BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 20 2012, pages 2654–2661
doi:10.1093/bioinformatics/bts487

Data and text mining

Advance Access publication August 1, 2012

A robust approach to extract biomedical events from literature

Quoc-Chinh Bui* and Peter M.A. Sloot

Computational Science, Informatics Institute, Faculty of Science, University of Amsterdam, The Netherlands

Associate Editor: Jonathan Wren

ABSTRACT

Motivation: The abundance of biomedical literature has attracted significant interest in novel methods to automatically extract biomedical relations from the literature. Until recently, most research was focused on extracting binary relations such as protein–protein interactions and drug–disease relations. However, these binary relations cannot fully represent the original biomedical data. Therefore, there is a need for methods that can extract fine-grained and complex relations known as biomedical *events*.

Results: In this article we propose a novel method to extract biomedical events from text. Our method consists of two phases. In the first phase, training data are mapped into structured representations. Based on that, templates are used to extract rules automatically. In the second phase, extraction methods are developed to process the obtained rules. When evaluated against the Genia event extraction abstract and full-text test datasets (Task 1), we obtain results with *F*-scores of 52.34 and 53.34, respectively, which are comparable to the state-of-the-art systems. Furthermore, our system achieves superior performance in terms of computational efficiency.

Availability: Our source code is available for academic use at http://dl.dropbox.com/u/10256952/BioEvent.zip

Contact: bqchinh@gmail.com

Received on May 8, 2012; revised on June 28, 2012; accepted on July 28, 2012

1 INTRODUCTION

Automatic relation extraction is an important and established way to extract information from biomedical literature (Ananiadou *et al*., 2010). Many relation extraction approaches have been proposed to extract binary relations between biomedical entities from text such as protein–protein interactions (Giles and Wren, 2008), drug–drug interactions (Tari *et al*., 2010) and causal relations on drug resistance (Bui *et al*., 2010). However, such binary relations cannot fully represent the biological meaning of the original relations (Cohen and Hunter, 2006). As a consequence, there is a need to have a better representation that can exemplify complex relations extracted from text. Recently, this shortcoming has been addressed in the BioNLP'09 Shared Task (Kim *et al*., 2009) by introducing the biomedical event extraction task, which aims to extract complex and nested events from text. In this shared task, an event is defined as follows: each event consists of a trigger, a type and one or more arguments. Depending on the event type, an argument can either be a protein or another event [see (Kim *et al*., 2009) for more details] as illustrated in Figure 1. Figure 1a shows examples of three event types where a simple event (E1) has one protein as the argument, a binding event (E2) has two proteins as the arguments and a regulatory event (E3) has two events as the arguments.

Several event extraction approaches have been proposed to the BioNLP'09 and BioNLP'11 challenges (Kim *et al*. 2009, 2011). In general, to extract events from a given sentence, two steps need to be carried out: identifying event triggers and determining their arguments. First, the sentence is typically preprocessed and converted into structured representations such as dependency parse tree and candidate event triggers are often pre-selected using a dictionary. Next, the candidate event triggers and the parse tree are used as the input for the event extraction process. The proposed approaches to extract events can be divided into two main groups, namely, rule-based and machine learning (ML)-based approaches.

Rule-based event extraction systems consist of a set of rules that are manually defined or generated from training data. To extract events from text, each pre-selected candidate trigger is first validated to determine its event type. Next, the defined rules are often applied to the parse tree which contains that trigger to find relations between the trigger and its arguments (Bui and Sloot, 2011; Kilicoglu and Bergler, 2009). Evaluation results from the BioNLP'09 and BioNLP'11 show that rule-based systems achieve high precision but low recall. In particular, the system proposed by Cohen*et al*. (2009) achieves the highest precision on the BioNLP'09 event extraction track.

ML-based systems model event extraction tasks as a classification problem. In these systems, pre-selected candidate event triggers and the arguments attached to them are classified as true event triggers or not. Extraction methods proposed in the BioNLP'09 exploit various learning algorithms and feature sets to leverage the system performance (Björne *et al*., 2009; Buyko *et al*., 2009). Methods proposed after the BioNLP'09 can be divided into two groups based on how event triggers and arguments are determined. The first ML-based group consists of systems that adopt the pipeline and feature sets proposed by Björne *et al*. (2009) and later improved by Miwa *et al*. (2010), in which the evaluation of the candidate triggers and the determination of their arguments are carried out independently. For this type of approach, errors made in the trigger detection step propagate into subsequent steps. Examples of such systems are Björne and Salakoski (2011), Miwa *et al*. (2012) and Quirk *et al*. (2011). To overcome this issue, the second ML-based group uses joint learning models in which event triggers and their arguments are jointly predicted. Systems belong to this group are Poon and Vanderwende (2010), Riedel and Andrew (2011) and Vlachos and Craven (2011). Overall, ML-based systems achieve

*To whom correspondence should be addressed.

