A Search-based Neural Model for Biomedical Nested and Overlapping Event Detection

Kurt Espinosa¹⁴, Makoto Miwa²³, Sophia Ananiadou¹

¹National Centre for Text Mining, School of Computer Science, The University of Manchester, UK
²Toyota Technological Institute, Nagoya, 468-8511, Japan
³Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan
⁴Department of Computer Science, University of the Philippines Cebu, Cebu City, Philippines

{kurt.junshean.espinosa, sophia.ananiadou}@manchester.ac.uk
makoto-miya@toyota-ti.ac.jp

Abstract

We tackle the nested and overlapping event detection task and propose a novel search-based neural network (SBNN) structured prediction model that treats the task as a search problem on a relation graph of trigger-argument structures. Unlike existing structured prediction tasks such as dependency parsing, the task targets to detect DAG structures, which constitute events, from the relation graph. We define actions to construct events and use all the beams in a beam search to detect all event structures that may be overlapping and nested. The search process constructs events in a bottom-up manner while modelling the global properties for nested and overlapping structures simultaneously using neural networks. We show that the model achieves performance comparable to the state-of-the-art model Turku Event Extraction System (TEES) on the BioNLP Cancer Genetics (CG) Shared Task 2013 without the use of any syntactic and hand-engineered features. Further analyses on the development set show that our model is more computationally efficient while yielding higher F1-score performance.

1 Introduction

Nested and overlapping event structures, which occur widely in text, are important because they can capture relations between events such as causality, e.g., a “production” event is a consequence of a “discovery” event, which in turn is a result of an “exploration” event. Event extraction involves the identification of a trigger and a set of its arguments in a given text. Figure 1 shows an example of a nested and overlapping event structure in the biomedical domain. The relation graph (topmost) forms a directed acyclic graph (DAG) structure (McClosky et al., 2011) and it encapsulates 15 event structures. It contains nested event structures such as $E_2,E_3$ because one of their arguments, in this case $E_1$, is an event. Specifically, $E_1$ is a flat event since its argument is an entity. Moreover, $E_2$ and $E_3$ are also overlapping events (explicitly shown in the relation graph having two induction triggers) because they share a common argument, $E_1$.

State-of-the-art approaches to event extraction in the biomedical domain are pipeline systems (Björne and Salakoski, 2018; Miwa et al., 2013) that decompose event extraction into simpler tasks such as: i) trigger/entity detection, which determines which words and phrases in a sentence potentially constitute as participants of an event, ii) relation detection, which finds pairwise relations between triggers and arguments, and iii) event detection, which combines pairwise...
relations into complete event structures. Joint approaches have also been explored (Rao et al., 2017; Riedel and McCallum, 2011; Vlachos and Craven, 2012; Venugopal et al., 2014), but they focus on finding relation graphs and detect events with rules. McClosky et al. (2011) treats events as dependency structures by constraining event structures to map to trees, thus their method cannot represent overlapping event structures. Other neural models in event extraction are in the general domain (Feng et al., 2016; Nguyen and Grishman, 2015; Chen et al., 2015; Nguyen et al., 2016), but they used the ACE2005 corpus which does not have nested events (Miwa et al., 2014). Furthermore, there are some efforts on applying transition-based methods on DAG structures in dependency parsing, e.g., (Sagae and Tsujii, 2008; Wang et al., 2018), however, they do not consider overlapping and nested structures.

We present a novel search-based neural event detection model that detects overlapping and nested events with beam search by formulating it as a structured prediction task for DAG structures. Given a relation graph of trigger-argument relations, our model detects nested events by searching a sequence of actions that construct event structures incrementally in a bottom-up manner. We focus on event detection since the existing methods do not consider nested and overlapping structures as a whole during learning. Treating them simultaneously helps the model avoid inferring wrong causality relations between entities. Our model detects overlapping events by maintaining multiple beams and detecting events from all the beams, in contrast to existing transition-based methods (Nivre, 2003, 2006; Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Vlachos and Craven, 2012). We define an LSTM-based neural network model that can represent nested event structure to choose actions.

