extracting an event template, the aim is to perform an event extraction automatically without a predefined template. While Chambers and Jurafsky have defined an event template as a set of related events (words describe an event) and semantic roles [9], we adapt the definition of the semantic role part. Instead of extracting the semantic role based on syntactic function information, we give attention to the word surface feature. Therefore, our goal in event template extraction is to extract templates, in which a template describes a specific event type that consists of similar events (represented by relation tuple trigger) and several words represent suitable arguments/slots for this event type. An example of the *Arrest-Jail* event type template is as follows:

Arrest-Jail template
Event trigger: arrest, jail
Event arguments: police, judge

We assign the argument roles based on the Open IE relation tuple argument roles.

4.2 Open IE

Open IE is an information extraction paradigm that has no restrictions as regards to the type of extracted

information. In contrast with conventional information extraction systems, Open IE needs no definition of the information type that should be extracted. Instead, the Open IE system extracts as much information as possible. Banko et al. introduced the Open IE paradigm with the TextRunner system [5]. From an input sentence, Open IE extracts a set of relation tuples. A relation tuple consists of two parts: a relation tuple trigger and several relation tuple arguments. A binary relation tuple is commonly written in the form $t = trigger(a_1, ..., a_n)$. a_1 and a_n denote the relation tuple arguments, and *trigger* denotes the relation tuple trigger.

Following the TextRunner system, several Open IE systems have been developed, such as: Reverb [12], OLLIE [39], PATTY [31], and Exemplar [28]. The primary difference between these Open IE systems lies in the extraction method. In this work, we use the Exemplar system to extract the relation tuples as input for the template extraction process. We chose the Exemplar Open IE system, since its output form is the most similar to an event structure. For instance, in contrast with other Open IE systems that produce arguments in whole clause form and contain no object type information, Exemplar only outputs the head word of an argument and provides object type information. The argument object type information is a useful guide for determining the event argument type. For example, a POBJ-AT argument of a relation tuple indicates that the argument is the location of an event.

The Exemplar system uses manually defined rules to extract the relation tuples. The features involved in the system include: POSTag, dependency type, and Named Entity. The Exemplar system defines three types of relations based on its trigger type: the verb relation, the verb+noun relation, and the copula+noun relation. To extract the relation tuples, Exemplar performs three main steps: 1) relation trigger identification, 2) relation argument identification, and 3) argument role assignment. In the first step, Exemplar looks for trigger candidates based on a POSTag criterion according to the relation types. In the second step, the argument candidate will be extracted based on the type of dependency relation. In the last step, the system assigns the argument roles based the type of its dependency relation. The system extracts n-ary relation tuples and separates arguments into subject and object. The object argument consists of the direct type object and the preposition object type. Table 1 show examples of Exemplar extraction results from input sentences. We show all the extraction results, including incomplete relation tuples (that have no subject or direct object).

4.3 Knowledge bases

We use information from WordNet, Predicate Matrix, FrameNet, and FrameNet-ACE Event event type mapping to obtain the relation tuple event types. We use WordNet to disambiguate the relation tuple trigger word sense and FrameNet to determine the relation tuple event type.

The main knowledge base used to obtain the event type is FrameNet, since its structure is similar to the event structure and Liu et al. have defined the ACE event type based on the FrameNet frame [26]. However, to determine a corresponding FrameNet frame from a relation tuple trigger, we need to know the trigger word sense. To obtain the relation tuple trigger word sense, we employ WordNet, based on its most frequent 'synset' (synonym set). The event types are obtained through mapping across the knowledge bases and are then used to build the constraints for the event trigger clustering. We also use sentence examples (exemplar) from FrameNet for relation tuple filtering.

WordNet WordNet is a large knowledge base containing English lexicon information. It is the most widely used knowledge base in natural language processing. WordNet organizes lexicon information in a group called 'synsets' (synonym sets). A synset defines the semantic concept (sense) of a word. Words that share the same synset have a synonym relationship. A word may have several senses, in which case it is called a *multi-sense* or *ambiguous* word. WordNet describes *multi-sense* words by assigning several synsets to a word. For example, the word *get* as a verb has 36 synsets. The first synset of *get* has the description *"come into the possession of something concrete or abstract"*, while the second synset has the description *"enter or*

Table 1 Example of extracted relation tuples

Sentence	Relation tuples
This happened at 3pm local time on Monday (2100 UTC), and as many as 14 others were injured before the rampage subsided.	(SUBJ:rampage;subsided; —) (—; injured; DOBJ:others)
One witness said of the gunman , he was " grinning and waving " .	(SUBJ:he;grinning;—) (SUBJ:he;waving;—)
" I looked him in the eye and ran in the room , and that 's when I hid , " Sondra Hegstrom told " The Pioneer of Bemidji " .	(SUBJ:Sondra Hegstrom;told;DOBJ:Pioneer, POBJ-OF:Bemidji) (SUBJI;looked;DOBJ:him,POBJ-IN:eye) (SUBJ:I;ran;POBJ-IN:room),(SUBJ:I;hide;—)

Trigger	Synset	WN Sense Key	FrameNet Frame	ACE Event Type
fight-v,fight-n	contend fight struggle, battle conflict fight engagement	fight%2:33:00, fight%1:04:01	Hostile_encounter: 0.421686746988	Attack

Fig. 2 Example of event type information from knowledge base mapping

assume a certain state or condition". Members of the first synset are: *get* and *acquire*, while members of the second synset are: *become*, *go*, and *get*. Besides providing synset descriptions, WordNet also provides sentence examples for each sense.