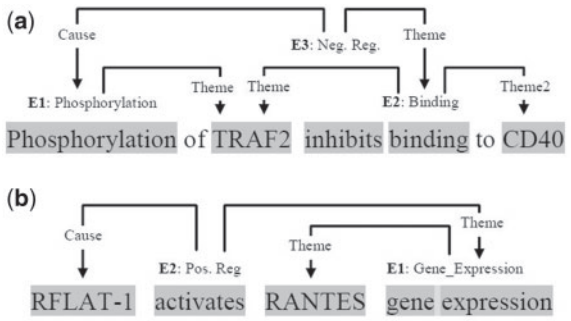

Fig. 1. (**a**) Examples of three event types: simple (E1), binding (E2), and regulatory (E3) events. **(a,b)** Variants of regulatory event arguments: the event E3 in Figure 1a has two events as the arguments whereas the event E2 in Figure 1b has one protein and one event as the arguments

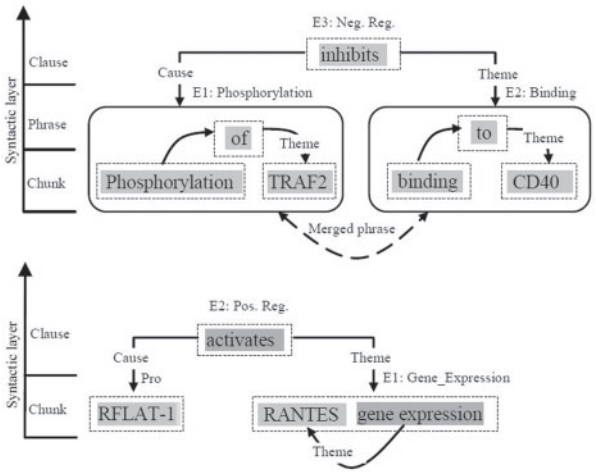

Fig. 2. Structured representations for the events in Figure 1

good results in terms of *F*-scores on both BioNLP'09 and BioNLP'11 challenges.

Apart from these two main approaches, there are a few systems that use different methods. Móra *et al*. (2009) propose a hybrid method that combines both rule- and ML-based approaches. Liu *et al*. (2011) use a system using sub-graph matching. Neves *et al*. (2009) use case-based reasoning system. McClosky *et al*. (2011) introduce a system using dependency graphs by extending the function of an existing dependency parser. However, the performance of these systems is not comparable to the state-of-the-art ML-based systems.

In this article we introduce a novel rule-based method to extract biomedical events from text. Our method differs in many ways from the existing methods. First, we define a new structured representation to express the relations between event triggers and their arguments. Next, we transform the input text into this structured representation using a shallow parser. We then map the annotated events from the training data into these structured representations, based on that rules are extracted automatically by using predefined templates. The obtained rules are unified to make them more flexible. Finally, we develop event extraction algorithms to apply these unified rules. By using the structured representation to decompose the nested and complex structures of events into simple syntactic layers, we can use simple methods to learn and apply extraction rules. When evaluated on the BioNLP'11 test set, our system achieves an impressive performance in terms of precision and computational times. To the best of our knowledge, this is the first system of its kind to extract biomedical events from text.

2 METHODS

Our method to event extraction consists of two phases: a learning phase and an extraction phase. In the learning phase, the system learns rules to extract events from training data. In the extraction phase, the system applies rules learned in the previous phase to extract events from unseen text. A text preprocessing step is used in both phases to convert unstructured text into structured representations. In the following sections, we present these steps in more detail.

2.1 Structured representation

We define the following syntactic layers that form a structured representation to express the structures of biomedical events:

Chunk: is the base syntactic layer that can express an event (see Figure 2). A chunk consists of one or more tokens, taken from the output of a shallow parser (see Section 2.5 for more detail). Four chunk types that are used to express events are as follows: adjective, noun, verb and preposition.

Phrase: consists of a list of chunks, connecting by preposition chunks (see Figure 2). A phrase can express one or more events where the event triggers and their arguments should belong to different chunks. To reduce the variants when learning rules and to simplify the event extraction process, all noun chunks that are in the coordinative form are merged into one noun chunk. For example, the following chunks: [NP PRO1], [NP PRO2], [CONJ and] and [NP PRO3] are merged into one noun chunk [NP PRO1, PRO2 and PRO3].

Clause: consists of a verb chunk that connects with a left child and a right child (see Figure 2). A child of a clause is a phrase but it can also be a chunk. A clause can express one or more events where the event trigger belongs to the verb chunk and its arguments belong to the clause's children.

Merged phrase: is a special noun phrase obtained by merging the verb chunk of a clause with its children. The merged phrase is used to express an event where its trigger and its arguments belong to the left child and the right child of a clause. For example, the binding event E2 in Figure 1 needs a merged phrase to be expressed.

Relation between layers: An upper layer can query its direct lower layer to find arguments. The returned values of the query can be a list of proteins, a list of events or an empty list. For example, a phrase can query its right chunks. A clause can query its left and right children.

