Exploring a Probabilistic Earley Parser for Event Composition in Biomedical Texts

Mai-Vu Tran[1] Hoang-Quynh Le[1] Van-Thuy Phi[1] Thanh-Binh Pham[1] Nigel Collier[2,3]
[1]University of Engineering and Technology, VNU, Hanoi, Vietnam
[2]National Institute of Informatics, Tokyo, Japan
[3]European Bioinformatics Institute, Cambridge, UK
{vutm,lhquynh,thuypv,binhpt}@vnu.edu.vn, collier@ebi.ac.uk

Abstract

We describe a high precision system for extracting events of biomedical significance that was developed during the BioNLP shared task 2013 and tested on the Cancer Genetics data set. The system achieved an F-score on the development data of 73.67 but was ranked 5th out of six with an F-score of 29.94 on the test data. However, precision was the second highest ranked on the task at 62.73. Analysis suggests the need to continue to improve our system for complex events particularly taking into account cross-domain differences in argument distributions.

1 Introduction

In this paper we present our approach to the BioNLP 2013 shared task on Cancer Genetics (CG) (Pyysalo *et al.*, 2013, Pyysalo *et al.*, 2012), aimed at identifying biomedical relations of significance in the development and progress of cancer. Our system explored a multi-stage approach including trigger detection, edge detection and event composition. After trigger edge detection is finished we are left with a semantic graph from which we must select the optimal subset that is consistent with the semantic frames for each event type. Previous approaches have derived sub-graph matching rules using heuristics (Jari Björne *et al.* 2009) or machine learning using graph kernels (Liu *et al.*, 2013). Based on McClosky et al. (2011)'s observation that event structures have a strong similarity to dependency graphs, we proposed a novel method for the composition of ambiguous events used a probabilistic variation of the Earley chart parsing algorithm (Stolcke 1995) for finding best derived trigger-argument candidates. Our method uses the event templates and named entity classes as grammar rules. As an additional novel step our chart parsing approach incorporates a linear interpolation mechanism for cross-domain adaptivity between the training and testing (development) data.

2 Approach

The system consists of five main modules: preprocessing, trigger detection, edge detection, simple event extraction, complex event extraction. Each of these is described below with an emphasis on event composition where we applied a probabilistic variation on the Earley parser.

2.1 Experimental Setting

As our team's first attempt at the BioNLP shared task we decided to focus our attention on the Cancer Genetic Task. The CG Task aims to extract events related to the development and progression of cancer.

A characteristic feature of the CG Task is that there are a large number of entity and event types: 18 entity classes, 40 types of event and 8 types of arguments. Among these events, there are 7 that may have no arguments: *Blood vessel development, Cell death, Carcinogenesis, Metastasis, Infection, Amino acid catabolism and Glycolysis*. On the other hand, some events may have more than one argument: *Binding* and *Gene Expression* may have more than one *Theme* argument, and *Planned process* may have more than one *Instrument* argument.

We divided events into two groups based on definitions of Miwa *et al.*(2010) : simple and complex events. Simple events include 36 events whose arguments must be entities. Complex events include 4 event types whose arguments may be other events.

2.2 Pre-processing

Pre-processing conventionally made use of the GeniaTagger (Tsuruoka and Tsujii, 2005) for sentence splitting and tokenizing, and the HPSG parser Enju[1] (Miyao and Tsujii, 2008). Both of these were provided in the supporting resources by the task organisers. Gold standard named entity annotations were also provided.

2.3 Trigger Detection

In the CG Task dataset, 95% of the triggers that indicate events are single token. We therefore treated trigger detection as a token labeling problem in a similar way to Björne *et al.* (2009). Here the system has to classify whether a token acts as a trigger for one of the forty event types or not. We used the Liblinear-java library[2] (Fan *et al.*, 2008) with the L2-regularized logistic regression method for both trigger detection and edge detection. We performed a manual grid search to select a C-value parameter of 0.5. This parameter value is same from that of the Turku system (Björne *et al.* (2009), in which the C-values were tuned for all of their detectors.

The major features used are primarily based on Miwa, *et al.* (2012) and shown in Table 1. In our experiments this led to a relatively large number of features: about 500k features for the trigger detection model, 900k features in the T-E model and 600k features in the EV-EV model. Our choice of the Liblinear library was partly motivated by its efficient performance with large feature sets.

Feature	Target
Token feature	- Current token
Neighbouring word feature	- Current token
Word n-gram feature	- Current token
Trigger dictionary feature	- Current token
Pair n-gram feature	- Between current token and named entities
Parse tree shortest path feature	- Between current token and named entities

Table 1: Features in the trigger detection module.

