Journal of Bioinformatics and Computational Biology
Vol. 8, No. 1 (2010) 131–146

DOI: 10.1142/S0219720010004586

Imperial College Press
www.icpress.co.uk

EVENT EXTRACTION WITH COMPLEX EVENT CLASSIFICATION USING RICH FEATURES

MAKOTO MIWA[*,§], RUNE SÆTRE[*,¶], JIN-DONG KIM[*,‖] and JUN'ICHI TSUJII[*,†,‡,**]

[*]*Department of Computer Science, University of Tokyo*
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan

[†]*School of Computer Science, University of Manchester, UK*

[‡]*National Center for Text Mining, UK*

[§]*mmiwa@is.s.u-tokyo.ac.jp*
[¶]*rune.saetre@is.s.u-tokyo.ac.jp*
[‖]*jdkim@is.s.u-tokyo.ac.jp*
[**]*tsujii@is.s.u-tokyo.ac.jp*

Received 4 December 2009
Revised 22 December 2009
Accepted 5 January 2010

Biomedical Natural Language Processing (BioNLP) attempts to capture biomedical phenomena from texts by extracting relations between biomedical entities (i.e. proteins and genes). Traditionally, only binary relations have been extracted from large numbers of published papers. Recently, more complex relations (biomolecular events) have also been extracted. Such events may include several entities or other relations. To evaluate the performance of the text mining systems, several shared task challenges have been arranged for the BioNLP community. With a common and consistent task setting, the BioNLP'09 shared task evaluated complex biomolecular events such as binding and regulation.Finding these events automatically is important in order to improve biomedical event extraction systems. In the present paper, we propose an automatic event extraction system, which contains a model for complex events, by solving a classification problem with rich features. The main contributions of the present paper are: (1) the proposal of an effective bio-event detection method using machine learning, (2) provision of a high-performance event extraction system, and (3) the execution of a quantitative error analysis. The proposed complex (binding and regulation) event detector outperforms the best system from the BioNLP'09 shared task challenge.

Keywords: Event extraction; Support Vector Machine; complex event detection.

1. Introduction

Relations among biomedical entities (i.e. proteins and genes) are important in understanding biomedical phenomena and must be extracted automatically from a large number of published papers. Most researchers in the field of Biomedical Natural Language Processing (BioNLP) have focused on extracting binary relations, including protein–protein interactions (PPIs)[1,2] and disease–gene associations.[3]

Binary relations are not sufficient for capturing biomedical phenomena in detail, and there is a growing need for capturing more detailed and complex relations. For this purpose, two large corpora, BioInfer[4] and GENIA,[5] have been proposed. The BioNLP'09 shared task[6,a] recently provided common and consistent task definitions, data sets, and evaluation criteria. The BioNLP'09 shared task contains simple events and complex events. Whereas the simple events consist of binary relations between proteins and their textual triggers, the complex events consist of multiple relations among proteins, events, and their textual triggers. Bindings can represent events among multiple proteins, and regulations can represent causality relations between proteins and events. These complex events are more informative than simple events, and this information is important in modeling biological systems, such as pathways.

In the present paper, we propose a system that enables the extraction of complex events. The proposed system generally follows the processing pipeline of the system reported by Björne *et al.*,[7] which was the best system in the BioNLP'09 shared task, and is referred to hereinafter as the Turku system. By solving a new classification problem, the proposed system constructs a model for extracting complex events using rich features. Using this model, the proposed complex event detector is demonstrated to perform better than the Turku system in finding complex events. We also present the results of an error analysis, which revealed several problems with the proposed system that must be resolved.

2. Related Research

The BioNLP'09 shared task included three subtasks: finding core events (Task 1), finding the secondary arguments (such as location and sites) (Task 2), and recognizing speculation and negation (Task 3).[6] The events included five simple events (Gene_expression, Transcription, Protein_catabolism, Phosphorylation, and Localization) and four complex events (Binding, Regulation, Positive_regulation, and Negative_regulation). A simple event is an event that includes only a single primary theme protein, and a complex event is an event that includes multiple primary theme and cause arguments (proteins and events).

The proposed system targets Task 1, and the goal of which is to identify events along with their types, textual triggers, and primary theme and cause arguments. Textual triggers are tokens that represent the events. In the following, we will explain two related systems participating in the BioNLP'09 shared task. These systems were developed in a limited time enforced by the BioNLP'09 shared task.

Björne *et al.*[7] introduced the system that had the best performing system for the task. The Turku system is used as a baseline for comparison purposes in Sec. 4. A pipeline method is adopted by the Turku system for the extraction of events, and the proposed system follows the processing pipeline. The Turku system first finds

[a]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

triggers and then attempts to extract the event edges, ultimately combining edges that share the same triggers to extract complex events with a rule-based module.