Having word sense information makes WordNet a useful resource for word sense disambiguation tasks. In this work, we use WordNet to define a relation tuple sense based on its trigger. Since a word can have different synsets, we should determine a synset as a relation trigger identifier. We picked the synset of a relation trigger based on the most frequent synset (MFS) principle. By using MFS, we choose the first sense of a word, according to the WordNet sense ranking. The MFS principle is the best known method in word sense disambiguation tasks. Several studies on word sense disambiguation have used MFS as the baseline method [17, 19]. A previous study on defining Open IE relation tuple similarity in the Resolver system has also used the MFS principle [46].

Predicate matrix The Predicate Matrix is a knowledge base that provides a mapping across different knowledge bases. The knowledge bases involved are: SemLink, VerbNet, FrameNet, PropBank, and WordNet. We use mapping information across WordNet and FrameNet to get corresponding FrameNet frames from a WordNet sense key.

FrameNet FrameNet is a rich knowledge base that contains information about words by providing their description and associated frames [3]. FrameNet represents the information by following the frame semantic paradigm. A frame corresponds to a scenario, in which several participants are involved, each of them has a specific role. A frame consists of a group of words that have a close semantic relatedness. The words in a frame are called a lexical unit (LU). Different senses of a word are described in different frames. Similar to WordNet, FrameNet also provides sentence examples containing the LU for each frame.

FrameNet-ACE event type mapping In this work, we employ ACE event type as the basis for building the constraints that will be used in relation tuple clustering. The structure of a FrameNet frame is similar to the structure of an ACE event. Based on this similarity, Liu et al. proposed an approach of mapping the FrameNet information onto ACE event types [26]. The mapping method is based on the Probabilistic Soft Logic (PSL) method.

The FrameNet-ACE event type mapping consists of two types of information: frame-ACE events and LU-ACE events. A FrameNet frame can be mapped onto several ACE event types, where each of the mappings has a probabilistic value. The LU-ACE event mapping also applies the same procedure. Similar to our approach of picking the WordNet sense key using the MFS principle, we pick the ACE event type based on the mapping with the highest probability. Figure 2 shows an example of event type assignment to pre-clustered relation tuples.

ACE event type keyword seeds The ACE event annotation guideline provides information about event type descriptions and several sentence examples describing each event type. The event trigger in a sentence is marked in bold. Bronstein et al. has proposed a method to expand the event trigger list based on the event triggers found in the ACE guideline sentence examples [6], which they call *seeds*. They identified words that have a semantic relatedness to the seeds in order to reduce the effort of manual event trigger labeling/annotation. Based on their findings, we believe that event type keyword seeds are very important and must appear in the event template. Therefore, we include event type keyword seeds as a source in the constraint generation process.

4.4 Relation tuple constrained clustering

The goal of relation tuple clustering is to identify relation tuples that have the same event type. Figure 3 shows the processes involved in relation tuple clustering. We use Exemplar[1] [28] as the Open IE system to extract the relation tuples from the input documents. Before clustering relation tuples, we perform a pre-clustering process. The first step of pre-clustering is merging relation tuples that have the same trigger or trigger synset into one relation tuple record. We also merge relation tuples with a trigger in different POSTags (derivationally related forms). For instance, the relation tuple with *abduct* (verb) as relation trigger will be merged with the relation tuple with *abduction* (noun) as trigger. A pre-clustered relation tuples is a set of relation tuples with derivationally related form triggers and synset triggers, $p = trigger_1 \cup trigger_2 \cup ... \cup trigger_n$ $(\{args_1\} \cup \{args_2\} \cup ... \cup \{args_n\})$.

[1] https://github.com/U-Alberta/exemplar

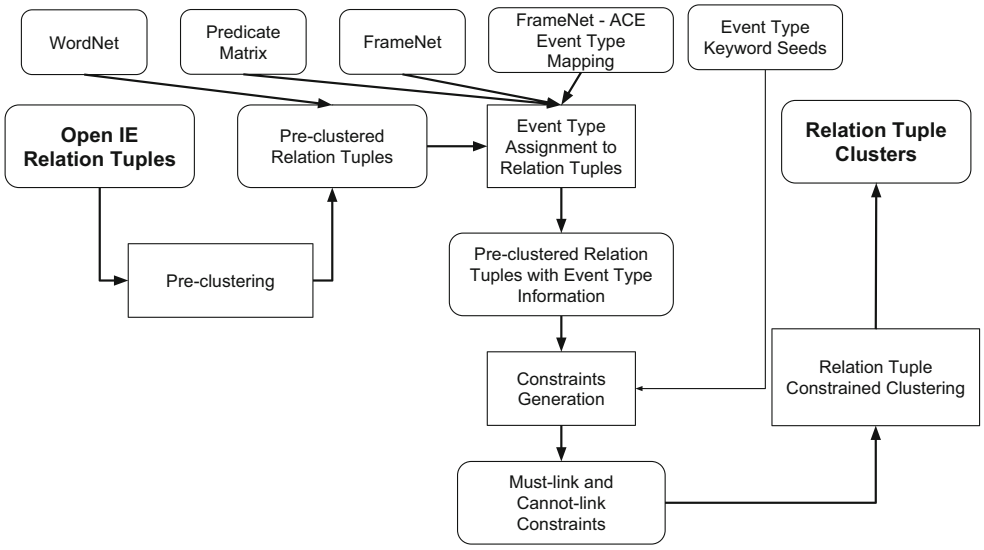

Fig. 3 Relation tuple constrained clustering process

For clustering the pre-clustered relation tuples, we employ instance-level constraints. The instance-level constraints define the rule of instances grouping, i.e. whether instances should be placed in the same cluster or in different clusters [44]. There are two types of instance-level constraints: must-link and cannot-link constraints. Must-link constraints define instances that should be grouped in the same cluster, while cannot-link constraints define instances that should not be grouped in the same cluster.