Representing biomedical events: with this structured representation, we can express most of biomedical events from the training data. Here, both event triggers and their arguments must belong to the same sentence. Figure 2 presents the events from Figure 1 using the structured representations.

2.2 Learning rules

To learn extraction rules from structured representations, we define a list of syntactic and semantic features to capture the underlying rules that govern the relations between event triggers and their arguments when

expressed by structured representations. These features are divided into three groups that correspond to three syntactic layers: chunk, phrase and clause. The descriptions of these features are as follows.

2.2.1 Features for events expressed in a chunk layer:

Frequency: counts frequency of an event trigger when expressed in chunk form.

Part-of-speech (POS): indicates which syntactic form (e.g. NN, ADJ) of an event trigger is used in the chunk form.

2.2.2 Features for events expressed in the phrase layer:

Frequency: counts frequency of an event trigger when expressed in phrase form.

POS: indicates which syntactic form (e.g. NN, NNS and VBG) of an event trigger is used in the phrase form.

Preposition: indicates which preposition is used to connect the chunk that contains an event trigger with the chunk containing its first argument (*theme*), e.g. [NP *expression*] ***of*** [NP PRO1].

Distance: measures the distance (by chunk) from the chunk that contains an event trigger to the chunk containing its 'theme'.

Preposition2: is used for binding and regulatory events. It indicates which preposition is used to connect the chunk that contains the 'theme' to the chunk that contains the second arguments (i.e. *theme2* for binding event or *cause* for regulatory event) of an event. For example, the *Preposition2* of the binding event: [NP *binding*] **of** [NP *PRO1*] **to** [NP *PRO2*] is **to**.

Distance2: is used for binding and regulatory events. It measures the distance (by chunk) from the chunk that contains an event trigger to the chunk that contains its second argument.

Cause order: is used for regulatory events. It indicates the relative position between the theme and the cause of an event. For example, consider a regulatory event: effect of *PRO1* on *PRO2* where PRO1 is the cause. This feature helps determining the argument which is closer to the trigger is a theme or a cause.

Theme position: is used for binding events. It indicates the position of the theme is on the left or on the right of an event trigger. For examples, PRO1 binding to PRO2, interaction between PRO1 and PRO2.

Theme 2 position: is similar to theme position but for the second argument.

2.2.3 Features for events expressed in the clause layer:

Frequency: counts frequency of an event trigger when expressed in clause form

POS: indicates which syntactic form (e.g. VBZ, VBD) of an event triggers is used in the clause form.

Passive: indicates whether the clause in the 'active' or the 'passive' form.

Theme position: indicates the theme is on the left child or on the right child when the clause in the active form.

Distance, Distance2: these features are similar to those of phrase form.

Beside these features, we use the following features to determine the number of arguments and argument types for binding and regulatory events.

2.2.4 Specific feature for binding events:

Theme2Count: counts frequency of an event trigger having *theme2*.

Algorithm 1 Learning extraction rules from annotated events

Input: Sentence **S,** list of events **E** belongs to **S**, a dictionary **D**, an empty hashtable **H** for each event type, template **T**
Output: List of rules for each event type.

```
(1)  Convert S into structured representation R
(2)    For e ∈ E
(3)      If event trigger of e in D
(4)        If e can be mapped to layer r ∈ R
(5)          Fill t ∈ T with features extracted from r
(6)            Generate key k for t
(7)            If k ∈ H
(8)              Update entry in H
(9)            Else
(10)           Store k into H
(11)     Return H
```

2.2.5 Specific features for regulatory events:

ThemePro: counts frequency of an event trigger having theme as a protein.

ThemeEvent: counts frequency of an event trigger having theme as an event.

CausePro: counts frequency of an event trigger having cause as a protein.

CauseEvent: counts frequency of an event trigger having cause as an event.

To learn extraction rules, we create three templates using the above features where each template corresponds to a layer (i.e. chunk, phrase). Next, we map each annotated event from the training data into a suitable layer. The extraction rule for each event is then extracted using a corresponding template. The learning procedure is shown in Algorithm 1.

The text conversion step used in this algorithm is described in Section 2.5. During this learning process, some events are skipped if their event triggers do not belong to a dictionary that is prepared based on single-word event triggers collected from the training and development datasets. In addition, events are skipped if they cannot be mapped into the structured representation.

2.3 Rule combination

For each event trigger and its specific POS tag, we may obtain one to three sets of extraction rules which correspond to three layers: chunk, phrase and clause. For example, the 'binding' trigger has three POS tags: ADJ, NN and VBG and therefore it may possess up to nine sets of extraction rules. Examples of extraction rules of the binding trigger and its POS tag NN, which learned from the phrase layer, are shown in Table 1. The first row of this table is an extraction rule that learned from binding events such as the binding event E2 in Figure 2.