Sætre *et al.*[8] introduced a system that treated complex events with a classifier. This system is similar to the proposed system. Their system also uses a pipeline method and finds triggers first. In finding events, their system is different from the Turku system. Complex events are treated, not as combinations of edges, but rather as event instances, without considering the dependencies among the events. Using the treatment of the complex events and features that are known to be effective for the extraction of PPIs that are related to binding events,[2] their system performed well in finding binding events.

3. Event Extraction System

The proposed event extraction system basically follows the processing pipeline of the Turku system, i.e. trigger detection, edge detection, and complex event detection. For extracting complex events, instead of applying rules, the proposed system solves a new classification problem and constructs a new model. Figure 1 exemplifies the flow of the event extraction. All modules solve classification problems to construct models. The differences between the proposed system and the Turku system lie in the features that include the parsers and the additional features, classification problems that include labels and problem separation, and machine-learning–based complex event detection.

In this section, we explain the proposed system, as well as the above-mentioned differences. We first introduce classifiers and their settings in Sec. 3.1. Then, we explain the proposed preprocessing method in Sec. 3.2. Finally, we present three

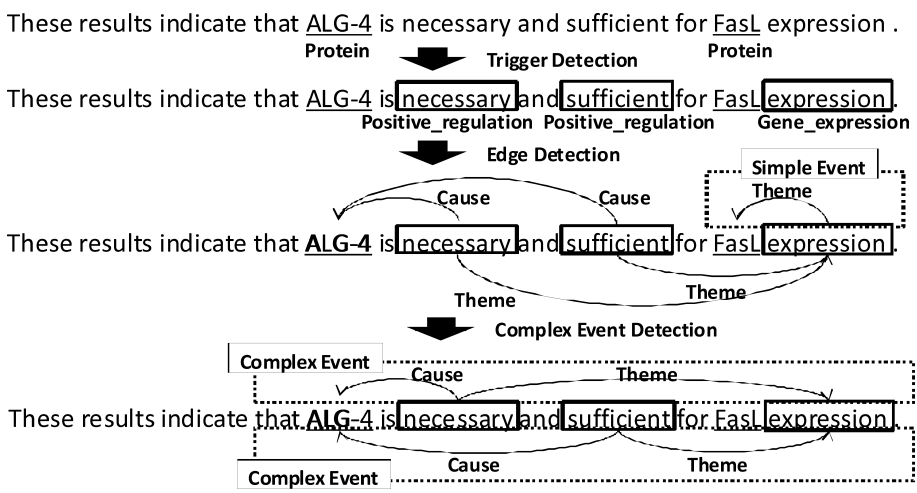

Fig. 1. Flow of event extraction. Proteins are provided. A trigger detector first finds triggers along with their classes. An event edge detector then finds edges, some of which are simple events. A complex event detector then combines edges to construct complex events.

modules with problems to be solved: trigger detector in Sec. 3.3, event edge detector in Sec. 3.4, and complex event detector in Sec. 3.5.

3.1. *Classifier*

For the construction of the proposed event extraction system, we solve multi-class classification problems and multi-label classification problems. We use one-versus-rest support vector machines (SVMs)[9] to solve these problems and fit a sigmoid function to the outputs by SVMs to calculate the confidences of the examples.[10]

3.2. *Preprocessing*

All of the sentences in the data set are parsed using two parsers: a deep parser Enju 2.3.1[11] and a dependency parser GDep beta1.[12] From a sequence of words of the sentence, the deep parser constructs graph structures that represent theory-specific syntactic/semantic relations among words, and the dependency parser constructs a dependency tree consisting of dependency links between words. We use the predicate argument structure (PAS) from the deep parser, along with the dependency trees from the dependency parser. To lessen the effect of the inconsistency among the arguments in a coordination structure in PAS, we convert the structures of the coordinations into flat structures, as shown in Fig. 2, for the deep parser.

We constructed a trigger dictionary by extracting all of the triggers normalized by the parsers. This dictionary is used to detect triggers in Table 2 and to find triggers that are substrings of a word segmented by the parsers. To find these triggers, we segment a word by splitting it with "-" if the following conditions are satisfied: (i) the word is not in the trigger dictionary, (ii) the word includes "-", and (iii) more than one of the resulting words are in the dictionary.

We use the "equiv" annotation in the data set (a2 files), which indicates equivalent protein mentions in a sentence. If the gold events (or annotated events) in the training data set contain equivalent proteins, we create all equivalent events to remove inconsistent negative examples and increase the number of positive examples. Considering the test data set setting, we remove the events that are sub-events of other events.