In order to generate must-link constraints, first, we use a list of ACE event type keyword seeds [6]. The event type keyword seeds are event triggers mentioned in ACE event descriptions.[2] A must-link constraint list contains words describing an event type. For example, a must-link constraint list for the *Charge-Indict* event type contains the following words: *charge*, *indict*, and *accuse*.

Algorithm 1 Algorithm for getting the ACE event type from a relation tuple trigger

Input : trigger tr, WN SenseKey W, PredicateMatrix P, FNACEMapping F, LexicalUnitMapping L

Output: the event type of a relation tuple

procedure GetACEEventType tr, W, P, F, L;
$EvType \leftarrow null$;
$EvType \leftarrow maxprob(F(P(W(\text{tr}))))$;
if $EvType = null$ **then**
    $EvType \leftarrow maxprob(L(\text{tr}))$
end
return $EvType$

[2]We made the modification on the seed list by placing the word fire that originally listed on End-Position event type to Attack event type. We think it is more suitable with the document domain that we use in experiment.

Besides utilizing event type keyword seeds, we make an addition to the must-link constraint list by executing a mapping procedure across knowledge bases. The goal of this procedure is to obtain the event type of pre-clustered relation tuples. Algorithm 1 shows the algorithm to obtain the event type, based on a mapping process from the Predicate Matrix and FrameNet-ACE Event type mapping knowledge bases. The inputs for the algorithm are the pre-clustered relation tuple triggers and their WordNet sense keys. Based on the WordNet sense key, we will obtain the corresponding FrameNet frame ID from Predicate Matrix and then the corresponding ACE event type from the FrameNet-ACE Event Type mapping. If we can not obtain the event type from the FrameNet frame-ACE Event Type mapping, we use the LU-ACE event type mapping.

We add the pre-clustered relation tuple triggers with the same event type to the must-link constraint lists. We discard pre-clustered relation tuple triggers with different event types from the keyword seed event type. Table 2 shows an example of a must-link constraint list generated from event type keyword seeds and knowledge base mapping.

Based on the must-link constraints, we generate the cannot-link constraints. The cannot-link constraints contain trigger pairs from different must-link constraints lists. For instance, the must-link constraint list of the *Charge-Indict* event type contains the words: *charge*, *indict*, while the must-link constraint list of the *Injure* event type contains the words: *injure*, and *harm*. Therefore we put the pairs: *charge* and *injure*, *charge* and *harm*, *indict* and *charge*, and *indict* and *harm* in a cannot-link constraint list.

We performed constrained clustering by employing K-Medoid constrained clustering. The algorithm involves a constraints violation checking procedure as depicted in Algorithm 2. Algorithm 3 is a constrained K-medoid clustering algorithm that we have adapted from

Table 2 Example of must-link constraint list

Event type	Trigger keyword seeds	Constraint list from KB mapping
Arrest-Jail	incarcerate, arresting, arrest, jail, imprison, custody	arrest, collar, imprison, jail, incarcerate, detain, apprehend, nab
Charge-Indict	charge, indict, indictment, accuse	accuse, indict
Convict	convict, guilty	rule, convict, acquit, exonerate, pronounce
Execute	execute	put, set, place, execute
Extradite	extradite, extradition	none

K-means constrained clustering [44]. The violate constraint procedure ensures that the must-link constraint list members are always placed in the same cluster and the cannot-link constraint list members are never placed in the same cluster.

Algorithm 2 Violate Constraints

Input : dataset D, must-link constraints $Con =$, cannot-link constraints $Con \neq$, clustersize k, repetition rep
Output: $ViolateState$
procedure ViolateConstraints $d, C, Con =, Con \neq$;
$ViolateState \leftarrow False$;
repeat
  if $d = not\ in\ Con =$ **then**
    $ViolateState \leftarrow True$;
  end
until *all data item pair* $(d, d =)$ *in* $Con =$ *has been checked*;
repeat
  if $d \neq not\ in\ Con \neq$ **then**
    $ViolateState \leftarrow True$;
  end
until *all data item pair* $(d, d \neq)$ *in* $Con \neq$ *has been checked*;
return $ViolateState$

Algorithm 3 Constrained K-medoid

Input : dataset D, must-link constraints $Con =$, cannot-link constraints $Con \neq$, clustersize k, repetition rep
Output: $C1..Ck$
procedure COPKMedoid $D, Con =, Con \neq, k, rep$;
$C1..Ck \leftarrow initial\_cluster\_center$;
$repcounter \leftarrow 0$;
while *not converge or repcounter¡rep* **do**
  repeat
    Assign each data item di in D except cluster center to closest cluster Cj, such that ViolateConstraints($di, Cj, Con =, Con \neq$) is False;
  until *all data item in D has been assigned to a cluster*;
  repeat
    Update the cluster center in each cluster;
  until *all cluster center in each cluster has been updated*;
  $repcounter \leftarrow repcounter + 1$;
end
return $C1..Ck$

4.5 Relation tuple filtering

A cluster in constrained clustering results may contain multi-sense triggers. We observe that a relation tuple's sense can depend heavily on its arguments. For instance, a relation tuple with *face* as relation trigger and *death penalty* as relation argument has *sentence* sense. Meanwhile, a relation tuple with *face* as relation trigger and *Smith competition* as relation argument has *confront* sense. Two different relation tuples have the same sense if and only if both of them have similar or related arguments. For example, a relation tuple with the word *sentence* as relation trigger is similar to a relation tuple with the word *face* as relation trigger, because both of them have *in prison* as relation argument, which describes the *sentence* sense. Hence, we need to filter out the relation tuples that have different event type senses based on their arguments.