To simplify the event extraction step and to make the extraction rules generalize well, we combine the extraction rules for phrase and clause layer of each event trigger and its specific POS tag to generate a unified rule. For consequent process, each unified rule is represented as a decision table. These decision tables are then used in the extraction step to determine event arguments. For example, Table 2 shows a decision table that results from the combination of all the extraction rules in Table 1. In addition, for each decision table of an event trigger, we calculate a

confidence score based on the frequency of that event trigger being extracted in a layer (e.g. chunk, phrase) divided by frequency of that event trigger being found in the training data. These specific feature values such as *Theme2Count* and *ThemePro* for binding and regulatory events are then normalized based on the frequencies of the combined rules.

Since simple, binding and regulatory events have different argument types and each event class has two sets of extraction rules that need to be combined (chunk layer has only one extraction rule), we need six variants of the decision tables to represent these unified rules. However, the procedures that are used to combine these rules are similar. Here, we use a simple voting scheme based on frequencies to select the rules that are more likely reliable and remove those that contradict to the selected ones. An example of such rule combination algorithm is shown in Algorithm 2.

Table 1. Examples of extraction rules of the binding trigger and its POS tag NN obtained from the phrase layer. Some features are omitted for clarity

POS	Preposition	Preposition 2	Distance	Distance 2	Frequency
NN		to	2	2	6
NN		of	2	4	1
NN	of		14	0	49
NN	of	to	2	8	21
NN	of	in	2	6	1
...	...				
NN	between	and	2	2	1

Table 2. A decision table of the binding trigger for extracting events of the phrase layer

Feature	Value	Feature	Value
Rule type	*Phrase*	Distance 1	*14*
POS	*NN*	Distance 2	*8*
Trigger	*Binding*	Theme position	*right; left*
Theme2Count	*0.35*	Preposition	*of; to; in; by;...*
Confidence score	*0.28*	Preposition 2	*of:[to, in];...*
Theme type	*Protein*	Theme 2 position	*right*

Algorithm 2 Combining extraction rules of the regulatory events obtained from the clause layer

Input: List of extraction rules **R**, list of event trigger **E**.
Output: Decision tables for each event trigger

(1) **For** $e \in \mathbf{E}$
(2) Retrieve rules **L** of e **in R**
(3) Split **L** into two groups **G1**, and **G2** based on *passive* feature
(4) Sort **G1** and **G2** based on *frequency*
(5) Select *active direction* dr_1, dr_2 for **G1** and **G2**.
(6) **For** i = 1 to 2
(7) **For** $g \in \mathbf{G_i}$
(8) Remove $r \in \boldsymbol{g}$ **if** $r_{active\ direction} \neq dr_{i\ active\ direction}$
(9) Set distance *d1, d2* for $\mathbf{G_i}$
(10) Calculate ThemPro, ThemeEvent, CausePro, CauseEvent for $\mathbf{G_i}$

2.4 Event extraction

In this section we present algorithms to extract events from an input sentence. First, the sentence is preprocessed to detect candidate event triggers and is converted into structured representations. Candidate event triggers are detected using the same dictionary as the one used in the learning rules step. Next, we evaluate each candidate trigger and determine its arguments by using a decision table. For each candidate trigger, its decision table is retrieved using a key which consists of the trigger, its POS, the layer containing it and its event type. For example, if the binding trigger belongs to a phrase and it has POS tag NN, then its retrieval key is *binding_NN_Phrase_Binding*.

We model the process of extracting events from a sentence as a pairing task where each protein may be paired with a trigger on the left or on the right of that protein based on a given rule. For example, consider the sentence in Figure 1b: RFLAT-1activates RANTES gene expression, with the flat representation, it is not clear whether *RANTES* should pair with the 'activates' trigger or with the 'expression' trigger. However, when mapped into structured representation as show in Figure 2, the decision can easily be made based on the syntactic layer. In this case, pairing *RANTES* with the expression trigger should have a higher priority than pairing it with the activates trigger. Based on the structured representation, our event extraction procedure consists of three steps: extract events from chunk, from phrase and from clause. At each layer, the extracted events can be used by its direct upper layer as arguments. The event extraction algorithm is shown in Algorithm 3.

2.5 Text preprocessing

The text preprocessing step consists of splitting sentences, replacing protein names with place-holders, tokenizing words, POS tagging, parsing sentences with a shallow parser and converting the output of the parser into structured representations. To split the input text (e.g. title, abstract, paragraph) into single sentences, we use the LingPipe sentence splitter (http://alias-i.com/lingpipe/). Sentences that do not contain protein names are skipped. We replace protein/gene names with a place-holder, e.g. PRO_i (i is the index of the i^{th} protein/gene in the text) to prevent the parser from segmenting multiple-word protein names and for subsequent processing. Each sentence is then tokenized and tagged with the LingPipe POS tagger. Finally, these outputs (tokens and their POS tags) are parsed with the OpenNLP shallow parser (http://incubator.apache.org/opennlp/) to produce chunks.