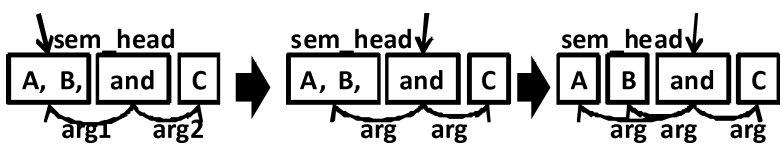

Fig. 2. Modification of the coordination structure in predicate argument structures produced by Enju. The semantic head (sem_head) of the phrase including a coordination is moved to the coordinating phrase. Argument names are converted to "arg", instead of "arg1" and "arg2", and the same level tokens are moved as the arguments of the coordinating phrase.

Table 1. Features for tokens (token features) and for the shortest paths (SPs) between two entities (SP-features).

Type	Features
Token	Token has a capital letter
	Token has a first letter of the sentence
	Token is in the trigger dictionary
	Token has a number
	Token has a symbol
	Token is in a protein
	N-grams ($n = 1, 2, 3, 4$) of characters
	Base form
	Token is after "-"
	Entries (e.g. part-of-speech, lexical entry) of token in the outputs of parsers
Shortest path	Vertex walks and their sub-structures
	Edge walks and their sub-structures
	N-grams of dependencies ($n = 2, 3, 4$)
	N-grams of words (base form + POS) ($n = 2, 3, 4$)
	N-grams of consecutive words (base form + POS) representing governor-dependent relationships ($n = 1, 2, 3$)
	Lengths of paths

3.3. *Trigger detection*

For trigger detection, we need to find two types of triggers: the trigger for a trigger–protein relation (TP-T) and the trigger for a trigger–trigger relation (TT-T). For the detection of these two types of triggers, we constructed two machine learning systems. One system (TP-T detector) is designed primarily to extract TP-T, while the other system (TT-T detector) is designed primarily to extract TT-T. Since the TT-T detector requires the information of triggers, we first train the TP-T detector and then train the TT-T detector using the outputs of the TP-T detector. This formulation of the problem is different from that in the Turku system, although how they perform their detection is not specified in detail in their paper.

Figure 3 shows the flow of trigger detection. The trigger detectors target all of the words in the data sets. Both of the detectors attempt to classify all of the words into event classes, including a negative event class (to extract event trigger words). The event classes of gold triggers surrounding a word (e.g. Binding) are used as the labels of a word in the classifications. Words are used as positive examples if they have more than one target label, and other words are used as negative examples. We extract rich features to represent the words, as shown in Table 2. For the TP-T detector, in addition to the same features for the trigger detection in Turku system, we extract the shortest path features between the event trigger candidate and the closest proteins (in the parser output) in order to include the information of supporting proteins. The shortest path features contain the features for event edge detection in the Turku system, as well as several additional n-grams and substructures, as shown in Table 1, which were used by Sætre *et al.*[8] As features for the TT-T detector, we add two types of features to the features in the TP-T

Table 2. Features of a trigger candidate for a TP-T (trigger for trigger–protein relation) detector.

Type	Features
Token	Token features in Table 1
Two words from candidate in parser output	Token features of word with dependencies from candidate N-grams (n=2) of dependencies N-grams ($n = 2, 3$) of words (base form + POS) N-grams ($n = 2, 3, 4$) of dependencies and words
Three words around candidate	N-grams ($n = 1, 2, 3, 4$) of words
Shortest paths	SP-features in Table 1 between candidate and the closest proteins (in the parser output) Lengths of paths between candidate and proteins

Table 3. Additional features of a trigger candidate for a TT-T (trigger for trigger–trigger relation) detector.

Type	Features
Token confidence	Confidences for all event classes by a TP-T detector
Shortest paths	SP-features in Table 1 between candidate and the other closest candidates by TP-T detector for all event classes

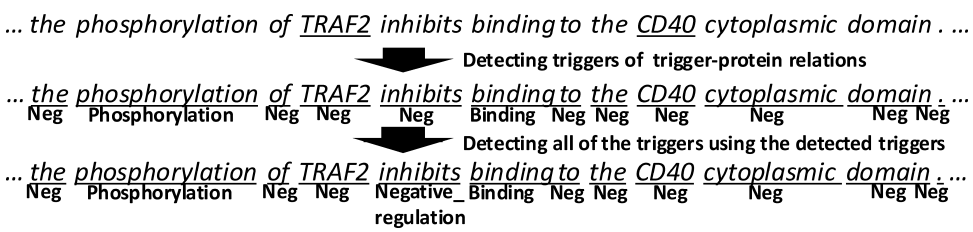

Fig. 3. Flow of trigger detection. Trigger detectors first attempt to find the triggers for trigger–protein relations and then find all of the triggers using the information of the detected triggers as features. Trigger detectors classify all of the words. The labels are the event classes.

detector, as shown in Table 3. One type is the confidence (for all event classes) of the event trigger candidate predicted by the TP-T detector; the other type is the shortest path between the event trigger candidate and the closest trigger for all event classes detected by the TP-T detector.