In order to filter out the relation tuples that have a multi-sense trigger, we employ a classification procedure. This procedure classifies whether a relation tuple has an event type sense based on its arguments. To build the classifier model, we use FrameNet exemplar as the initial source for generating the training dataset. Since the exemplar size is small and there is no negative training dataset, we employ the ASTRE relevant dataset [33]. We enlarge the positive training dataset from the ASTRE relevant dataset based on cosine similarity. Meanwhile, for negative training dataset generation, we employ the PU learning approach [25].

To generate the positive training dataset, first, we take the FrameNet exemplars from an event keyword seed frame. For instance, we take the exemplars of LU *sentence* in the *Sentence* frame. The input sentences from FrameNet are fed into the Open IE system, and we pick the extracted relation tuples that contain an event keyword seed as relation trigger, which are called *event relation tuples*. The arguments of event relation tuples, except pronouns, are selected as a positive training dataset. We then calculate the cosine similarity of the positive training dataset and the ASTRE relevant relation tuples arguments. The ASTRE relevant tuple arguments that have a cosine similarity higher than a certain threshold will be selected as an additional positive training dataset.

As for the generation of the negative training dataset, it consists of two steps: potential negative (PN) and reliable negative (RN) dataset creation. Li et al. performed a two-step process because using only the first step results in an insufficient and inaccurate PN dataset [25]. In the PN dataset generation step, we calculate the cosine similarity between the positive dataset and the unlabeled dataset from ASTRE. The ASTRE relevant tuple arguments with cosine

similarity lower than a certain threshold are selected as the PN training dataset. In the next step, RN dataset generation, we employ a Rocchio classifier to classify the unlabeled dataset based on a cosine similarity comparison between the positive and unlabeled dataset, and between the PN and unlabeled dataset. The unlabeled dataset with greater similarity to the PN dataset will be selected as the RN training dataset.

Based on the positive and the RN training dataset, we build the classifier. We represent the argument words as a single vector, employing word embedding. We use word embedding as the representation of the argument word because we aim to get similar words that describe an event argument. We use word embedding as word representation because it can capture semantic similarity or relatedness better than using discrete representation and employing thesauri such as WordNet [16, 35]. For example, the word *jury* is a frequent argument found in Acquit sentence examples. We could not obtain any synonym using WordNet, while based on Google Word2Vec word embedding, we get several related words, such as: *judges*, *court*, and *charge*. We take the word embedding from the GoogleNews word2vec, since our dataset is small, hence using it to train the word embedding does not allow us to capture the word semantic well.

Relation tuple arguments can consist of several words, so we need to perform an operation to obtain a single representation. We perform a linear combination operation, i.e averaging the word vectors, since there is no modifier for argument words. In that case, a linear combination can capture the argument meaning well [42] and the averaging method performs well on word vector combination operation [30].

5 Experiment

We conducted an experiment on event template extraction with and without constraints, and with and without relation tuple filtering. We used the ASTRE dataset, which consists of 16 event types [33]. The ASTRE relevant corpus consists of 1038 documents, which we used to build the relation tuple clusters and to build the argument classifier models in the filtering process. Meanwhile, the ASTRE test corpus consists of 100 documents, annotated with: event trigger words (and also the event type of the trigger), event arguments, and argument co-references.

We used the Pointwise Mutual Information (PMI) similarity measure for the relation tuple clustering. PMI gives a similarity score between two triggers based on co-occurrence. We adapted the PMI score formula from Chambers and Jurafsky [9].

We ran the K-medoid algorithm, with 100 iterations. This is a process for updating the cluster center (medoid). Other parameters, such as the number of K, we set in the range of 80,90,...,150. For cluster centroid/medoid initialization, we chose the KMPP (K-means ++) method [2]. The KMPP method selects the initial cluster centroid randomly. The algorithm then compute the distance between each data point to the selected cluster centroid. The next centroid is updated based on proportional probability to the distance.

The relation tuple filtering is performed by classifying the relation arguments. We employed the linear SVM method to perform the classification process, following the previous work of Li et al. [25].

6 Experimental result

In this section we analyze the effect of structured knowledge base utilization on relation tuple clustering, the performance of relation tuple filtering in relation to event identification, and the application of discovered event templates to event argument extraction.

