<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>A robust approach to extract biomedical events from literature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Bui, Quoc-Chinh; Sloot, Peter M. A.</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Bui, Q. C. &amp; Sloot, P. M. A. (2012). A robust approach to extract biomedical events from literature. Bioinformatics, 28(20), 2654-2661.</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2012</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/10109">http://hdl.handle.net/10220/10109</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>© 2012 The Author(s). Published by Oxford University Press. This paper was published in Bioinformatics and is made available as an electronic reprint (preprint) with permission of The Author(s). The paper can be found at the following official DOI: <a href="http://dx.doi.org/10.1093/bioinformatics/bts487">http://dx.doi.org/10.1093/bioinformatics/bts487</a>. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper is prohibited and is subject to penalties under law.</td>
</tr>
</tbody>
</table>
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

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 Cohenet 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 to leverage the system performance (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 Vlahos and Craven (2011). Overall, ML-based systems achieve
Extracting biomedical events from literature

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 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 on the left or on the right of an event trigger. For examples, PRO1 binding to PRO2, interaction between PRO1 and PRO2.

Theme2 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 trigger 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.

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
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. These specific feature values such as \textit{Theme2Count} and \textit{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.

### 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 \textit{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-1 activates RANTES gene expression, with the flat representation, it is not clear whether \textit{RANTES} should pair with the 'activates' trigger or with the 'expression' trigger. However, when mapped into structured representation as shown in Figure 2, the decision can easily be made based on the syntactic layer. In this case, pairing \textit{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/](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. \textit{PRO}\_i (i is the index of the \textit{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/](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 [\textit{VB is}] \textit{[ADJ dependent]} are merged into [\textit{VB is dependent}]; the merged adjective chunk into the verb chunk. For example, the following chunks [\textit{VB is}] \textit{[ADJ dependent]} are merged into [\textit{VB is dependent}]; the merged adjective chunk into the verb chunk. For example, the following chunks [\textit{VB is}] \textit{[ADJ dependent]} are merged into [\textit{VB is dependent}]; the merged adjective chunk into the 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](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
queryArg(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/Theme two
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 = queryArg(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 phrase l ∈ T containing g
(7) L1 = queryArg(l, g, r) / / theme
(8) If r ≠ null and L1 = null
(9) g2 = getNext(G) / / get next trigger
(10) If inRange(g2, g)
(11) L1 = PhraseExtraction(g2, G, T);
(12) If L1 ≠ null
(13) If hasCause(g, null)
(14) L2 = queryArg(l, g, r) / / cause / theme 2
(15) formEvent(g, l, 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 = queryArg(l, g, r)
(7) If r ≠ null and L1 ≠ null
(8) If hasCause(g, null)
(9) L2 = queryArg(l, g, r) / / cause / theme 2
(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

Main function

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

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

<table>
<thead>
<tr>
<th>Items</th>
<th>Training</th>
<th>Development</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstracts + full papers</td>
<td>800 (+5)</td>
<td>150 (+5)</td>
<td>260 (+4)</td>
</tr>
<tr>
<td>Sentences</td>
<td>8759</td>
<td>2954</td>
<td>3437</td>
</tr>
<tr>
<td>Proteins</td>
<td>11625</td>
<td>4690</td>
<td>5301</td>
</tr>
<tr>
<td>Events</td>
<td>10310</td>
<td>3250</td>
<td>4457</td>
</tr>
<tr>
<td>Availability of events</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
| Values in parentheses denote the number of full papers.

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.

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

<table>
<thead>
<tr>
<th>Event class</th>
<th>Abstract</th>
<th>Full text</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>P</td>
<td>F</td>
</tr>
<tr>
<td>Gene_Expr.</td>
<td>63.43</td>
<td>85.29</td>
</tr>
<tr>
<td>Transcription</td>
<td>58.39</td>
<td>81.63</td>
</tr>
<tr>
<td>Pro_catabolism</td>
<td>42.86</td>
<td>60.00</td>
</tr>
<tr>
<td>Phosphorylation</td>
<td>71.11</td>
<td>96.00</td>
</tr>
<tr>
<td>Localization</td>
<td>47.70</td>
<td>97.65</td>
</tr>
<tr>
<td>SVT-TOTAL</td>
<td>61.17</td>
<td>87.11</td>
</tr>
<tr>
<td>Binding</td>
<td>39.48</td>
<td>64.93</td>
</tr>
<tr>
<td>EVT-TOTAL</td>
<td>56.25</td>
<td>82.61</td>
</tr>
<tr>
<td>Regulation</td>
<td>20.62</td>
<td>52.17</td>
</tr>
<tr>
<td>Pos_regulation</td>
<td>32.55</td>
<td>56.44</td>
</tr>
<tr>
<td>Neg_regulation</td>
<td>24.54</td>
<td>49.21</td>
</tr>
<tr>
<td>REG-TOTAL</td>
<td>28.61</td>
<td>54.31</td>
</tr>
<tr>
<td>ALL-TOTAL</td>
<td>41.89</td>
<td>69.72</td>
</tr>
</tbody>
</table>

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

Table 5. Performance comparison on the abstract test dataset

<table>
<thead>
<tr>
<th>System</th>
<th>SVT</th>
<th>BIND</th>
<th>REG</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>P</td>
<td>F</td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>Riedel (2011)</td>
<td>71.54</td>
<td>50.76</td>
<td>45.51</td>
<td>48.74</td>
</tr>
<tr>
<td>Björne (2011)</td>
<td>70.36</td>
<td>47.50</td>
<td>44.30</td>
<td>50.06</td>
</tr>
<tr>
<td>Quirk (2011)</td>
<td>70.08</td>
<td>43.86</td>
<td>40.85</td>
<td>48.52</td>
</tr>
<tr>
<td>Kilicoglu (2011)</td>
<td>67.75</td>
<td>37.41</td>
<td>40.96</td>
<td>43.09</td>
</tr>
<tr>
<td>Ours</td>
<td>71.87</td>
<td>49.10</td>
<td>37.48</td>
<td>41.89</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>System</th>
<th>SVT</th>
<th>BIND</th>
<th>REG</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>P</td>
<td>F</td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>Riedel (2011)</td>
<td>79.18</td>
<td>44.44</td>
<td>40.12</td>
<td>47.84</td>
</tr>
<tr>
<td>Björne (2011)</td>
<td>76.98</td>
<td>34.39</td>
<td>39.16</td>
<td>48.31</td>
</tr>
<tr>
<td>Kilicoglu (2011)</td>
<td>78.39</td>
<td>35.56</td>
<td>38.12</td>
<td>44.71</td>
</tr>
<tr>
<td>Quirk (2011)</td>
<td>75.63</td>
<td>36.14</td>
<td>38.21</td>
<td>48.94</td>
</tr>
<tr>
<td>Ours</td>
<td>80.39</td>
<td>60.45</td>
<td>34.99</td>
<td>44.47</td>
</tr>
</tbody>
</table>

The results in Table 5 show 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. 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. 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...
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