2.5.1 Converting chunks into structured representation: We adapt the method described in Segura-Bedmar *et al.* (2011) to convert chunks output from the parser into structured representations. Here complex clauses are split into multiple simple clauses. To reduce the variants of the structured representations, coordinative noun chunks are merged into one noun chunk as mentioned in Section 2.1. Furthermore, if there is an adjective chunk that immediately follows a verb chunk, we merge that adjective chunk into the verb chunk. For example, the following chunks [VB is] [ADJ dependent] are merged into [VB is dependent]; the merged chunk is considered as a verb chunk.

3 RESULTS AND DISCUSSION

3.1 Datasets

We use the Genia Event Extraction datasets provided by the BioNLP'11 (https://sites.google.com/site/bionlpst/ home/genia-event-extraction-genia) to evaluate our extraction method.

Algorithm 3 Extracting biomedical events from a sentence

Input: Sentence **S**, a list of proteins **P**, a list of extraction rules **R**, a dictionary **D**, a list of candidate triggers **G**.
Output: list of extracted events.

Sub functions
isChunk(trigger g, protein p): return **True** if g and p can form an event
queryArgs(layer l, trigger g, rule r): query l for a list of protein/events based on the data of r and g.
formEvent(trigger g, List L1, List L2): form events based on g, L1, and L2
hasCause(trigger g, rule r): return **True** if has Cause/Theme2
inRange(trigger g, rule r, trigger g2): return **True** if events formed by g2 can be used as arguments for g.

ChunkExtraction(List **G,** Structure **T)** / / extract events from chunk layer.

(1) **For** $g \in$ **G**
(2) Retrieve *chunk rule* r of g **in** R
(3) Locate chunk l ∈ **T** containing g
(4) List **L1** = **queryArgs**(l, g, r)
(5) **If** $r \neq$ ***null*** and **isChunk**(g, p ∈ **L1**)
(6) **formEvent**(g, p, null)
(7) **Else If** *r* **is null** and **isChunk**(g, p∈**L1**)
(8) Remove g

PhraseExtraction(Trigger **g,** List **G,** Structure **T) / /** extract events from phrase layer

(1) **Repeat**
(2) List **L1** = null;
(3) List **L2** = null;
(4) g = **getNext**(**G**) / / get next trigger
(5) Retrieve *phrase rule* r of g **in R**
(6) Locate phase l ∈ **T** containing g
(7) **L1** = **queryArgs**(l,g,r) / / theme
(8) **If** r ≠ **null** and L1 = **null**
(9) g2 = **getNext**(**G**) / / get next trigger
(10) **If inRange**(g,r,g2)
(11) **L1** = **PhraseExtraction**(g2,**G**,**T**);
(12) **If** L1 ≠ **null**
(13) **If hasCause**(g, r)
(14) L2 = **queryArgs**(l,g,r) / / cause / theme2
(15) **formEvent**(g,**L1**,**L2**)
(16) **Until** empty(**G**)

ClauseExtraction(List **G**, Structure **T**) / / extract events from clause layer

(1) **For** $g \in$ **G**
(2) List **L1** = null; / / theme
(3) List **L2** = null; / /cause / theme 2
(4) Retrieve *clause rule* r of g **in R**
(5) Locate clause l ∈ T containing g
(6) **L1** = **queryArgs**(l, g, r)
(7) **If** $r \neq$ ***null*** and L1 ≠ **null**
(8) **If hasCause**(g, r)
(9) **L2** = **queryArgs**(l,g,r) / / cause / theme2
(10) **formEvent**(g, **L1**, **L2**)

The datasets include training, development and test datasets and each dataset consists of two parts, an abstract dataset and a full-text dataset. For training and development datasets, a list of proteins/genes and annotated events is given. For the test set, only a list of proteins/genes is given. Statistics of the datasets are shown in Table 3.

Main function

(1) Convert s into structured representation **T**
(2) Map p∈ **P** into chunks ∈ **T**
(3) **ChunkExtraction**(**G**,**T**)
(4) **PhraseExtraction**(null,**G**,**T**)
(5) **ClauseExtraction**(**G**,**T**)

Table 3. Statistics of training, development, and test datasets

Items	Training	Development	Test
Abstracts + full papers	800 (+5)	150 (+5)	260 (+4)
Sentences	8759	2954	3437
Proteins	11 625	4690	5301
Events	10 310	3250	4457
Availability of events	Yes	Yes	No

Values in parentheses denote the number of full papers.

3.2 Evaluation settings

We use both training and development datasets for building the dictionary and for learning extraction rules, which are then used to extract biomedical events from the test dataset (Task 1). The extracted events are submitted to the online evaluation system (https://sites.google.com/site/bionlpst/home) to evaluate the results. To obtain realistic results, we do not apply any tuning techniques to optimize for *F*-score of the test dataset. We set the thresholds for dictionary entries and confidence scores of the extraction rules to 0.1 and 0.03, respectively. These threshold values are determined empirically based on the development dataset.