6.1 Relation tuple clustering result

First, we evaluated cluster quality by adapting CMatch as the metric for cluster evaluation [45]. CMatch measures overlap between two clusters. If two clusters are the same this will result in a CMatch score of 1. In this evaluation we compared the clusters produced in the trigger clustering experiment with the clusters from the ASTRE annotated development dataset. We performed the evaluation of event identification and argument extraction against the annotated ASTRE test dataset, which is annotated with several specific event types. We are aware that the clustering result might contains event types that are not among annotated dataset event types, however, we only conduct an evaluation based on listed event types. A cluster in the ASTRE annotated dataset contains words describing an event type. For instance, the *Sentence* cluster consists of the following words: *sentence*, *get*, and *carry*.

$$CMatch(Cr, Ct) = \frac{\left\|\varepsilon(Cr) \bigcap \varepsilon(Ct)\right\| \cdot \sum_{c \epsilon \{\varepsilon(Cr) \bigcap \varepsilon(Ct)\}} Score(c)}{\|\varepsilon(Cr) \cup \varepsilon(Ct)\|} \tag{1}$$

$$Score(trigger) = \frac{\sharp trigger\_occurrence}{\sharp event\_type\_occurrence} \tag{2}$$

Formula (1) shows the CMatch calculation we used in this work. Let *Cr* be a cluster of trigger of the relation tuple clustering process result and *Ct* a cluster from the annotated

dataset. The CMatch formula computes the overlap number between the *Cr* members and *Ct* members, and multiplies it with the overlap word/trigger score. The trigger score, as shown in formula (2), is a fraction of the number of trigger occurrences in the ASTRE annotated development dataset. A word has a higher score if it is annotated as an event trigger more frequently. For example, in the dataset, there are 60 *Sentence* events that are described by the following words: *sentence*, *get*, and *carry*. The word *sentence* was annotated 50 times, *get* five times, and *carry* five times. Therefore, *sentence* has the highest score.

Table 3 shows the highest average CMatch comparison between constrained K-medoid clustering and non-constrained K-medoid clustering. The scores in the table were averaged over 100 trials, with K cluster size = 100. We picked the cluster with the highest CMatch score for each event type as the representative cluster. The result shows that constrained K-medoid clustering had better cluster quality compared to non-constrained clustering. Exceptions occurred in the case of the *die* and *execute* event types. The reason for the lower CMatch value for those event types is that we used the referenced cluster (Ct) from the annotated development dataset. Meanwhile, there are several event keywords that exist in the test dataset only. As a result, the overlap of those keywords was not calculated.

To analyze the effect of constrained clustering, we measured recall in the event trigger identification. The use of constraints in event trigger clustering affects the clustering result by forcing triggers that should be grouped into one cluster to always stay in the same cluster. Event trigger identification with constrained clustering should result in the same or a higher recall value than without constraint clustering. Without using constraints, event triggers that should be grouped into the same cluster could be spread over different clusters. We performed event trigger identification evaluation specifically for each event type.

Table 4 shows a comparison between the recall values of constrained clustering and non constrained clustering. Based on the recall values of each event type, we can see that constrained clustering produced more complete cluster members. An exception occurred in the case of the *Sentence* event type: its recall value was slightly lower in constrained clustering, because the must-link constraint on the *Sentence* event type only consists of one word, i.e. *sentence*. Meanwhile, the *Sentence* event type can be described using different words and their co-occurrence frequency is high, which makes the probability of these words being grouped into one cluster in non constrained clustering higher.

6.2 Relation tuple filtering result for event identification

Our proposed method outputs are event templates, where each template consists of event triggers and an event argument classifier model. Event argument classifier classifies whether relation tuple arguments hold a particular event type semantic. Figure 4 shows the event templates that were described by trigger keywords and several arguments that represent their event type, which were classified as a positive argument by the argument classifier.

Table 3 CMatch comparison

Event type	Non-constrained clustering	Constrained clustering
Sentence	0.334	**0.369**
Arrest-Jail	0.338	**0.498**
Attack	0.127	**0.177**
Injure	0.104	**0.126**
Trial-Hearing	0.001	**0.005**
Charge-Indict	0.006	**0.010**
Convict	0.002	**0.003**
Release-Parole	0.002	**0.007**
Appeal	0.001	**0.010**
Extradite	**0.003**	**0.003**
Sue	**0.002**	**0.002**
Die	**0.004**	0.003
Pardon	0.003	**0.006**
Fine	**0.003**	**0.003**
Execute	**0.003**	0.002
Acquit	**0.001**	**0.001**
Average	0.058	**0.076**(+**31**%)

Table 4 Event trigger identification recall comparison

Event type	Without constraint	With constraint
Sentence	**0.645**	0.629
Arrest-Jail	0.261	**0.369**
Attack	0.108	**0.349**
Injure	0.383	**0.545**
Trial-Hearing	0.328	**0.481**
Charge-Indict	0.350	**0.729**
Convict	**0.167**	**0.167**
Release-Parole	0.842	**0.850**
Appeal	**0.875**	**0.875**
Extradite	**0.667**	**0.667**
Sue	0.187	**0.365**
Die	0.095	**0.626**
Fine	0.495	**0.500**
Execute	0.800	**0.867**
Acquit	0.000	**0.003**
Average	0.414	**0.535**(+**29**%)