We show that our event detection model achieves performance comparable to the event detection module of the state-of-the-art TEES system (Björne and Salakoski, 2018) on the BioNLP CG Shared Task 2013 (Pyysalo et al., 2013) without the use of any syntactic and hand-engineered features. Furthermore, analyses on the development set show our model performs fewer number of classifications in less time.

2 Model

We describe our search-based neural network (SBNN) model that constitutes events from a relation graph by structured prediction. SBNN resembles an incremental transition-based parser (Nivre, 2006), but the search order, actions and representations are defined for DAG structures. We first discuss how we generate the relation graph in §2.1, then describe the structured prediction algorithm in §2.2 and the neural network in §2.3 and lastly, we explain the training procedure in §2.4.

2.1 Relation Graph Generation

To train our model, we use the predicted relations merged with pairwise relations decomposed from the gold events. We then generate a relation graph from the merged relations. During inference, the relation graph is generated only from the predicted relations. Figure 1 shows an example of the generated DAG-structured relation graph.

2.2 Structured Prediction for DAGs

We represent event structures with DAG structures and find them in a relation graph. Our model performs beam search on relation graphs by choosing actions to construct events. Unlike existing beam search usage where they only choose the best path in the beam, e.g., (Nivre, 2006), we use all the beams to predict event structures, which enables the model to predict overlapping and nested events.

Event structures are searched and fixed for each
trigger of the relation graph in a bottom-up manner. The model predicts the flat events first and then the representations of the flat events become the argument of the nested events. Figure 2 shows the proposed neural model (described in detail in §2.3), which illustrates that the event representation of event E1 becomes the argument to event E2. If the flat event is not detected, its nested event will not be detected consequently. When this happens, the search process stops.

Figure 3 shows an snapshot of the search procedure within one time step as applied to a relation graph with a trigger induction to detect event E2 (see Figure 1 and 2). To do the search, the model maintains two data structures: a buffer B which holds a queue of arguments to be processed and a structure S which contains the partially built event structure. The initial state is composed of the buffer B with all the arguments and a structure S empty. At each succeeding time step, the model applies a set of predefined actions to each argument and uses a neural network to score those actions. Processing completes when B is empty and S contains all the arguments for the trigger along with the history of actions taken by the model.

We have chosen only three actions for simplicity. Figure 3 shows how actions are applied to the argument angiogenesis (due to space constraints we only show two actions indicated by the arrows). Concretely, we show two distinct structures (S_1, S_2) that are created in time step t + 1 after CONSTRUCT and IGNORE actions were applied to the current structure S_1 in time step t.

The event candidate structures are fixed as events if the scores of the CONSTRUCT actions are above a certain threshold. The resulting state after a CONSTRUCT action is removed from the beams. We maintain multiple beams and use all of them to predict overlapping events.

2.3 Neural Network as Scoring Function

Figure 2 shows the proposed neural model. We employ a BiLSTM network to generate the word representations from pre-trained word embeddings. To represent phrases, we averaged the word representations. The LSTM network is shared among the states during search in the sentence. We build a relation embedding for each argument, which concatenates the information of the trigger t, the role o, the argument a and the action c. We include both the type information p and the word or phrase representation w of the trigger or entity argument. Formally, each relation r_i is represented as a relation embedding: r_i = [t_p; t_w; o_p; a_p; a_w; c], where t_p is the representation of the type of the trigger and so on. Each r_i is passed to a linear hidden layer, then to a rectified linear unit (ReLU) non-linearity and summed to produce the structure and buffer embeddings: S_t and B_t.

We use a neural network as the action scoring function (indicated by the dotted box in Figure 2) defined as σ(a_t|S_{t−1}, B_{t−1}). We model the action scoring function using S_t and B_t, which are composed by adding an action a_t for a relation r_i to S_{t−1} and moving r_i from B_{t−1} to S_{t−1}. The state at any time step t is composed of the buffer B_t and the partially built structure S_t. Each of S_t and B_t contains a set of relations \{r_1, r_2, r_3, \ldots, r_n\}. In Figure 2, there is no arrow to B in event E1 because the diagram only shows the model snapshot at a specific time step during the search process. In this particular time step t, the buffer B in E1 is already empty and thus, it does not contain any relations r_i.