3.4. *Event edge detection*

For edge detection, we select event edges from edges among detected triggers and named entities (proteins). An edge contains an event trigger node and an argument terminal node, which is a trigger or a protein. Using this edge detection, we can find simple events, which have only one argument.

Table 4. Features of an event edge candidate for an edge detector.

Type	Features
Each terminal node	Token features in Table 1 of terminal node Confidences for all event classes by TT-T detector
Three words around the pair	N-grams ($n = 1, 2, 3, 4$) of words
Shortest paths	SP-features in Table 1 between terminal nodes SP-features in Table 1 between the argument trigger and the closest proteins

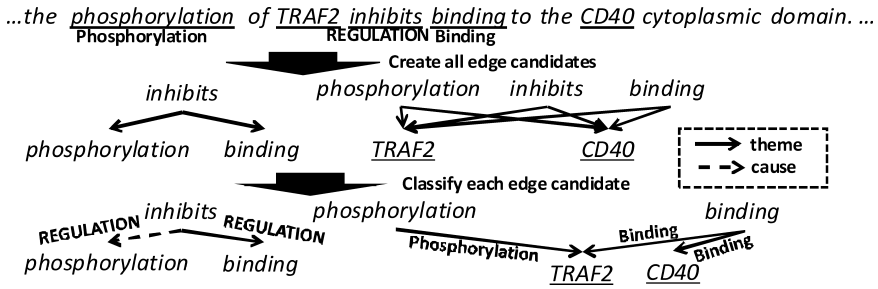

Fig. 4. Flow of edge detection. Edge detectors create all possible edges and classify all of the edges. The labels indicate the combinations of event classes and edge types.

We solve two separate classification problems: the trigger–trigger edge detection problem and the trigger–protein edge detection problem. Figure 4 shows the flow of edge detection. We use all regulation events as a single event class for the edge detection and the following complex event detection, and the combinations of event classes and edge types (theme or cause) are used as the labels of edges (e.g. Binding:Theme). All detected triggers (detected by the TT-T detector in Sec. 3.3) are used, and we create positive and negative examples from the edges among the triggers and named entities. We extract the features of an event edge candidate for edge detection, as shown in Table 4. In addition to the features for the edge detection in the Turku system, we use the confidences of terminal nodes obtained by trigger detection for the features. We also add the shortest path features between the argument trigger in the trigger–trigger edge node and the closest proteins in order to add information of the supporting proteins to the features.

For trigger–trigger edges, the latter terminal node must be on another edge. The edges were checked recursively, and unacceptable edges were removed until all of the edges met the appropriate condition.

3.5. *Complex event detection*

Complex events can be represented by finding the best combinations of event edges that are detected by the edge detector in Sec. 3.4. To find the appropriate combinations, we construct a complex event detection system. The Turku system combined

Table 5. Features of a complex event candidate for a complex event detector.

Type	Features
Each event edge	Edge features in Table 4
All pairs among arguments	Edge features in Table 4 except shortest paths between an argument trigger and the closest proteins
All edges including event trigger outside of events	Edge features in Table 4
All pairs between argument proteins and their closest proteins in binding	Edge features in Table 4

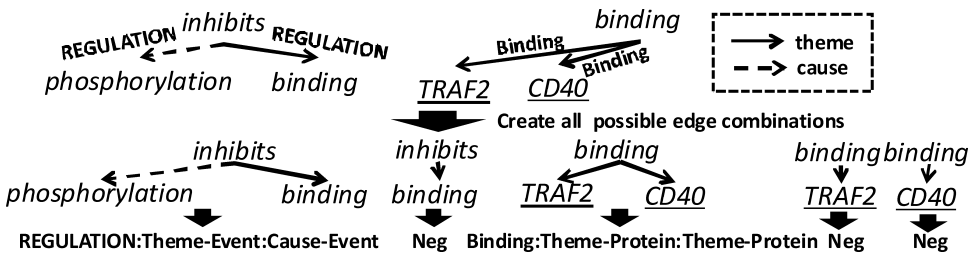

Fig. 5. Flow of complex event detection. Complex event detectors create all possible combinations of detected event edges and classify all of the edges. The labels indicate the combinations of the event class, the edge types, and the connected terminal node types.

the edges with rules, and their rules could capture the majority of the appropriate combinations. Another possible approach for finding the appropriate combinations is a machine-learning–based approach. This type of approach can automatically construct models from the training data and find combinations missed by a rule-based system. We construct classification models for the complex event detection. In this approach, events are selected from event candidates constructed by combining event edges.