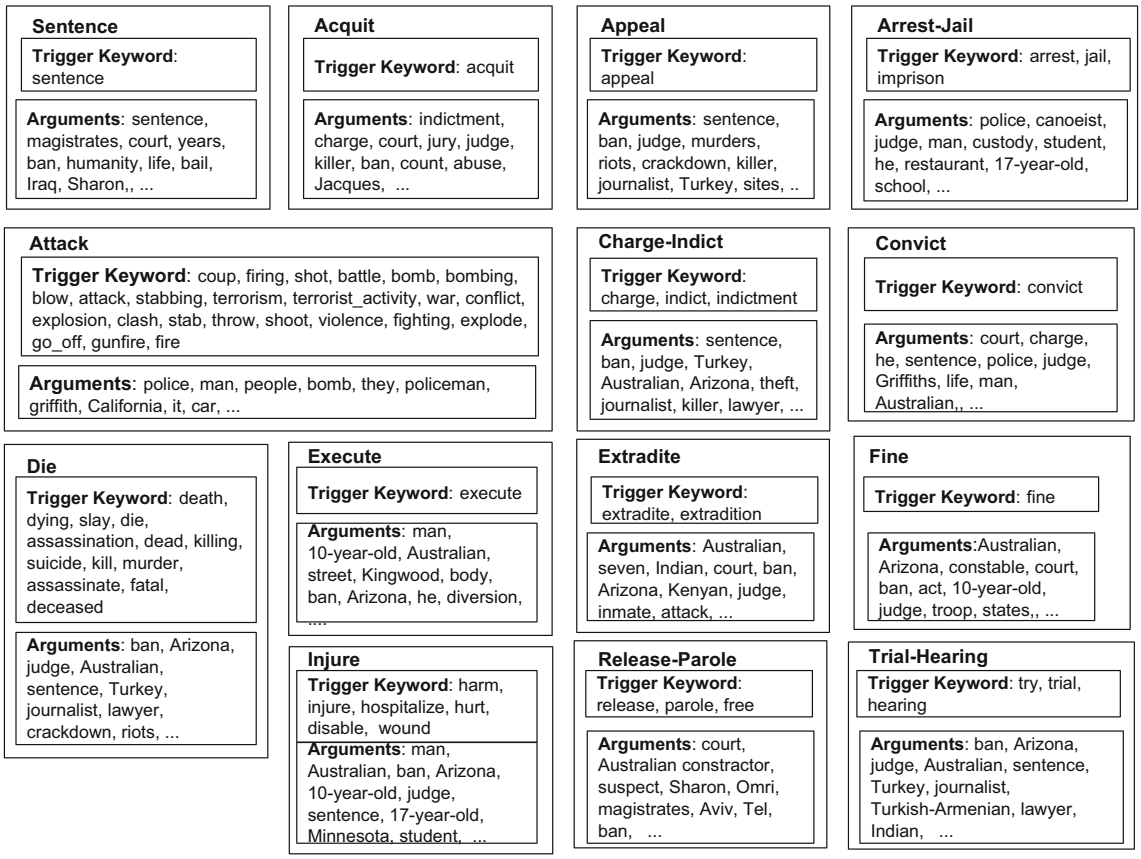

Fig. 4 Event template illustration

To evaluate relation tuple filtering, we compared event identification performance with and without filtering. Evaluating event identification, we ignored event type information and treated all annotated events as a single event type. Table 5 shows a comparison between the performance of event identification with and without relation tuple filtering.

The result shows that the filtering process increased the precision value, both in constrained and non-constrained clustering. The improved precision value indicates that filtering out relation tuples without event sense made the event template more accurate. On the other hand, the recall value decreased slightly. The lower recall value indicates that the filtering process missed several relation tuples with event sense. We observe that those relation tuples have pronouns as relation arguments, therefore the classifier failed to classify them as having event sense. However, the percentage of precision improvement is higher than the decrease in recall. As a result, the overall performance as denoted by F1 value still shows a positive improvement.

6.3 Event argument extraction result

In order to evaluate the event template, we used it in an event argument extraction task. We took the relation tuple arguments from the relation tuples that were identified

Table 5 Event identification evaluation

Metric	Non constrained clustering		Constrained clustering	
	Without relation tuple filtering	With relation tuple filtering	Without relation tuple filtering	With relation tuple filtering
Precision	0.21	**0.46**(+**119**%)	0.24(+14%)	**0.46**(+**119**%)
Recall	0.45	0.39(−13%)	**0.58**(+**29**%)	0.54(+20%)
F-1	0.28	0.42(+50%)	0.34(+21%)	**0.50**(+**79**%)

Table 6 Event argument extraction evaluation

Metric	Non constrained clustering		Constrained clustering	
	Without relation tuple filtering	With relation tuple filtering	Without relation tuple filtering	With relation tuple filtering
Precision	0.13	**0.22**(+**69**%)	0.16(+23%)	**0.22**(+**69**%)
Recall	0.13	0.12(−8%)	**0.17**(+**31**%)	**0.17**(+**31**%)
F-1	0.13	0.15(+15%)	0.18(+38%)	**0.19**(+**46**%)

as events. In this evaluation, we also ignored event type information.

Table 6 shows the result of the argument extraction evaluation. Similar to the event identification results, we found an improvement in all performance metrics by employing relation tuple constrained clustering and an improvement in precision by applying relation tuple filtering. However, the precision increase was lower than the precision increase in event identification. The reason for this is that the Open IE relation tuple arguments and their co-references are incomplete. However, in general, constrained clustering and filtering have a significant effect on improving the template quality as the argument extraction performance result shows.

We also undertook a performance comparison with state-of-the-art systems for event template extraction [8, 32]. Table 7 shows the performance comparison for event argument extraction. We took the previous systems performance based on the experimental result in the ASTRE dataset paper [33], the same dataset that we use in our experiment. The result shows that our proposed system outperforms the other systems on precision. However, the recall falls behind the previous systems recall, which also causes the F-1 to be lower than that of the other systems. We examined that the low recall on argument extraction is due to incomplete Open IE relation tuple arguments and that our proposed system process used a different approach. Our system uses an event model that is based on an event trigger. On the other hand, the state-of-the-art systems used a different approach, employing the entity/argument chain as the anchor of the event model. The argument-based approach has a higher probability to extract more complete arguments.