S_t and B_t are then concatenated to form the
**event embedding.** The event embedding has the same dimension as the sum of argument type and word dimensions so that it can be used as argument representation in nested events as shown in Figure 2. Then, we passed the event embedding into a linear hidden layer and output $z_t$. Finally, the scoring function $\sigma$ is calculated as $\sigma(a_t|S_{t-1}, B_{t-1}) = \text{sigmoid}(z_t)$.

**2.4 Training**

From the relation graph generated in §2.1, we calculate gold action sequences that construct the gold event structures on the graph. The loss is summed over all actions and for all the events during the beam search and thus the objective function is to minimise their negative log-likelihood. We employ early updates (Collins and Roark, 2004): if the gold falls out of the beam, we stop searching and update the model immediately.

**3 Experimental Settings**

We applied our model to the BioNLP CG shared task 2013 (Pyysalo et al., 2015). We used the original data partition and employed the official evaluation metrics. We focussed on the CG task dataset over other BioNLP datasets because of its complexity and size (Nédellec et al., 2013; Björne and Salakoski, 2018; Pyysalo et al., 2013). The CG dataset has the most number of entity types and event types and thus is the most complex among the available (and accessible) BioNLP datasets. Furthermore, the CG dataset is the largest dataset in terms of the number of event instances and the proportion of nested and overlapping events. Evaluating our model extensively on other BioNLP tasks and datasets is part of our future work.

The development set contains 3,217 events of which 36.46% are nested events, 43.05% are overlapping events and 44.07% are flat events. Note that the total does not equal to 100% because nested and overlapping events may have intersection: a nested event can be overlapping and vice versa.

We compared our model with the event detection module of the state-of-the-art model TEES (Björne and Salakoski, 2018), which employs convolutional neural network and uses syntactic and hand-engineered features for event detection. Björne and Salakoski (2018) found that the dependency parse features increased the performance of the convolutional model. In contrast, we do not use these syntactic features nor hand-engineered features. Furthermore, instead of the ensemble methods, we used TEES’s published single models, as this enables us to make a direct comparison with TEES in a minimal setting. We train our model using the predicted relations from TEES merged with the pairwise relations decomposed from the gold CG events. During inference, we predict event structures using only the predicted relations from TEES.

**3.1 Nested and Overlapping Event Evaluation Process**

Similarly, we used the official evaluation script to measure the performance of the model on nested, overlapping and flat events. We first separated the nested, overlapping and flat events, respectively. Then we compute the precision and recall for each category in the following way. For example, for nested events, to compute the precision, we compare the predicted nested events with all gold events and to compute recall, we compare gold nested events with all predicted events. The evaluation script detects nested events by comparing the whole tree structure down to its sub-events until it reaches the flat events. Hence, the performance scores of the nested events inevitably include the performance on flat events.

**3.2 Training Details and Model Parameters**

We implemented our model using the Chainer library (Tokui et al., 2015). We initialised the word embeddings using pre-trained embeddings (Chiu et al., 2016) while other embeddings are initialised using the normal distribution. All the embeddings and weight parameters were updated with mini-batch using the AMSGrad optimiser (Reddi et al., 2018). We also incorporated early stopping to choose the number of training epochs and tuned hyper-parameters (dropout, learning rate and weight decay rate) using grid search. The model parameters can be found in appendix A.

**4 Results and Analyses**

Table 1 shows the event detection performance of the models on the test set. Our model achieves performance comparable to the state-of-the-art TEES event detection module without the use of any syntactic and hand-engineered features, suggesting it can be applied to other domains with no need for feature engineering. We validated it to have no sig-
Table 1: Event detection performance on the CG task 2013 test dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEES</td>
<td>61.42</td>
<td>52.93</td>
<td>56.86</td>
</tr>
<tr>
<td>SBNN</td>
<td>63.67</td>
<td>51.43</td>
<td>56.90</td>
</tr>
</tbody>
</table>

Table 2: Event detection performance on CG task 2013 development set for SBNN with the top three performing $k$-values and when $k = 1$.