2.4 Event edge detection

For edge detection, we used Liblinear to construct two models: one model is designed primarily to extract trigger-entity edges (T-E model), while the other system is designed primarily to extract event-event edges (EV-EV model).

The T-E model classifies edge candidates to one of the 8 argument roles (*theme, cause, site, atloc, toloc, fromloc, instrument, participant*) and a negative argument class. Relation pairs are identified through the simple event extraction module (cf Section 2.5).

The EV-EV model identifies relations in the sentences between 4 types of complex events (*Regulation, Negative regulation, Positive Regulation* and *Planned process*) and other events (including events belonging to the 4 complex events). The relations are classified into three classes: the two argument roles (*theme or cause)* or NOT.

The features used in these two models are mostly the same as those used in the earlier trigger detection module. Table 2 shows features and their applied target objects used in T-E model, Table 3 shows features and target objects for each feature of EV-EV model.

Feature	Target
Token feature	- Current trigger - Trigger argument entity
Class feature	- Current trigger - Trigger argument entity
Neighbouring word feature	- Current trigger - Trigger argument entity
Word n-gram feature	- Current trigger - Trigger argument entity
Pair n-gram feature	- Between current trigger and argument entity
Parse tree shortest path feature	- Between current trigger and rigger argument entity

Table 2: Features in the T-E model.

Feature	Target
Token feature	Current trigger, target trigger, current arguments, target arguments
Class feature	Current trigger, target trigger, current arguments, target arguments
Neighbouring word feature	Current trigger, target trigger, current arguments, target arguments
Word n-gram feature	Current trigger, target trigger, current arguments, target arguments
Pair n-gram feature	Between current trigger and target trigger, between current trigger and target arguments, between current arguments and target trigger, between current arguments and target arguments
Parse tree shortest path feature	Between current trigger and target trigger, between current trigger and target arguments, between current arguments and target trigger, between current arguments and target arguments

Table 3: Features in the EV-EV model.

2.5 Simple event extraction

In order to minimise the incorrect combination of arguments and triggers it seemed natural to try and solve the edge classification problem first between triggers and entities (simple edge detection) and then apply these as features in a stacked model to the complex event recogniser (cf Section 2.6). In the simple event extraction module,

[1] http://www-tsujii.is.s.u-tokyo.ac.jp/enju/
[2] http://www.bwaldvogel.de/liblinear-java/

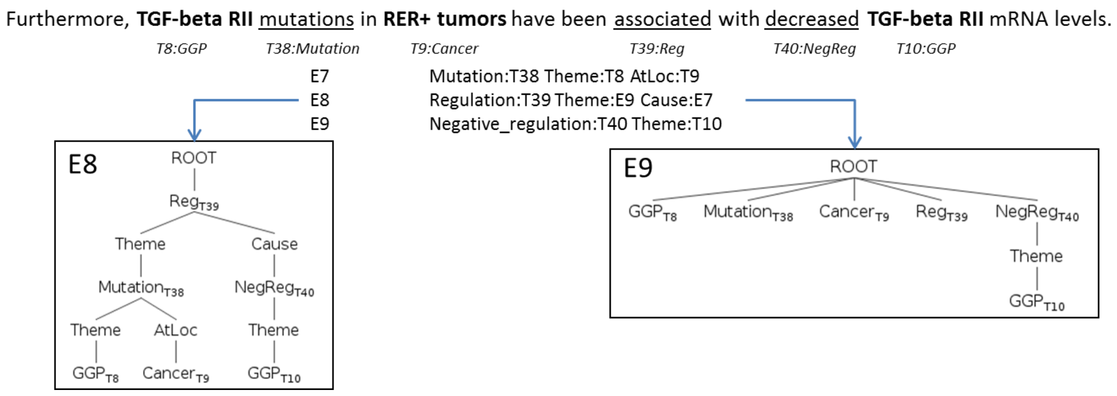
Furthermore, TGF-beta RII mutations in RER+ tumors have been associated with decreased TGF-beta RII mRNA levels.
T8:GGP
T38:Mutation
T9:Cancer
T39:Reg
T40:NegReg
T10:GGP
E7 Mutation:T38 Theme:T8 AtLoc:T9
E8 Regulation:T39 Theme:E9 Cause:E7
E9 Negative_regulation:T40 Theme:T10
E8
ROOT
Reg T39
Theme
Cause
Mutation T38
NegReg T40
Theme
AtLoc
Theme
GGP T8
Cancer T9
GGP T10
E9
ROOT
GGP T8
Mutation T38
Cancer T9
Reg T39
NegReg T40
Theme
GGP T10

Figure 1: An example of representing two complex events as two event trees.

we combined edge candidates identified in the T-E model into complete simple events. After this step, we had the results which belong to the 36 simple event types and relations between 4 complex events and entities.