Table 7 Performance comparison with state-of-the-art systems

System	Precision	Recall	F-1
Chambers,2013	15	**28**	19
Nguyen et.al,2015	21	26	**23**
Proposed System	**22**	17	19

6.4 Error analysis

The event identification and event argument extraction evaluation shows that several relation tuples were misidentified. Based on our observation, this is due to limitations in our approach and Open IE.

In the clustering process, our approach missed several relation tuples with triggers that actually have event sense. The reason is that our approach depends heavily on constraint list completeness. Words that have close semantic relatedness but rarely co-occur may not be grouped in the same cluster based on PMI measure. In that case, the constraints should help. However, if the words are not listed in the must-link constraints, they could be grouped in different clusters. For example, the words *blast* and *struck* were not identified as event triggers, because they have low PMI scores compared to the other event type keywords and were not listed in the constraint list. Therefore, *blast* and *struck* were not found in the *Attack* main cluster. We also notice that several words were found in the test dataset only, such as *repatriate*. As a result, our system could not identify them in the clustering process.

On the other hand, because of a limitation of the Open IE system that we used, some relation tuples with event sense were not extracted. There are two reasons for this. First, the Open IE system only extracts verbs and nouns as relation triggers, while several annotated triggers in the key dataset are adjectives. For instance, we found the words *injured*, *guilty* and *dead* as event trigger on the key dataset. Secondly, the Open IE system that we used is based on manually defined rules. The rules are based on POSTag and dependency type information. There are several relation tuples that could not be extracted due to the rule limitation. For example, the trigger *killed* from the sentence *Eight killed in Dagestan* could not be extracted because the dependency type between *eight* and killed is acl, which was not listed in the extraction rules.

As for the argument extraction errors, based on our observations, the reason for these is that the Open IE system unable to extract all valid arguments. This also causes low recall in co-reference argument identification, because co-reference depends on the relation tuple argument extraction results.

7 Conclusion

In this paper, a method of utilizing structured knowledge bases in clustering-based event template extraction performed on Open IE relation tuples was presented. The knowledge bases were used as constraints in relation tuple clustering and as the source for the training dataset that was used to generate the classifier model for the relation tuple filtering. The experimental result showed that the use of structured knowledge bases improves event trigger identification and event argument extraction performance. Our proposed method is also able to discover relation tuple patterns describing an event type.[3]

Despite the fact that the experiment results shows that our proposed method of structured knowledge base utilization outperformed similar methods without structured knowledge base utilization, it still has several limitations. The first limitation is incomplete information from the knowledge bases: the constraints still did not include several event identifiers. Moreover, the relation tuple filtering could not identify several words describing an event. Another limitation lies in the Open IE extraction process. The cause of the low recall value in argument extraction is that the Open IE system is unable to extract relevant triggers describing events and that argument information from Open IE relation tuples is incomplete.

In future work, we will explore a method to learn the Open IE relation tuple representation as a single vector representation. In the method proposed here, we still divide the relation tuple representation into two parts: event trigger and event argument. By representing a relation tuple using one single representation, we may get relation tuple characteristics based on its trigger and arguments simultaneously.

Acknowledgements This work was funded by Institut Teknologi Bandung, under the P3MI program.

References

1. Altmeyer R, Grishman R (2009) Active learning of event detection patterns. Proteus Project Technical Report, pp 09–014
2. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, society for industrial and applied mathematics, pp 1027–1035
3. Baker CF, Fillmore CJ, Lowe JB (1998) The berkeley framenet project. In: Proceedings of the 17th international conference on computational linguistics, vol 1. Association for Computational Linguistics, pp 86–90
4. Balasubramanian N, Soderland S, Mausam OE, Etzioni O (2013) Generating coherent event schemas at scale. In: EMNLP, pp 1721–1731
5. Banko M, Cafarella MJ, Soderland S, Broadhead M, Etzioni O (2007) Open information extraction from the web. In: IJCAI, vol 7, pp 2670–2676
6. Bronstein O, Dagan I, Li Q, Ji H, Frank A (2015) Seed-based event trigger labeling: how far can event descriptions get us?. In: ACL, vol 2, pp 372–376
7. Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka Jr ER, Mitchell TM (2010) Toward an architecture for never-ending language learning. In: AAAI, vol 5. Atlanta, p 3
8. Chambers N (2013) Event schema induction with a probabilistic entity-driven model. In: EMNLP, vol 13, pp 1797–1807
9. Chambers N, Jurafsky D (2011) Template-based information extraction without the templates. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, vol 1. Association for Computational Linguistics, pp 976–986
10. De Lacalle ML, Laparra E, Rigau G (2014) Predicate matrix: extending semlink through wordnet mappings. In: LREC, pp 903–909
11. Doddington GR, Mitchell A, Przybocki MA, Ramshaw LA, Strassel S, Weischedel RM (2004) The automatic content extraction (ace) program-tasks, data, and evaluation. In: LREC, vol 2, pp 837–840
12. Etzioni O, Fader A, Christensen J, Soderland S, Mausam M (2011) Open information extraction: the second generation. In: IJCAI, vol 11, pp 3–10
13. Exner P, Nugues P (2011) Using semantic role labeling to extract events from wikipedia. In: Proceedings of the workshop on detection, representation, and exploitation of events in the semantic web (DeRiVE 2011). Workshop in conjunction with the 10th international semantic web conference, pp 23–24
14. Fader A, Zettlemoyer L, Etzioni O (2014) Open question answering over curated and extracted knowledge bases. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 1156–1165
15. Feng X, Huang L, Tang D, Ji H, Qin B, Liu T (2016) A language-independent neural network for event detection. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: short papers), vol 2, pp 66–71
16. Goikoetxea J, Agirre E, Soroa A (2016) Single or multiple? Combining word representations independently learned from text and wordnet. In: AAAI, pp 2608–2614
17. Grycner A, Weikum G (2014) Harpy: hypernyms and alignment of relational paraphrases. In: 25th International conference on computational linguistics. ACL, pp 2195–2204
18. Huang L, Cassidy T, Feng X, Ji H, Voss CR, Han J, Sil A (2016) Liberal event extraction and event schema induction. In: ACL (1)
19. Izquierdo R, Suárez A, Rigau G (2007) Exploring the automatic selection of basic level concepts. In: Proceedings of RANLP, vol 7
20. Jiang T, Sha L, Sui Z (2014) Event schema induction based on relational co-occurrence over multiple documents. In: Natural language processing and chinese computing. Springer, pp 23–33
21. Kokkinakis D (2012) Initial experiments of medication event extraction using frame semantics. In: Scandinavian Conference on Health Informatics 2012; October 2-3; Linköping; Sverige, Linköping University Electronic Press, vol 070, pp 41–47
22. (LDC) LDC (2005) English annotation guidelines for entities. https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-entities-guidelines-v5.6.6.pdf, visited 20-July-2018
23. (LDC) LDC (2005) English annotation guidelines for events. https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf, visited 20-July-2018