We solve two separate classification problems (Binding and Regulation) for four complex event classes. Figure 5 shows the flow of complex event detection. We treat all of the regulations as a single event class (such as edge detection). For each problem, the event class, the edge types, and the connected terminal node types (event or protein) are used as the labels of complex events (e.g. Regulation:Theme-Event:Cause-Protein, Binding:Theme-Protein:Theme-Protein). We then create positive and negative examples from the combinations of detected event edges. We design features considering the edges inside and outside of the events. The arguments of the inside edges should interact with each other, and the arguments of the outside edges should not interact with the arguments of the inner edges. We extract the features of a complex event candidate for complex event detection, as shown in Table 5, using the feature extractor for the edge detection in Sec. 3.4. The features contain three relations: relations between arguments, relations between triggers and outer proteins, and relations between arguments and outer nodes. The outer nodes

(proteins) are nodes (proteins) that are not included in the event candidate. The features are a combination of the features in Table 4 for several edges and are designed to remove inappropriate event candidates. The first relations are used to remove candidates that contain non-related arguments, and the second and third relations are used to remove candidates by finding edges that should be included in the candidates and more appropriate combinations of event edges.

4. Evaluation

4.1. *Evaluation settings*

We evaluated the performance of the proposed system using the evaluation script[b] for the development data set and the evaluation system[c] for the test data set. The script and the evaluation system are provided by the BioNLP'09 shared task organizers. Errors were also analyzed using the development data set.

Liblinear-java[9,d] was used as the one-versus-rest SVMs explained in Sec. 3.1. The proposed system contains many classification problems, and tuning thresholds for each problem require significant computational cost. We used the same settings for all of the problems. The one-versus-rest SVMs must solve several unbalanced classification problems. To ease the problem, we balanced the positive and negative examples by placing more weight on the positive examples. To obtain a selection of as many confident examples as possible, we selected examples with confidences of greater than 0.5, in addition to the examples with the most confident labels. The C-values were set to 1.0. This set up is different from that of the Turku system, in which the C-values and thresholds were tuned for all of their detectors. This parameter tuning is left as future work and this will be discussed in Sec. 5.

4.2. *Performance*

Table 6 shows the performance of the proposed system on the development data set produced by the evaluation script, the performance on the test data set produced using the evaluation system, and the performance of the Turku system on the test data set. Table 7 shows the performance with event decomposition in finding complex events. Table 8 summarizes these two tables for the comparison of the results obtained using the proposed system and the results obtained using the Turku system.

The proposed system is comparable to the Turku system in finding simple events and regulations, and the proposed system performed much better than the Turku system in finding binding events. With respect to overall performance, the proposed system outperformed the Turku system.

[b]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/downloads.shtml
[c]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/eval-test.shtml
[d]http://www.bwaldvogel.de/liblinear-java/

Table 6. Approximate span matching/approximate recursive matching on development (Dev) data set, test data set, and test data set with the Turku system.

	Dev data set			Test data set			Turku (test)		
Event class	recall	prec.	fscore	recall	prec.	fscore	recall	prec.	fscore
Gene_expression	78.65	79.49	79.07	68.70	79.87	73.86	69.81	78.50	73.90
Transcription	65.85	71.05	68.35	54.01	60.66	57.14	39.42	69.23	50.23
Protein_catabolism	95.24	90.91	93.02	42.86	75.00	54.55	42.86	66.67	52.17
Phosphorylation	85.11	68.97	76.19	84.44	69.51	76.25	80.74	74.66	77.58
Localization	71.70	82.61	76.77	47.13	86.32	60.97	49.43	81.90	61.65
=[SVT-TOTAL]=	77.28	77.94	77.61	65.31	76.44	70.44	64.21	77.45	70.21
Binding	50.81	47.55	49.12	52.16	53.08	52.62	40.06	49.82	44.41
=[EVT-TOTAL]=	69.14	68.09	68.61	62.33	70.54	66.18	58.73	71.33	64.42
Regulation	36.69	46.62	41.06	28.87	39.81	33.47	25.43	38.14	30.52
Positive_regulation	43.92	51.92	47.59	38.05	48.32	42.57	38.76	48.72	43.17
Negative_regulation	38.78	43.93	41.19	35.88	47.22	40.78	35.36	43.46	38.99
=[REG-TOTAL]=	41.65	49.40	45.19	35.93	46.66	40.60	35.63	45.87	40.11
=[ALL-TOTAL]=	54.05	58.69	56.27	48.62	58.96	53.29	46.73	58.48	51.95

Table 7. Event decomposition/approximate span matching/approximate recursive matching in finding complex events on the test data set with the proposed system and with the Turku system.

