Relation extraction methods for biomedical literature

Bui, Q.C.

Publication date
2012

Chapter 5. A robust approach to extract biomedical events from literature

Abstract

The abundance of biomedical literature has attracted significant interest in novel methods to automatically extract biomedical relations from texts. 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 such as biomedical events.

In this chapter we propose a novel rule-based 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 on the Genia event extraction test set (Task 1) using '*Approximate Span/Approximate Recursive matching*' criteria, we obtain results with recall, precision and F-score of 42.63, 68.77 and 52.63%, respectively, which is comparable to the state-of-the-art systems. Furthermore, our system achieves superior performance in terms of speed.

---

[*] This chapter based on Bui & Sloot., "A robust approach to extract biomedical events from literature". *Bioinformatics* 2012. DOI: 10.1093/bioinformatics/bts487

5.1 Introduction

Automatic relation extraction is an important and established way of extracting information from biomedical literature [1, 107]. Many relation extraction approaches have been proposed to extract binary relations between biomedical entities from text such as protein-protein interactions [43, 45], drug-drug interactions [123], and causal relations on drug resistance [124]. However, such binary relations cannot fully represent the biological meaning of the original relations. As a consequence, there is a need to have a better representation that can exemplify complex relations extracted from text. Recently, this shortcoming is addressed in the BioNLP'09 Shared Task (BioNLP, hereafter) [121] by introducing the biomedical event extraction task, which aims to extract complex and nested events from text. An event is defined as follows: each event consists of a trigger, a type, and one or more arguments. Depending on the event type (e.g. simple, binding, regulation), an argument can either be a protein or another event (see [121] for more details). Figure 5.1 illustrates the complex and nested structures of the biomedical events. Figure 5.1a shows a *Negative_Regulation* event which has two events as the arguments, whereas Figure 5.1b shows a *Positive_Regulation* event which has one protein and one event as the arguments.

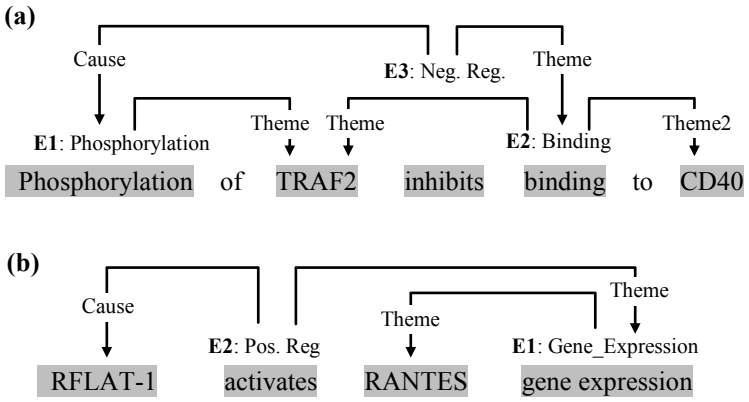

Figure 5.1. (**a**) Examples of three event types: simple (E1), binding (E2), and regulatory (E3) events. (**a**, **b**) Variants of regulatory event arguments: the event E3 in Fig. 1a has two events as the arguments whereas the event E2 in Fig. 1b has one protein and one event as the arguments

Several event extraction approaches have been proposed to the BioNLP'09 and BioNLP'11 challenges [60, 121]. 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 detected 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 used 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 is manually defined or generated from training data. To extract events from text, first each candidate trigger is validated to determine its event type. Second, the defined rules are often applied to the parse tree which contains the trigger to find relation between the trigger and its arguments [125–127]. 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 Kilicoglu and Bergler [127] ranks second on the BioNLP'09 event extraction track (task 1).

ML-based systems deal with the event extraction task as a classification problem. In which, candidate event triggers are classified as true event triggers or not and which arguments attached to them are correct. Extraction methods proposed in the BioNLP'09 exploit various learning algorithms and feature sets to leverage the system performance [128, 129]. Methods proposed after the BioNLP'09 can be divided into two groups based on how event triggers and arguments are determined. In the first group, some systems adopt the workflow and feature sets proposed by Björne et al. [129] and later improved by Miwa et al. [130], in which the evaluation of the candidate triggers and the determination of their arguments are carried out independently. In this type of approach, errors made in the trigger detection step propagate into subsequent steps. Examples of such systems are [102, 103, 131]. 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 [132–134]. Overall, ML-based systems achieve good results on both BioNLP'09 and BioNLP'11.