In order to select the edge candidates for each trigger, we used event-argument pattern based probabilities derived from the training set. An example of a *Development* event-arguments pattern is:

Development → Theme(Gene_expression) + AtLoc(Cancer)

In practice there are several problems that arose when opting for this simple strategy:

- Firstly, there may be multiple candidates with the same argument role label linking to a trigger (such triggers do not belong to *Binding*, *Gene Expression* and *Planned process*). We used the output probability from the logistic regression event edge classifiers to select the best candidate in these cases.

- Secondly, there are triggers whose candidate edge types link to entities that do not match patterns observed in the training set or do not have any relation. We introduced a rule-based semantic post-processing step: triggers are checked to see if they belong to the 7 event types which have no argument; if they do not, we rejected these from the results.

- Thirdly, there may be an imbalance between the argument distribution in the training and testing data (development data). In the development data, we observed some event-argument patterns which do not occur in training set, this problem may lead to false negatives. For example: *Cell_transformation → Theme(Cell) + AtLoc(Cell)* or *Mutation → Site(DNA_domain_or_region)*. This was one cause of false negatives in our system's performance (cf Section 3).

2.6 Complex event extraction with probabilistic Earley Parser

For complex event extraction, based on the idea of McClosky *et al.* (2011) that treats event extraction as dependency parsing, we represent complex events in the form of event trees which are similar to dependency trees. Our idea differs from McClosky *et al.* in that they represented all events in a sentences within a single tree, whereas we build a tree for each complex event. This solution helps avoid the problem of directed cycles if there are two complex event that relate to the same entity or event.

Figure 1 shows an example of representing two complex events as two event trees. To build the event tree, we create a virtual ROOT node; the complex event target will be linked directly to this ROOT node, and triggers and entities that do not belong to sub-structure of the target event will also have links to ROOT node, too. In the event tree, labels of entity classes and event types are retained while terms of triggers and entities are removed.

For event tree parsing, we used the Earley parsing algorithm proposed by Jay Earley (1970) to find alternative structures. The event tree is stored in memory in the form of Earley rules. The inputs to the parser are the entities and triggers which have been identified in the trigger detection module, and the outputs are the event tree candidates.

To choose the best event tree candidates, we built a probabilistic Earley parser which developed from the idea of Hale (2001). As a first attempt at introducing robustness for edge classifier error our parser used linear interpolation on the probability from the edge detection module and the prior edge probabilities to calculate a score for each event tree candidate. The interpolation parameter λ was set using a manual grid search and reflects the confidence we have in the generalisability of the edge detection module on the testing (development) data.

The scoring function for each node is:

$$Score(\text{node}) = \frac{\sum_{edges \in node} P(\text{edge} \mid \text{argrument})}{num(\text{edges})} + P_{\text{Occurrence}}(\text{arguments} \mid \text{node})$$

where,

- *num(edge)* is the number of edges that have a link to the node
- $P_{Occurence}(arguments|node)$ is a distribution which represents the co-occurrence of entity/trigger labels in the arguments of an event type.
- $$P(\text{edge} \mid \text{argrument}) = \lambda * P_{Classifier}(\text{edge} \mid \text{argument}) + (1-\lambda) * P_{Prior}(\text{edge} \mid \text{argument})$$
- λ is a linear interpolation parameter in the range of [0,1]
- $P_{Classifier}(edge|argument)$ is the probability obtained from the edge classifier.
- $P_{Prior}(edge|argument)$ is the training set's prior probability for the edge.

Edges that linked directly to ROOT and did not relate to the target complex event had a default value of zero. The final score of an event tree candidate was calculated as ROOT's value.

We used a *filter_threshold* parameter to remove event tree candidates which had an edge with *P(edge|argument)* less than *filter_threshold.* On the other hand, we used a *cut-off_threshold* parameter to choose event tree candidates which have highest value. Event tree candidates which are sub-structure of other event tree candidates were removed from the final results.

3 Results and Discussion

We evaluated each component of the system on the training and held out data sets. The optimal configuration of parameters was then used on the shared task test data. We set these as follows:*α=0.5;filter_threshold=0.2;cutoff_threshold=0.45.*

Table 4 shows F-score performance for event composition on the development data set. We found that complex events such as regulation and planned process performed at the lower end of accuracy due to their high complexity. This resulted in relatively low recall compared to precision (figures not shown). The three *Regulation* events in particular are very productive in terms of the variety of named entities and triggers they take as arguments and their distribution in the development data was quite different to the training data.