	Test data set			Turku (test)		
Event class	recall	prec.	fscore	recall	prec.	fscore
Binding	58.79	73.12	65.18	53.27	64.63	58.40
Regulation	32.25	45.61	37.78	28.11	44.19	34.36
Positive_regulation	43.76	58.05	49.90	44.60	59.91	51.14
Negative_regulation	38.94	52.43	44.69	38.46	51.61	44.08
=[REG-TOTAL]=	40.72	54.79	46.72	40.41	55.68	46.83

In the complex event detection, the proposed approach is better than the rules of the Turku system, as indicated in Table 8. The loss in the event composition of the proposed approach is less than that of the rules of the Turku system.

For binding events, the F-score is much better than for the other systems that were submitted to the BioNLP'09 shared task on the test data set. The performance is better for the test data set than for the development data set with respect to binding. This is partially because the evaluation script does not consider the "equiv" annotation, as shown in Sec. 3.2. The script can output a lower score than

Table 8. Comparison of the results for F-score obtained using the proposed system and the results obtained using the Turku system on the test data set. F-scores in parentheses indicate the results of event decomposition/approximate span matching/approximate recursive matching.

	Simple	Binding	Regulation	All
Ours	70.44 (70.44)	52.62 (65.18)	40.60 (46.72)	53.29 (57.09)
Turku	70.21 (70.21)	44.41 (58.40)	40.11 (46.83)	51.95 (56.29)

the evaluation system. The event decomposition results imply that the additional features are useful for finding binding events.[8] The loss in event composition shows that the proposed complex event detector is useful for finding complex binding events. The correct combinations of arguments are selected by using the proposed rich feature vector.

For regulation events, the F-score on the development data set decreased by approximately 2% when we set the threshold for the complex events to zero. This setting without the threshold is the same as the rules in the Turku system. The results of this setting indicate that, in finding regulation events, the proposed system could not produce results that were comparable to those of the Turku system with respect to edge detection. This weakness in finding edges is also observed in the event decomposition results in Table 7. The proposed system performed comparably to the Turku system with respect to event detection, and the complex event detector may also improve the Turku system. However, the complex event detector did not drastically affect the finding of regulation events. This is because most of the causal events could not be found by the edge detector and the threshold for the edge detection was too strict. Causes are not easy to find because most downstream events are selected as causes in regulation events, and the processing pipeline of events must be resolved in order to find causes.

4.3. *Error analysis*

For the further improvement and practical application of the event extraction system, an improvement in recall is necessary. Table 9 summarizes the analysis of 100 missing false negatives.

A total of 61 triggers were missing from the 100 errors. Ten of the errors include coreferences/exemplification problems between the trigger and the theme or cause, and the distance from the trigger to the protein was great. These problems include

Table 9. Error classification among 100 missing false negatives on the development data set. A total of 61 triggers were missing, and 39 events were missing.

Cause\missing type	Trigger	Event
Missing theme and/or cause	9	10
Coreferences/Exemplification	10	7
No training instance	7	—
Ambiguity in event classes	7	—
Binding (de)composition	—	6
Self interaction	3	—
Parse problem	3	—
Inference	3	2
Regulation hierarchy	—	2
Hidden subject/object	—	2
Threshold	19	10
Total	61	39

pronouns, anaphora, and apposition. In nine regulation events, the themes and causes were not found as triggers, even though the events have only event classes as their arguments. Finding these events without clues is difficult. Seven errors were missing because of a lack of training instances, and some triggers did not appear in the training data set. We may be able to discover some of the triggers by using other resources, such as variations of terms, to find such instances. Seven errors were caused by ambiguity in the event classes for clues. For example, the word "induction" can be a gene_expression, transcription, or positive_regulation. In these errors, the system could not disambiguate the event classes, and so the system designated these errors as different types. These errors also include a few difficult cases, which can be ambiguous for the annotators (such as the ambiguity between regulation and positive/negative regulation). Three errors include self interactions, such as transfections, which can be regulation events without other triggers or proteins in the BioNLP'09 shared task data set. Three errors are caused by parse problems, with PP-attachment problems, and three errors contain inference problems. The remaining 19 errors were caused primarily by the threshold.