[3]The dataset used in experiment could be accessed in https://github.com/aromadhony/kb-openie-eventtemplate

24. Li Q, Ji H, Hong Y, Li S (2014) Constructing information networks using one single model. In: EMNLP, pp 1846–1851
25. Li XL, Liu B, Ng SK (2010) Negative training data can be harmful to text classification. In: Proceedings of the 2010 conference on empirical methods in natural language processing. Association for Computational Linguistics, pp 218–228
26. Liu S, Chen Y, He S, Liu K, Zhao J (2016) Leveraging framenet to improve automatic event detection. In: ACL (1)
27. Liu S, Chen Y, Liu K, Zhao J (2017) Exploiting argument information to improve event detection via supervised attention mechanisms. In: Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: long papers), vol 1, pp 1789–1798
28. Mesquita F, Schmidek J, Barbosa D (2013) Effectiveness and efficiency of open relation extraction. New York Times 500:150
29. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
30. Mitchell J, Lapata M (2008) Vector-based models of semantic composition. In: Proceedings of ACL-08: HLT pp 236–244
31. Nakashole N, Weikum G, Suchanek F (2012) Discovering and exploring relations on the web. Proc VLDB Endowment 5(12):1982–1985
32. Nguyen KH, Tannier X, Ferret O, Besanċon R (2015) Generative event schema induction with entity disambiguation. In: ACL, vol 1, pp 188–197
33. Nguyen KH, Tannier X, Ferret O, Besanċon R (2016) A dataset for open event extraction in english. In: LREC
34. Nguyen TH, Grishman R (2018) Graph convolutional networks with argument-aware pooling for event detection. In: The Thirty-Second AAAI conference on artificial intelligence (AAAI-18)
35. Niraula NB, Gautam D, Banjade R, Maharjan N, Rus V (2015) Combining word representations for measuring word relatedness and similarity. In: FLAIRS Conference, pp 199–204
36. Peng H, Song Y, Roth D (2016) Event detection and co-reference with minimal supervision. In: EMNLP, pp 392–402
37. Qin B, Zhao Y, Ding X, Liu T, Zhai G (2010) Event type recognition based on trigger expansion. Tsinghua Sci Technol 15(3):251–258
38. Ruppenhofer J, Ellsworth M, Petruck MR, Johnson CR, Scheffczyk J (2006) Framenet ii: extended theory and practice
39. Schmitz M, Bart R, Soderland S, Etzioni O et al (2012) Open language learning for information extraction. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. Association for Computational Linguistics, pp 523–534
40. Segers R, Laparra E, Rospocher M, Vossen P, Rigau G, Ilievski F (2016) The pred-icate matrix and the event and implied situation ontology: making more of events. In: Proceedings of GWC2016
41. Sha L, Li S, Chang B, Sui Z (2016) Joint learning templates and slots for event schema induction. In: Proceedings of NAACL-HLT, pp 428–434
42. Socher R, Huval B, Manning CD, Ng AY (2012) Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. Association for Computational Linguistics, pp 1201–1211
43. Stanovsky G, Dagan I et al (2015) Open ie as an intermediate structure for semantic tasks. In: Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 2: short papers), vol 2, pp 303–308
44. Wagstaff K, Cardie C, Rogers S, Schrödl S et al (2001) Constrained k-means clustering with background knowledge. In: ICML, vol 1, pp 577–584
45. Widyantoro DH (2004) Concept drift learning and its application to adaptive information filtering. PhD thesis, Texas A&M University
46. Yates AP, Etzioni O (2009) Unsupervised methods for determining object and relation synonyms on the web. Journal of Artificial Intelligence Research
47. Zhou W, Zhang Y, Su X, Li Y, Liu Z (2016) Semantic role labeling based event argument identification. Int J Database Theory Appl 9(6):93–102