Apart from these two main approaches, there are a few systems that employ different methods. Móra et al. [135] propose a hybrid method that combines both rule- and ML-based approaches, Liu et al. [73] employ a system using sub-graph matching, Neves et al. [136] use case-based reasoning system, McClosky et al. [137] introduce a system using dependency graphs by extending the function of an existing dependency parser. Cohen et al. [138] propose a system using manually defined patterns while Nguyen et al. [75] employ a method that generates patterns automatically from training data. However, the performance of these systems is not comparable to the state-of-the-art ML-based systems.

In this chapter we propose 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, from that rules are extracted automatically by applying predefined templates. The obtained rules are unified to make them more flexible. Finally, we develop event extraction algorithms to process these unified rules. By employing the structured representation to decompose the nested and complex structures of events into simple syntactic layers, we can use simple methods to learn and process extraction rules. When evaluated on the test set, our system achieves a comparative performance in terms of precision and computational times. To the best of our knowledge, this is the first system of its type used to extract biomedical events from text.

5.2 System and methods

Our core 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 a structured representation. In the following sections, we present these steps in more detail.

5.2.1 Structured representation

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

Chunk: is the base syntactic layer that can express an event (see Figure 5.2). 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: adjective, noun, verb, and preposition.

Phrase: consists of a list of chunks, connecting by preposition chunks. A phrase can express one or more events in which the event triggers and 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], [O and] [NP PRO3] are merged into one noun chunk [NP PRO1, PRO2, and PRO3].

Clause: consists of a verb chunk connecting with a left child and a right child. A child is a phrase but it can also be a chunk. A clause can express one or more events in which the event trigger belongs to the verb chunk and its arguments belong to the clause's children.

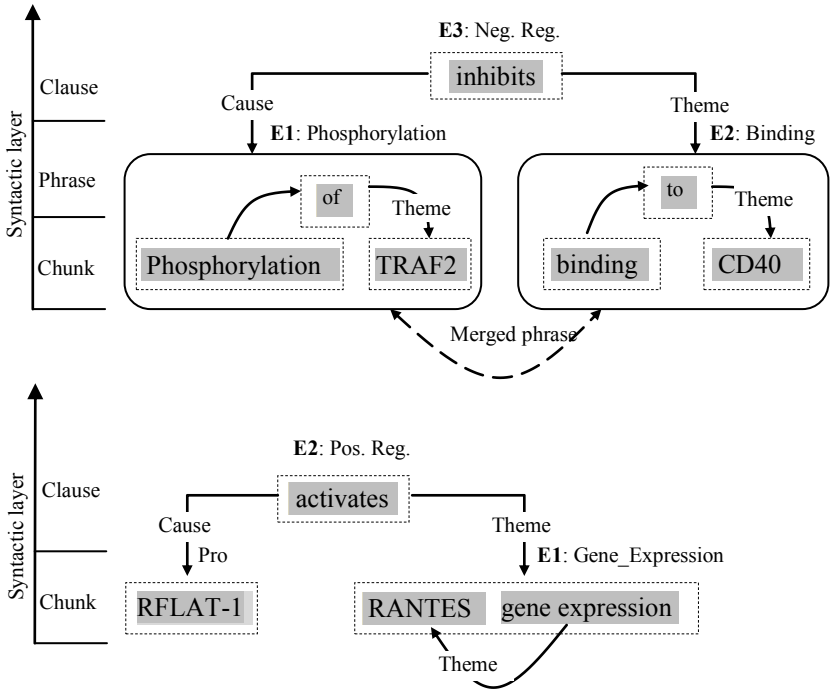

Figure 5.2. Structured representations for the events in Figure 5.1

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 when its trigger and arguments belong to the left child and the right child of a clause. For example, the binding event E2 in Figure 5.1 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, 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 training data. Of which, both event triggers and their arguments must belong to the same sentence. Figure 5.2 presents the events in Figure 5.1 using structured representations.

5.2.2 Learning rules

To learn extraction rules from the structured representations, we define a list of syntactic and semantic features to capture the underlying rules which govern the relations between event triggers and their arguments when expressed by structured representations. These features are divided into three groups which correspond to

three syntactic layers: chunk, phrase, and clause. The descriptions of these features are as follows.

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 that event trigger used in the chunk form.