For cases in which events are missing, 39 errors were found. Seven errors were caused by coreferences. Ten regulation events were not found because of missing themes and/or causes. Six errors were caused by the wrong composition of edges in complex binding events. Two errors contain inference problems. Two errors occurred because hidden subjects or objects of triggers could not be resolved. Two regulation events, including the causes, were missed because the proposed system had some difficulty in resolving the event hierarchy explained in Sec. 4.2.

5. Discussion

One disadvantage of the proposed system is the problem of properly tuning the following three mutually dependent sets of parameters (see Sec. 3 for details): (1) "learning parameters", which shift the ranges of predicted confidence values in each module, (2) thresholds for including trigger, edge, and non-recursive event instances in the edge and event modules, and (3) thresholds for including recursive (regulation) events. Non-recursive events have only proteins as their arguments, while recursive events always have at least one trigger (i.e. "event") as their argument.

Among the *learning parameters*, normalization of the features and the regularization parameters decide the range of confidence values. Furthermore, the weights on positive and negative examples decide the precision — recall break-even point.

As for thresholds, removing too many uncertain triggers and edges may also cause removal of candidates, which would otherwise be classified as certain events in following detection modules. To improve the recall, one possible approach is to use the confidence of all of the predicted candidates as features in the following modules. However, this is not trivial, because of the large number of candidates, and the uncertainty involved in computing the confidence values.

Regulation events are different from other events, because they may have other events as their argument, and they may also have a cause argument. In order to better treat regulation events, the predicted confidence values of the argument events should also be considered. This means that we first have to find all non-recursive events and then use that information to find the recursive regulation events.

For the event extraction, the analysis in Sec. 4.3 illustrated several types of NLP problems. The analysis indicated that 71% of the errors are related to missing triggers. In finding both the triggers and events, coreferences are included as a major error type and directly or indirectly play a large role in the missing trigger problems. Coreferences increase the distance between a trigger and the arguments. Coreferences also cause errors in finding PPIs, and it is necessary to focus on resolving these errors in order to find the biomedical relations.[2]

The BioNLP'09 shared task data sets contain two problems that need to be avoided for the construction of more consistent event extraction systems.

One problem involves the selection of the target proteins and events.[6] The BioNLP'09 shared task data sets were generated from the GENIA corpus.[5] Through the selection, a number of meaningless or incomplete events remained in the BioNLP'09 shared task data sets. Moreover, a number of events were missing. These events should be between regulation events and the causal events for representing the hierarchy of events. This problem may be diminished by using the GENIA corpus instead of the BioNLP'09 shared task data sets or by removing these events from the BioNLP'09 shared task data set using a number of rules.

The other problem is the annotation inconsistency in triggers. Annotations for some triggers can be ambiguous, and the policy for the selection of the description is not strictly defined.[5] The evaluation scripts alleviate this problem in the evaluation, but this type of annotation inconsistency can confuse classifiers. Trigger detections should be helpful for finding events in learning and prediction, but detecting triggers is less important than finding event classes as the results of an event extractor. Therefore, more weight should be placed on the event classes that involved the proteins. We can attempt to evaluate the performance of the proposed system considering the event classes without the triggers for a more general analysis of the proposed event extraction system. This evaluation can then clarify the issues that require greater focus. Considering this evaluation policy, we can devise other approaches to find events. For example, first finding event types that include proteins, instead of triggers, is one possible approach.

For the BioNLP'09 shared task evaluation system, we observed that the recall tended to be lower in the test data set than in the development data set. In order to avoid tuning the recall for the system (which was not allowed in the challenge) and compare the event extraction systems from different points of view, we need more evaluation criteria than simply the F-score. We can use the area under the receiver operating characteristic curve (ROC) (referred to as the AUC) used in the evaluations of recent PPI extraction systems.[1,2]

6. Conclusion

In the present paper, we proposed an event extraction system that focuses primarily on the extraction of complex events. The proposed complex event detector performed better than the rule-based detector in the best system, i.e. the Turku system, and the proposed system performed better than the other systems in the BioNLP'09 shared task data set. Note that the other systems were developed within the limited time period that was permitted for the BioNLP'09 shared task.