Event	F1	Event	F1
Development	86.67	Phosphorylation	68.45
Blood vessel development	84.15	Dephosphorylation	66.67
Growth	76.77	DNA methylation	85.71
Death	61.95	DNA demethylation	-
Cell death	53.06	Pathway	61.81
Breakdown	77.68	Localization	66.11
Cell proliferation	59.82	Binding	70.68
Cell division	100.00	Dissociation	100.00
Remodeling	60.00	Regulation	69.55
Reproduction	-	Positive regulation	68.13
Mutation	78.74	Negative regulation	68.57
Carcinogenesis	60.67	Planned process	49.99
Metastasis	74.39	Acetylation	100.00
Metabolism	62.50	Glycolysis	69.89
Synthesis	52.63	Glycosylation	-
Catabolism	59.27	Cell transformation	66.67
Gene expression	79.18	Cell differentiation	71.18
Transcription	75.00	Ubiquitination	75.00
Translation	80.00	Amino acid catabolism	100.00
Protein processing	100.00	Infection	75.86
		Total	**73.67**

Table 4: Baseline results for event composition on the development data.

From our analysis on the development set we found that trigger detection was performing well overall with F-scores in the range 78 to 80. We choose 50 false negative events at random for error analysis. There are 29 triggers and 21 events missing. Table 5 shows a stratified analysis by major error type (we note that errors may of course have multiple causes).

Cause	Trigger	Event
Ambiguity in event class	9	
Co-reference	6	
Do not match with any event argument patterns	7	
No training instance	7	4
Choose best argument entity in simple event extraction		5
No argument		4
No Earley parser rule		8
Total	**29**	**21**

Table 5: Error classification of 50 missing false negatives.

Performance on the shared task testing set was overall disappointing with an F-score of 29.94 (Recall = 19.66, Precision = 62.73, F-score of simple event extraction = 47.96 and F-score of complex event extraction = 12.49) indicating low coverage caused by severe over-fitting issues. Analysis revealed that one cause of this was the imbalance in the distribution of arguments between training and testing sets.

4 Conclusion

We presented a system built on supervised machine learning with rich features, semantic post-processing rules and the dynamic programming Earley parser. The system achieved an F-score of 29.94 on the CG task with high precision of 62.73. Future work will focus on extending recall for complex events and looking at how we can avoid over-fitting to benefit cross-domain adaptivity.

Acknowledgements

We thank the shared task organisers for supporting this community evaluation and to the supporting resource providers. Nigel Collier also gratefully acknowledges funding support from the European Commission through the Marie Curie International Incoming Fellowship (IIF) programme (Project: Phenominer, Ref: 301806).

References

David McClosky, Mihai Surdeanu, and Chris Manning. 2011. *Event extraction as dependency parsing*. In Proceedings of the BioNLP Shared Task 2011 Workshop at the Association for Computational Linguistics Conference, pp. 41-45.

Jari Björne, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. *Extracting complex biological events with rich graph-based feature sets*. In Proceedings of the BioNLP 2009 Shared Task Workshop at the Association for Computational Linguistics Conference, pp. 10–18.

Jari Björne, Filip Ginter, Tapio Salakoski: University of Turku in the BioNLP'11 Shared Task. BMC Bioinformatics 13(S-11): S4 (2012)

Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. 2008. *LIBLINEAR: A library for large linear classification*. J Machine Learn Res 9:1871–1874.

Miwa, M., Thompson, P., McNaught, J., Kell, D., Ananiadou, S. 2012. *Extracting semantically enriched events from biomedical literature*. BMC Bioinformatics 13, 108.

Makoto Miwa, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010. *Event extraction with complex event classification using rich features*. Journal of Bioinformatics and Computational Biology (JBCB), 8(1):131–146.

Earley, Jay (1970). *An efficient context-free parsing algorithm*. Communications of the ACM 13 (2): 94–102

Andreas Stolcke (1995). *An efficient probabilistic context-free parsing algorithm that computes prefix probabilities*. Journal Computational Linguistics (1995) Volume 21 Issue 2: 165-201

Sampo Pyysalo, Tomoko Ohta and Sophia Ananiadou. (2013). *Overview of the Cancer Genetics (CG) task of BioNLP Shared Task 2013*. Proceedings of BioNLP Shared Task 2013 Workshop at the Association for Computational Linguistics Conference, (in press).

Sampo Pyysalo, Tomoko Ohta, Makoto Miwa, Han-Cheol Cho, Jun'ichi Tsujii and Sophia Ananiadou. (2012). *Event extraction across multiple levels of biological organization*. Bioinformatics, 28(18):i575-i581.

Haibin Liu, Lawrence Hunter, Vlado Kešelj, Karin Verspoor (2013). *Approximate Subgraph Matching-Based Literature Mining for Biomedical Events and Relations*. PLOS ONE, 2013.