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 that event trigger used in the phrase form.

Preposition: indicates which preposition is used to connect the chunk which contains the 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 which contains the 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 which contains the *theme* to the chunk containing the second arguments (i.e. *theme2* for binding event or *cause* for regulatory event), e.g. [NP *binding*] **of** [NP *PRO1*] **to** [NP *PRO2*]; [NP *effect*] **of** [NP *PRO1*] **on** [NP *PRO2*].

Distance2: is used for binding and regulatory events. It measures the distance (by chunk) from the chunk which contains the event trigger to the chunk containing 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 the event: *effect* of *PRO1* on *PRO2* in which PRO1 is the *cause*. This feature helps determining whether 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 the event trigger. For example: PRO1 *binding* **to** PRO2, *interaction* **between** PRO1 **and** PRO2.

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

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 the event triggers 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.

Specific feature for binding events:

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

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.

We create three templates, where each template consists of the features designated for a corresponding layer (i.e. chunk, phrase, and clause), to learn extraction rules by mapping each annotated event from the training data into a suitable layer. The learning procedure is shown in Algorithm 5.1.

Algorithm 5.1. Learning extraction rules from annotated events

Input: Sentence **S,** list of events **E** belongs to **S**, a dictionary **D**, map **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
```

The text conversion step used in this algorithm is described in section 5.2.5. During this learning process, some events are skipped if their event triggers do not belong to a dictionary which is prepared based on single-word event triggers

collected from the training data. In addition, events are skipped if they cannot be mapped into the structured representation.

5.2.3 Rule combination

For each event trigger and its specific POS tag, we may obtain from one to three sets of extraction rules which correspond to three layers: chunk, phrase, and clause. The number of rule sets for each event trigger depends on its POS tags. For example, the *binding* trigger has three POS tags: ADJ, NN, and VBG therefore it may possess up to nine sets of extraction rules. Examples of a set of extraction rules of the *binding* trigger and its POS tag NN which learned from the phrase layer are shown in Table 5.1. The first row of this table is an extraction rule that learned from binding events such as the binding event E2 in Figure 5.2.

Table 5.1. Examples of a set of extraction rules of the *binding* trigger obtained from phrase layer. Some features are omitted for clarity

POS	…	Prep.	Prep.2	Dist.	Dist.2	Freq.
NN		to		4	0	3
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
NN		after	of	7	2	1

To simplify the event extraction step and to make the extraction rules generalize well, we combine the extraction rules of the same rule set which obtained from the phrase layer or the 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. Example of such a decision table, which results from the combination of the set of extraction rules in Table 5.1, for the *binding* trigger and its POS tag NN is shown in Table 5.2. In addition, for each decision table of an event trigger, we calculate a confident 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.

Table 5.2. A decision table of the *binding* trigger and its POS tag NN used for phrase layer

Feature	Value	Feature	Value
Rule type	*Phrase*	Distance 1	*6*
POS	*NN*	Distance 2	*10*
Trigger	*Binding*	Search order	*right; left; both*
Theme2 score	*0.35*	Preposition	*of; to; in; by*
Confident score	*0.28*	Preposition2	*of:[to, in];*
Theme type	*Protein*	Theme2 type	*Protein*

Since simple, binding and regulatory events have different argument types and there are two types of extraction rules need to be combined (i.e. extraction rules extracted from phrase and clause layers). Therefore, we need 6 variants of the decision tables to represent these unified rules. However, the procedures used to combine these rules are similar in which we use a simple voting scheme based on frequency to select the rules that are more likely reliable and remove those contradict to the selected one. An example of such rule combination algorithms is show in Algorithm 5.2.

Algorithm 5.2. 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** $e \in$ **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 $\mathbf{G_i}$

5.2.4 Event extraction

In this section we present an algorithm used to extract events from an input sentence. First, the sentence is preprocessed to detect candidate event triggers and to convert into structured representations. Candidate event triggers are detected using the same dictionary as 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 candidate trigger, its POS, the layer containing it, and its event type. For example, if the *binding* trigger belongs to a phrase 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 might 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 5.2, the decision can easily be made based on the syntactic layer. In this case, pairing *RANTES* with the *expression* trigger should have higher priority than pairing with the *activates* trigger. Based on the structured representation, we define an event extraction procedure which 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 5.3.

Algorithm 5.3. 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 ∈ 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 ≠ 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 ∈ 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 ≠ null and L1 ≠ null
8         If hasCause(g, r)
9           L2= queryArgs(l,g,r) // cause/ theme2
10      formEvent(g, L1, L2)
```