We also analyzed false negatives in the development data set and showed that the missing triggers caused 71% of all errors and that coreferences were a major problem. The proposed system is integrated into U-compare.[13]

Based on the error analysis, we need to integrate a coreference resolution system. Here, the coreferences are the combination of several problems, including pronouns, anaphora, and apposition. We will continue to analyze errors in order to determine the problems that must be addressed in order to improve the general event extraction system. Furthermore, we need to determine the problems caused by the BioNLP'09 shared task settings. Focusing on the former problems would further improve the general event extraction system.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

1. Airola A, Pyysalo S, Björne J, Pahikkala T, Ginter F, Salakoski T, All-paths graph kernel for protein–protein interaction extraction with evaluation of cross corpus learning, *BMC Bioinformatics* **9**:S2, 2008.
2. Miwa M, Sætre R, Miyao Y, Tsujii J, A rich feature vector for protein–protein interaction extraction from multiple corpora, in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 121–130, August 2009.
3. Chun H-W, Tsuruoka Y, Kim J-D, Shiba R, Nagata N, Hishiki T, Tsujii J, Extraction of gene-disease relations from medline using domain dictionaries and machine learning, in *The Pacific Symposium on Biocomputing (PSB)*, pp. 4–15, 2006.
4. Pyysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Järvinen J, Salakoski T, BioInfer: A corpus for information extraction in the biomedical domain, *BMC Bioinformatics* **8**:50, 2007.
5. Kim J-D, Ohta T, Tsujii J, Corpus annotation for mining biomedical events from literature, *BMC Bioinformatics* **9**:10, 2008.
6. Kim J-D, Ohta T, Pyysalo S, Kano Y, Tsujii J, Overview of BioNLP'09 shared task on event extraction, in *BioNLP '09: Proceedings of the Workshop on BioNLP*, pp. 1–9, 2009.
7. Björne J, Heimonen J, Ginter F, Airola A, Pahikkala T, Salakoski T, Extracting complex biological events with rich graph-based feature sets, in *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pp. 10–18, 2009.

8. Sætre R, Miwa M, Yoshida K, Tsujii J, From protein-protein interaction to molecular event extraction, in *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pp. 103–106, 2009.
9. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J, LIBLINEAR: A library for large linear classification, *J Machine Learn Res* **9**:1871–1874, 2008.
10. Platt JC, Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods, in *Advances in Large Margin Classifiers*, pp. 61–74, 1999.
11. Miyao Y, Sætre R, Sagae K, Matsuzaki T, Tsujii J, Task-oriented evaluation of syntactic parsers and their representations, in *Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08:HLT)*, 2008.
12. Sagae K, Tsujii J, Dependency parsing and domain adaptation with LR models and parser ensembles, in *EMNLP-CoNLL 2007*, 2007.
13. Kano Y, Baumgartner W, McCrohon L, Ananiadou S, Cohen K, Hunter L, Tsujii J, U-Compare: Share and compare text mining tools with UIMA, *Bioinformatics* **25**(15):1997–1998, 2009.

Makoto Miwa received his M.Sc. and Ph.D. degrees, both in science, from the University of Tokyo, Tokyo, Japan, in 2005 and 2008, respectively. He has since been working as a researcher at the University of Tokyo under the supervision of Prof. Jun'ichi Tsujii. His interests include machine learning in natural language processing for bio-medical texts (BioNLP) and computer games.

Rune Sætre received his M.Sc. and Ph.D. degrees, both in Computer Science, from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2003 and 2006, respectively. He has since been working as a researcher at the University of Tokyo, under the supervision of Prof. Jun'ichi Tsujii. His primary interest is natural language processing for bio-medical texts (BioNLP) and he is collaborating with NTNU's cancer research and computer science groups, in order to make useful real-life applications. He is working on Relation Extraction in the PathText project, and he is funded by the bilateral Japan-Slovenia AGRA project on Gene Ranking Algorithms.

Jin-Dong Kim received his M.Sc. and Ph.D. degrees in natural language processing (NLP) from Korea University in 1996 and 2000, respectively. In 2001, he joined Tsujii Laboratory at the University of Tokyo. Since then, he has mainly worked for the GENIA project, a bio-text mining project. He is one of the main authors of the GENIA resources and one of the organizers of the BioNLP shared tasks for 2004 and 2009. He is currently working as a project lecturer at the University of Tokyo.

Jun'ichi Tsujii received his B.Eng, M.Eng. and Ph.D. degrees in Electrical Engineering from Kyoto University, Japan, in 1971, 1973, and 1978, respectively. He was an Assistant Professor and Associate Professor at Kyoto University before taking up the position of Professor of Computational Linguistics at the University of Manchester's Institute for Science and Technology (UMIST) in 1988. Since 1995, he has been a Professor of Department of Computer Science, the University of Tokyo. He is also a Professor of Text Mining at the University of Manchester (part-time) and Research Director of the UK National Centre for Text Mining (NaCTeM) since 2004. He was President of the Association for Computational Linguistics (ACL) in 2006 and has been a permanent member of the International Committee on Computational Linguistics (ICCL) since 1992.