Event Extraction with Complex Event Classification using Rich Features

Makoto Miwa[†], **Rune Sætre**[†], **Jin-Dong Kim**[†], **and Jun'ichi Tsujii**[†‡*]
[†]Department of Computer Science, the University of Tokyo, Japan
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan.
[‡]School of Computer Science, University of Manchester, UK
[*]National Center for Text Mining, UK
{mmiwa,rune.saetre,jdkim,tsujii}@is.s.u-tokyo.ac.jp

Abstract

To capture biomedical phenomena more deeply, it is required to extract relations that are more complex than binary relations. To extract such complex relations, the BioNLP'09 shared task provided complex events; binding and regulation were provided as complex relations. To improve the biomedical event extraction systems, finding these complex events automatically is important; thus, we focus on the extraction of the complex events. In this paper, we propose an automatic event extraction system, which contains a model for complex events, by solving a classification problem with rich features. Our complex event detector performed better than the top system (in the shared task), and in overall performance, our system outperformed the top system.

1 Introduction

Relations among biomedical entities (i.e. proteins and genes) are important in understanding biomedical phenomena. Relations needed to be extracted automatically from within enormous 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) (Airola et al., 2008; Miwa et al., 2009) and disease-gene associations (DGAs) (Chun et al., 2006).

Binary relations are not sufficient for capturing biomedical phenomena deeply; thus, there is a growing need for more detailed and complex relations. For this purpose, two large corpora, BioInfer (Pyysalo et al., 2007) and GENIA (Kim et al., 2008), have been proposed. The BioNLP'09 Shared Task (Kim et al., 2009)[1] recently provided common and consistent task definitions, data sets, and evaluation. In the shared task, there are simple events and complex events. Whereas the simple events are binary relations, the complex events are complex relations, and the events consist of more than one binary relation. Bindings can represent the events including multiple proteins, and regulations can represent the events among events and proteins with representing their causality and direction. These complex events are much more informative than simple events, and these information are important in modeling biological systems, e.g. pathways.

In this paper, we propose a system with a focus on extracting complex events. Our system generally follows the hierarchy of the system by Björne et al. (2009), which was the top system in the shared task. By solving a new classification problem, our system constructs a model for extracting the complex events using rich features. As an evaluation result, with the model, our complex event detector is shown to perform better than the system in finding complex events, and in overall performance, our system is better than the system. We also display the results of the error analysis, which revealed several problems that needed to be resolved.

2 Related Works

In the BioNLP'09 Shared Task (Kim et al., 2009), there were three subtasks: finding core events (Task

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

Proceedings of the 3rd International Symposium on Languages in Biology and Medicine, Long Papers, pages 11-19, Jeju Island, South Korea, 8-10 November 2009.

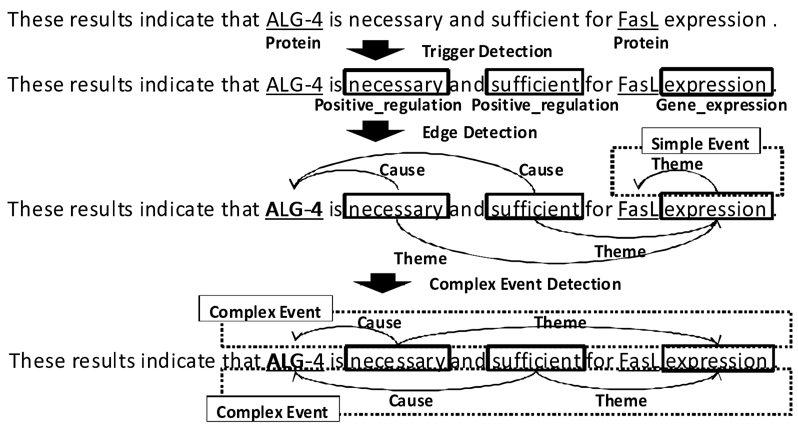

Figure 1: Flow of event extraction. Proteins are provided. A trigger detector first find triggers with their classes. An event edge detector then finds edges, some of which are simple events. A complex event detector then combines edges to construct complex events.

1), finding the secondary arguments (like location and sites) (Task 2), and recognizing speculation and negation (Task 3). Events consist of 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 including only a single primary theme protein, and a complex event is an event including multiple primary theme and cause arguments (proteins and events).

Our system targets the Task 1, and the goal of the task is to identify events with their types, textual triggers, and primary theme and cause arguments. The textual triggers are tokens which represent the events. We will explain two related systems participated on the task. Björne et al. (2009) introduced one system, which was the best performing system in the task. We will refer to the system as Turku System. The system extracted events with a hierarchical way, and our system follows the hierarchy of the system. The system first found triggers, then tried to extract the event edges, and ultimately combined the edges sharing the same triggers to extract complex events with a rule-based module. Sætre et al. (2009) introduced another system. The system treated complex events with a classifier, which is similar to our system. The system is also a hierarchical system, and the system found triggers first, but the system was different from Turku System in finding events. The system treated complex events as event instances and classified the complex events instead of treating them as the combinations of the edges, although they treated complex events without considering the dependencies among the events. The system performed well in finding binding events, because the system treated a complex event as an instance, and because the system used features known to be effective for the extraction of PPIs that are related to binding events (Miwa et al., 2009).