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)
```

5.2.5 Text preprocessing

The text preprocessing step consist of splitting sentences, replacing protein names with place-holders, tokenizing words into tokens, 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.

Converting chunks into structured representation

We adapt the method described in Segura-Bedmar et al. [62] to convert chunks 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 5.2.1. Furthermore, if 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.

5.3 Results and discussion

5.3.1 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. The datasets consist of training, development, and test datasets. For training and development datasets, a list of proteins/genes and annotated events are given. For the test set, only a list of proteins/genes is given. Statistics of the datasets are shown in Table 5.3.

Table 5.3. Statistics of training, development, and test datasets. Values in parentheses denote the number of full papers

Items	Training	Development	Test
Abstracts + full papers	800 (+5)	150 (+5)	260 (+4)
Sentences	8759	2954	3437
Proteins	11625	4690	5301
Events	10310	3250	4457
Availability of events	Yes	Yes	No

5.3.2 Evaluation settings

We use both training and development datasets for building the dictionary and 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 the 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.

5.3.3 Event extraction

Table 5.4 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 easy 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. Overall, the results on the full text dataset are better than on the abstract data set. The results on simple and binding events in the full text 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.

Table 5.4. Evaluation results of the test dataset. R, P, and F denote recall, precision and F-score, respectively

Event Class	Abstract			Full text		
	R	P	F	R	P	F
Gene_Expression	63.43	85.29	72.76	76.43	88.43	81.99
Transcription	58.39	81.63	68.09	40.54	100.00	57.69
Pro_catabolism	42.86	60.00	50.00	100.00	100.00	100.00
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**

Table 5.5 and Table 5.6 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 detail we also present the evaluation results on each dataset separately, where Table 5.5 show the results on the abstract dataset and Table 5.6 show the results on the full text dataset.

Table 5.5 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 than that system on simple events. However, the performance on regulatory (REG) events is lower than best system, with a gap of 12 F-score points. Overall, on the abstract dataset, the Riedel and Andrew [133] 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 [102] system achieves good results in terms of recall. Furthermore, the abstract dataset is the same dataset as used in the BioNLP'09, indicating that the performance of the current extraction methods has been improved, with the best system 4 F-score points higher than the previous state-of-the-art system (56.05% vs. 51.95%) [129].

Table 5.5. Performance comparison on the abstract dataset

System	SVT	BIND	REG	TOTAL		
	F	F	F	R	P	F
Riedel [101]	71.54	**50.76**	**45.51**	48.74	65.94	**56.05**
Björne [102]	70.36	47.50	44.30	**50.06**	59.48	54.37
Quirk [103]	70.08	43.86	40.85	48.52	56.47	52.20
Kilicoglu [64]	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

Table 5.6. Performance comparison on the full text dataset

System	SVT	BIND	REG	TOTAL		
	F	F	F	R	P	F
Riedel [101]	79.18	44.44	**40.12**	47.84	59.76	53.14
Björne [102]	76.98	34.39	39.16	**48.31**	53.38	50.72
Kilicoglu [64]	78.39	35.56	38.12	44.71	57.75	50.40
Quirk [103]	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**

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 [139]. The results in Table 5.6 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 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. A closer look at 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% vs. 49.38% for Kilicoglu's system; 64.93% vs. 65.34% for our system), three ML-based systems drop precision significantly (e.g. 60.89% vs. 47.62 for Riedel's system; 50% vs. 31.76% for Björne's system; 44.51%

vs. 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 [64]. 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, on full text dataset our system outperforms the best system of the BioNLP'11 in terms of both precision and F-score.

5.3.4 Performance time

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 [140], 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 & McCallum [133] report that their system needs from 60ms to 297ms, depending on the learning models, to extract events from a sentence. However, this is not included the parsing times and features extraction times. Björne et al. [118] report in their large-scale experiment that, on average, their system needs 954ms to parse a sentence and 486ms to extract events from that sentence. Since Riedel & McCallum [133] use the same parser as Björne et al. [118], we assume that the parsing times of the two systems are equal. Therefore, both systems need from 1040ms to 1400ms to extract events from a sentence. In contrast, our system needs 6.4ms to do so. Details of the performance times are shown in Table 5.7.

Table 5.7. Performance times of our system on the test set and training set. The experiments are run on a PC with Core™i5 2.3MHz CPU, 4 GB of memory

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