3.3 Event extraction

Table 4 shows the results of our extraction method evaluated on the test dataset using the 'Approximate Span/Approximate Recursive matching' criteria. We present the evaluation results of the abstract and full-text datasets in parallel to easily analyze the results of both types of text. The data show that our system performs well on both abstract and full-text datasets. In particular, it achieves the best results on simple events (SVT-TOTAL), followed by binding events and regulatory events (REG-TOTAL). Overall, the results on the full-text dataset are better than on the abstract dataset. The results on simple and binding events in the full-text dataset are significantly better than in the abstract dataset with 8–10 *F*-score points higher, whereas the results on the regulatory events in the abstract dataset are slightly better than in the full-text dataset.

Tables 5 and 6 present comparison results of our system and the top four systems that participated in the BioNLP'11 challenge. To study the results of each system in more details, we also

Table 4. Evaluation results of the test dataset

Event class	Abstract			Full text		
	R	P	F	R	P	F
Gene_Expr.	63.43	85.29	72.76	76.43	88.43	81.99
Transcription	58.39	81.63	68.09	40.54	100	57.69
Pro_catabolism	42.86	60.00	50.00	100	100	100
Phosphorylation	71.11	96.00	81.70	88.00	95.65	91.67
Localization	47.70	97.65	64.09	64.71	55.00	59.46
SVT-TOTAL	61.17	87.11	71.87	74.03	87.96	80.39
Binding	39.48	64.93	49.10	56.25	65.32	60.45
EVT-TOTAL	56.25	82.61	66.93	69.19	81.70	74.92
Regulation	20.62	52.17	29.56	17.02	42.11	24.24
Pos_regulation	32.55	56.44	41.29	29.57	51.13	37.47
Neg_regulation	24.54	49.21	32.75	25.52	49.49	33.68
REG-TOTAL	28.61	54.31	37.48	26.94	49.88	34.99
ALL-TOTAL	41.89	69.72	52.34	44.47	66.63	53.34

R, P and F denote recall, precision and *F*-score, respectively.

Table 5. Performance comparison on the abstract test dataset

System	SVT	BIND	REG	Total		
	F	F	F	R	P	F
Riedel (2011)	71.54	50.76	45.51	48.74	65.94	56.05
Björne (2011)	70.36	47.50	44.30	50.06	59.48	54.37
Quirk (2011)	70.08	43.86	40.85	48.52	56.47	52.20
Kilicoglu (2011)	67.75	37.41	40.96	43.09	60.37	50.28
Ours	71.87	49.10	37.48	41.89	69.72	52.34

present the evaluation results on each dataset separately, where Table 5 shows the results on the abstract test dataset and Table 6 shows the results on the full-text test dataset.

Table 5 shows that our system achieves good results compared to the best system on simple (SVT) and binding (BIND) events. In fact, it performs slightly better on simple events than those systems. However, the performance on regulatory (REG) events is lower than the best system, with a gap of 12 *F*-score points. Overall, on the abstract dataset, the Riedel and Andrew (2011) system is the best since their system yields the highest *F*-score, whereas our system yields good results in terms of precision and the Björne and Salakoski (2011) system achieves good results in terms of recall.

With the inclusion of full-text documents in the test dataset, the BioNLP'11 brings more challenges to the event extraction task than the BioNLP'09 does. This requires the adaption ability of each system since the structure and content of biomedical abstracts and full-text bodies are different (Cohen *et al*., 2010). The results in Table 6 show that all systems yield better *F*-scores on the simple events in the full-text dataset than in the abstract dataset. Our system still leads on this type of events. Interestingly, while the other systems drop the performance on

Table 6. Performance comparison on the full-text test dataset

System	SVT	BIND	REG	Total		
	F	F	F	R	P	F
Riedel (2011)	79.18	44.44	40.12	47.84	59.76	53.14
Björne (2011)	76.98	34.39	39.16	48.31	53.38	50.72
Kilicoglu (2011)	78.39	35.56	38.12	44.71	57.75	50.40
Quirk (2011)	75.63	36.14	38.21	48.94	50.77	49.84
Ours	80.39	60.45	34.99	44.47	66.63	53.34

the binding events, our system gains 11 *F*-score points, from 49.11% for the abstract dataset to 60.45% for this dataset. This yields the best result on binding events reported so far. An analysis of the results on the binding events of these systems (data not shown) showed that while two rule-based systems achieve nearly the same precision on the binding events in both datasets (e.g. 49.76% versus 49.38% for Kilicoglu's system; 64.93% versus 65.34% for our system), three ML-based systems drop precision significantly (e.g. 60.89% versus 47.62 for Riedel's system; 50% versus 31.76% for Björne's system; 44.51% versus 32.77% for Quirk's system). This might be due to over-fitting since the number of binding events which is available for training in the abstract dataset is much higher than in the full-text dataset (881 events vs. 101 events). This implies that, for binding events, the aforementioned rule-based systems generalize better than their counterpart ML-based systems. This finding was also pointed out by Kilicoglu and Bergler (2011). For regulatory events, our system drops the performance on this dataset but the gap of results between our system and the best system on these event classes is reduced to 6 *F*-score points. Overall, our system outperforms the best system of the BioNLP'11 on full-text dataset in terms of both precision and *F*-score.