<table>
<thead>
<tr>
<th>Model</th>
<th>P</th>
<th>R</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEES</td>
<td>56.81</td>
<td>48.21</td>
<td>52.16</td>
</tr>
<tr>
<td>SBNN</td>
<td>48.95</td>
<td>43.79</td>
<td>46.23</td>
</tr>
<tr>
<td>$k = 8$</td>
<td>63.60</td>
<td>47.46</td>
<td>54.36</td>
</tr>
<tr>
<td>$k = 64$</td>
<td>60.30</td>
<td>49.16</td>
<td>54.17</td>
</tr>
<tr>
<td>$k = 256$</td>
<td>60.91</td>
<td>48.53</td>
<td>54.02</td>
</tr>
</tbody>
</table>

To gain a deeper insight into the model, we performed analyses on the development set. Table 2 shows the performance of SBNN by varying the $k$-best parameter in beam search. We tested $2^i$ values for $i = 1, 2, 3, \ldots, 11$ and found that the best value was 8 with F1-score of 54.36%, which is 2.2 percentage points (pp) higher than TEES.

Table 3 shows the number of classifications (or action scoring function calls in our model) performed by each model with the corresponding actual running time. SBNN performs fewer classifications and in less time than TEES, implying it is more computationally efficient.

Table 4 shows the performance comparison of the models on nested, overlapping and flat event detection. Our model yields higher F1-scores than TEES which can be attributed to its ability to maintain multiple beams and to detect events from all these beams during search.

Finally, we computed the upper bound recall given the predicted relations from TEES. The upper bound is computed by setting the threshold parameter of our model to zero, which then constructs all gold events possible from the predicted relations of TEES. Since we evaluate our model on the output relations of TEES, the event detection performance is bounded or limited by these predicted relations. For instance, if one of the relations in an event was not predicted, the event structure will never be formed. We observe that this remains a challenging task since the upper bound recall is still at 53.47% (6.01pp higher than our current model’s score). Closing this gap requires among others addressing inter-sentence and self-referential events, which account for 3.1% of the total events.

5 Conclusions and Future Work

We presented a novel search-based neural model for nested and overlapping event detection by treating the task as structured prediction for DAGs. Our model achieves performance comparable to the state-of-the-art TEES event detection model without the use of any syntactic and hand-engineered features, suggesting the domain-independence of the model. Further analyses on the development set revealed some desirable characteristics of the model such as its computational efficiency while yielding higher F1-score performance. These results set the first focussed benchmark of our model and next steps include applying it to other event datasets in the biomedical and general domain. In addition, it can also be applied to other DAG structures such as nested/discontiguous entities (Muis and Lu, 2016; Ju et al., 2018).

Acknowledgement

We thank our anonymous reviewers for their invaluable feedback. This research was supported by funding from BBSRC Japan Partnering Award, Text mining and bioinformatics platforms for metabolic pathway modelling [Grant ID: BB/P025684/1] and AIRC/AIST. The first author gratefully acknowledges financial support from the University of the Philippines System Doctoral Studies Fund.
References


## A Model Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value/Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini-Batch Size</td>
<td>100</td>
</tr>
<tr>
<td>Word Embedding</td>
<td>200</td>
</tr>
<tr>
<td>BiLSTM Word Embedding</td>
<td>100</td>
</tr>
<tr>
<td>Role Type Embedding</td>
<td>10</td>
</tr>
<tr>
<td>Trigger/Argument Type Embedding</td>
<td>20</td>
</tr>
<tr>
<td>Early Stopping Patience</td>
<td>5</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0.001</td>
</tr>
<tr>
<td>Hidden Layer Size</td>
<td>60</td>
</tr>
<tr>
<td>Event Embedding</td>
<td>100</td>
</tr>
<tr>
<td>Action Score Threshold</td>
<td>0.5</td>
</tr>
<tr>
<td>Beam Size</td>
<td>8</td>
</tr>
<tr>
<td>Action Embedding</td>
<td>4</td>
</tr>
<tr>
<td>Weight Decay Rate</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table 5: Hyper-parameters used in our experiments.