In general, it is not straightforward to directly compare the computational times between systems due to the differences in 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 5.7 where the text processing times and event extraction times vary on two datasets even though they are run on the same system. However, it is

apparent that our system outperforms the mentioned systems, being a 150-fold faster in terms of computational times.

5.3.5 Performance analysis

The results in the previous sections show that our event extraction system outperforms the existing rule-based systems in terms of both precision and recall. In general, its recall is lower than the recall of ML-based systems but the precision of our system is higher than these systems. Furthermore, our system generalizes well on full text dataset and is computationally effective. These characteristics are very important since full text documents are potentially provide richer source of information for events to be extracted but also require more computational resources.

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

First, most event extraction systems use a dictionary to detect candidate event triggers however the construction and the use of such a dictionary vary between systems [121]. We also use a dictionary to detect candidate event triggers from input text. Each entry in our dictionary is assigned a confident score as proposed by Kilicoglu and Bergler [127]. In our experiments, we set the threshold for confident score of the dictionary entries to 0.1. We found that raising the threshold would increase precision for some event classes but this would also decrease recall of the others. 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 fail to return a desired proteins/events list. In addition, a recent comprehensive study carried out by Tikk et al. [33] on extraction methods for protein-protein interactions reveals that systems are tuned using specific-corpus parameters drop F-score considerably when evaluated on unknown characteristic datasets. Therefore we decided not to tune the dictionary threshold for performance.

Table 5.8 shows recall of the dictionary (see column 3) evaluated on the training and development datasets. The results indicate that the recall varies significantly among event classes where binding and regulatory events have lower recall than the simple events. This is one of the reasons why the performance of the binding and regulatory events is lower than the simple events. Furthermore, the effect of the dictionary on the learning phase can be observed clearly in Table 5.8. In which, the percentage of events learned from each event class is affected by the recall of the dictionary for that event class. This value can be considered as the performance upper bound (PUB) of that event class and it is observed that the higher PUB of an event class, the better performance that class may achieve. For instance, among three event classes: Gene_Expr, Binding, and Pos.Reg, the Gene_Expr class has the

highest PUB therefore it achieves good F-score (89 vs. 72) whereas the Binding class has lower PUB and also achieves lower F-score (77 vs. 49) and the Positive_Regulation class achieves the lowest F-score due to its lowest PUB (70 vs. 41.29).

Table 5.8. Statistics of the learning phase on the training and development datasets

Event type	# events	Recall of dict.	% event learned	# unified rules
Gene_Expression	3014	91	89	70
Transcription	825	72	72	20
Pro_catabolism	133	87	87	6
Phosphorylation	303	93	89	8
Localization	348	81	79	29
Binding	1363	79	77	72
Regulation	1409	77	68	98
Pos_Regulation	4385	78	70	220
Neg_Regulation	1780	76	65	143

The second factor that affects the system performance is the rule combination step. While combining extraction rules definitely simplifies the event extraction method, it also causes the loss of information. As shown in the Algorithm 5.3 (line 8), during the combination process, we remove some rules that contradict to the selected rules. This leads to lower 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 might have a *cause* as a protein or as an event. However, these data do not tell 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. Positive_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.

Finally, the variants of event triggers in each event class also affect the system performance. Table 5.8 shows that the variants in regulatory classes, which indicated by the number of unified rules, are much higher than in the simple classes. In our experiments, we set the threshold for all unified rules to 0.03. We found that setting threshold for each event class or even individual trigger could increase recall for

some cases. However, as we mentioned, this is specific-corpus parameter tuning, we decided not doing so.

5.4 Conclusion

In this chapter 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 simplifying the learning and extraction phases but also 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 speed. It is clearly suited for large-scale experiments.

Our event extraction method is simpler than the existing ML-based approaches. It is also more robust than the previous proposed rule-based approaches since it learns rules automatically from training data. Its simplicity and robustness has been proven by the performance on simple and binding events. Our approach still has modest performance on regulatory events however this issue has been addressed and therefore opens opportunities for improvements. Its structured representation is generic and is capable to represent any relation types. The proposed feature sets based on the structured representation consist of mainly 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.