3.4 Computational performance

When the system is applied to large-scale extractions such as the whole PubMed database or used in Question–Answer systems as envisioned by Wren (2011), then computational resources required to run the system should be taken into account. Despite this obvious fact, only few systems report on the computational time needed to run their systems. Riedel and McCallum (2011) report that their system needs from 60 to 297 ms, depending on the learning models, to extract events from a sentence. However, these numbers do not include the parsing times and feature extraction times. Björne *et al*. (2010) report in their large-scale experiment that, on average, their system needs 954 ms to parse a sentence and needs 486 ms to extract events from that sentence. Since Riedel and Andrew (2011) use the same parser as Björne *et al*. (2010), we assume that the parsing times of the two systems are equivalent. Therefore, both systems need from 1040 to 1400 ms to extract events from a sentence. In contrast, our system needs 6 ms to do so. Details of the computational times are shown in Table 7.

In general, it is not straightforward to directly compare the computational times between systems due to the differences in

Table 7. Computational times of the system on the test set and training set

Dataset	# sentences processed	Text prepossessing (average)	Event extraction (average)
Test set	2067	5.9 ms	0.49 ms
Training set	5186	4.9 ms	0.68 ms

The experiments are run on a PC with Core-i5 2.3 MHz CPU, 4 GB of memory.

hardware as well as other factors, e.g. the length of sentences and the number of events per sentence. These differences are shown in Table 7 where the text processing times and event extraction times vary on two datasets even though they run on the same system. However, it is apparent that our system outperforms the mentioned systems with 150-fold faster in terms of computational time.

3.5 Performance analysis

There are some issues that affect our system performance, mostly in terms of recall. In the following section we address these issues and discuss possible directions to improve the overall performance.

First, the use of the dictionary to detect candidate event triggers and to disambiguate event triggers affects the performance considerably. We found that raising the threshold of the dictionary entries would increase precision for some event classes but this would also decrease recall of the others. There are many cases where event triggers may belong to more than one event class and they cannot be determined by simply increasing the dictionary threshold. This problem was also discussed by Cohen *et al*. (2011). Furthermore, our extraction algorithm relies on both candidate triggers and proteins to extract events. If some candidate triggers are filtered by the threshold, then the procedure used to find event arguments fails to return a desired proteins/events list.

Another factor that affects the system performance is the rule combination step. While combining extraction rules definitely simplify the event extraction method, it also causes the loss of information. As shown in Algorithm 2 (line 8), during the combination process, we remove some rules that contradict to the selected rules. This increases precision but decreases the recall of the system. Furthermore, the loss of information is clearly visible in the case of binding and regulatory events. For example, the *CausePro*, and *CauseEvent* values in the decision table of an event trigger can provide statistical data for that event trigger such as indicating how often that trigger may have a cause as a protein or as an event. However, these data do not indicate in which particular case that trigger has a cause. When analyzing 50 false-positive regulatory events obtained from the developing dataset, we found that of 22 cases due to wrong event classes (e.g. Pos_Regulation vs. Regulation) and of 28 cases due to wrong number of arguments (e.g. with or without causes) or wrong argument types (e.g. protein vs. event). Therefore, to reduce the false-positive regulatory events, we need a better strategy such as adding more specific features for these event classes or modify the current extraction algorithm to retain more rules.

4 CONCLUSION

In this article we have proposed a novel rule-based method to extract biomedical events from text. Our core method to event extraction is the use of a structured representation to decompose nested and complex events into syntactic layers. This representation not only allows us to simplify the learning and extraction phases but also it requires less syntactic analysis of input sentences. The evaluation results show that our system performs well on both abstract and full-text datasets. Furthermore, it achieves superior performance in terms of computational efficiency. It is clearly suited to large-scale experiments.

Our event extraction method is simpler than the existing ML-based approaches. It is also more robust than the previously proposed rule-based approaches since it learns rules automatically from training data. Its simplicity and robustness have been proven by the performance on simple and binding events. Its structured representation is generic and is capable of representing any relation types. The proposed feature sets based on the structured representation mainly consist of generic features and a few specific features. Therefore, it is suited to extract many types of relations. If needed, its specific features can be easily adapted to any new domain.

ACKNOWLEDGEMENTS

The authors sincerely thank Dr Makoto Miwa and Rick Quax for their useful comments.

Funding: This research was supported by the European Union through the DynaNets project http://www.dynanets.org EU grant agreement no. 233847.

Conflict of Interest: none declared.

REFERENCES

Ananiadou,S. *et al*. (2010) Event extraction for systems biology by text mining the literature. *Trends in Biotechnol*., **28**, 381–390.

Björne,J. *et al*. (2009) Extracting complex biological events with rich graph-based feature sets. In: *Proceedings of BioNLP'09 Shared Task Workshop,* pp. 10–18.

Björne,J. *et al*. (2010) Complex event extraction at PubMed scale. *Bioinformatics (Oxford, England)*, 26, i382–i390.

Björne,J. and Salakoski,T. (2011) Generalizing biomedical event extraction. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 183–191.

Bui,Q.C. and Sloot,P.M.A. (2011) Extracting biological events from text using simple syntactic patterns. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 143–146.

Bui,Q.C. *et al*. (2010) Extracting causal relations on HIV drug resistance from literature. *BMC Bioinformatics*, **11**, 101.

Buyko,E. et al. (2009) Event extraction from trimmed dependency graphs. In: *Proceedings of BioNLP'09 Shared Task Workshop. Association for Computational Linguistics*, Morristown, NJ, pp. 19–27.

Cohen,K.B. *et al*. (2011) High-precision biological event extraction: effects of system and of data. *Comput. Intelligence*, **27**, 681–701.

Cohen,K.B. *et al*. (2009) High-precision biological event extraction with a concept recognizer. In: *Proceedings of BioNLP'09 Shared Task Workshop,* pp. 50–58.

Cohen,K.B. *et al.* (2010) The structural and content aspects of abstracts versus bodies of full text journal articles are different. *BMC Bioinformatics*, **11**, 492.

Cohen,K.B. and Hunter,L. (2006) A critical review of PASBio's argument structures for biomedical verbs. *BMC Bioinformatics*, **7** (Suppl. 3), S5.

Giles,C.B. and Wren,J.D. (2008) Large-scale directional relationship extraction and resolution. *BMC Bioinformatics*, **9** (Suppl. 9), S11.

Kilicoglu,H. and Bergler,S. (2011) Adapting a general semantic interpretation approach to biological event extraction. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 173–182.

Kilicoglu,H. and Bergler,S. (2009) Syntactic dependency based heuristics for biological event extrac-tion. In: *Proceedings of BioNLP'09 Shared Task Workshop. Association for Computational Linguistics*, Morristown, NJ, pp. 119–127.

Kim,J.D. et al. (2009) Overview of BioNLP'09 shared task on event extraction. In: *Proceedings of BioNLP'09 Shared Task Workshop. Association for Computational Linguistics*, Morristown, NJ, pp. 1–9.

Kim,J.D. *et al.* (2011) Overview of Genia Event Task in BioNLP Shared Task 2011. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 7–15.

Liu,H. *et al.* (2011) From Graphs to Events: a subgraph matching approach for information extraction from biomedical text. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 164–172.

Mcclosky,D. *et al.* (2011) Event extraction as dependency parsing for BioNLP 2011. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 41–45.

Miwa,M. *et al.* (2012) Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics*, DOI: 10.1093/bioinformatics/bts237.

Miwa,M. *et al.* (2010) Event extraction with complex event classification using rich features. *J. Bioinform. Comput. Biol.*, **08**, 131.

Móra,G. et al. (2009) Exploring ways beyond the simple supervised learning approach for biological event extraction. In: *Proceedings of BioNLP'09 Shared Task Workshop. Association for Computational Linguistics*, Morristown, NJ, pp. 137–140.

Neves,M.L. et al. (2009) Extraction of biomedical events using case-based reasoning. In: *Proceedings of BioNLP'09 Shared Task Workshop. Association for Computational Linguistics*, Morristown, NJ, pp. 68–76.

Poon,H. and Vanderwende,L. (2010) Joint Inference for Knowledge Extraction from Biomedical Literature. In: *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics,* pp. 813–821.

Quirk,C. *et al.* (2011) MSR-NLP Entry in BioNLP Shared Task 2011. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 155–163.

Riedel,S. (2011) Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 46–50.

Riedel,S. and Andrew,M. (2011) Fast and robust joint models for biomedical event extraction. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing,* pp. 1–12.

Segura-Bedmar,I. *et al.* (2011) A linguistic rule-based approach to extract drug-drug interactions from pharmacological documents. *BMC Bioinformatics*, **12** (Suppl. 2), S1.

Tari,L. *et al.* (2010) Discovering drug-drug interactions: a text-mining and reasoning approach based on properties of drug metabolism. *Bioinformatics (Oxford, England)*, **26**, i547–i553.

Vlachos,A. and Craven,M. (2011) Biomedical event extraction from abstracts and full papers using search-based structured prediction. In: *Proceedings of BioNLP Shared Task 2011 Workshop,* pp. 36–40.

Wren,J.D. (2011) Question answering systems in biology and medicine—the time is now. *Bioinformatics (Oxford, England)*, **27**, 2025–2026.