3 Event Extraction System

Our event extraction system basically follows the hierarchy of Turku System; trigger detection, edge detection, and complex event detection. Instead of applying rules for extracting complex events, our system solves a new classification problem for complex events, and construct a new model for extracting complex events. Figure 1 exemplifies the flow of the event extraction. All modules solve classification problems to construct models. The differences between our system and Turku System are in the features including the parsers and the additional features, the classification problems including the labels and the problem separation, and the machine learning based complex event detection.

In this section, we will explain about our system including these differences. We first introduce classifiers and their settings in Section 3.1. Then, we explain our preprocessing method in Section 3.2.

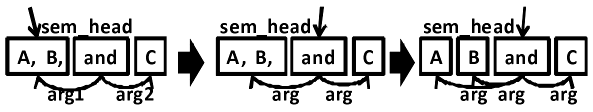

Figure 2: Modification of coordination structure in predicate argument structures produced by Enju. The semantic head (sem_head) of the phrase including a coordination is moved to coordinating phrase, argument names are converted to "arg" instead of "arg1" and "arg2" and the same level tokens are moved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

Lastly, we present three modules with problems to be solved: trigger detector in Section 3.3, event edge detector in Section 3.4, and complex event detector in Section 3.5.

3.1 Classifier

For the construction of our event extraction system, we solve the multi-class classification problems and multi-label classification problems. We use one-vs-rest support vector machines (SVMs) for solving these problems (Fan et al., 2008). We also fit a sigmoid function to the outputs by SVMs to calculate the confidences of the examples (Platt, 1999).

3.2 Preprocessing

All of the sentences in the data set are parsed using two parsers: a deep parser Enju 2.3.1 (Miyao et al., 2008) and a dependency parser GDep beta1 (Sagae and Tsujii, 2007). We use the predicate argument structures (PAS) by the deep parser, along with the dependency structures by 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 the flat structures, as shown in Figure 2, for the deep parser.

We constructed a trigger dictionary by extracting all of the triggers normalized by the parsers. This dictionary is used for detecting triggers in Table 1, and is also used for finding the triggers which 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 show equivalent protein mentions. If the gold events in the training data set contain the equivalent proteins, we create all equivalent events to remove inconsistent negative examples, and gain more positive examples. Considering the test data set setting, we remove the events that are sub-events of other events.

- Token has a big 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 (POS), lexical entry) of token in the outputs of parsers

Figure 3: Features for tokens.

- 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

Figure 4: Features for the shortest paths (SPs) between two entities (SP-Features).

3.3 Trigger Detection

For trigger detection, we need to find two types of triggers: one is the trigger of a trigger-protein relation (TP-T), and the other is the trigger of 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) mainly aims at extracting TP-T, while the other system (TT-T detector) aims at extracting 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 problem separation is different from that in Turku System, although the detail of their separa-

Type	Features
Token	Token features in Figure 3
Two words from candidate in parser output	Token features in Figure 3 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 Figure 4 between candidate and the closest proteins Lengths of paths between candidate and proteins

Table 1: The features of a Trigger Candidate for a TP-T (Trigger of Trigger-Protein relation) Detector

Type	Features
Token confidence	Confidences for all event classes by a TP-T detector
Shortest paths	SP-features in Figure 4 between candidate and the other closest candidates by TP-T detector for all event classes

Table 2: The additional Features of a Trigger Candidate for a TT-T (Trigger of Trigger-Trigger relation) Detector

tion approach is not open.

The trigger detectors target all words in the data sets. The detectors both try to classify all of the words into some event classes, including a negative event class (to extract event trigger words). As the labels of a word in the classifications, the event classes of gold triggers surrounding a word are used (e.g. Binding). 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 1. For the TP-T detector, we extract, in addition to the same features for the trigger detection in Turku System, the shortest path features between the event trigger candidate and the closest proteins (in the parser output) for including the information of supporting proteins. The shortest path features contain the features for event edge detection in Turku System, and their several additional n-grams and substructures as shown in Figure 4, which were used by Sætre et al. (2009). As features for the TT-T detector, we add two types of features to the features in the TP-T detector as shown in Table 2. One type is the confidences (for all event classes) of the event trigger candidate predicted by the TP-T detector; the other type is the shortest paths 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. By this edge detection, we can find simple events, which have only one argument.

We solve two separate classification problems: trigger-trigger edge detection and trigger-protein edge detection. We use all regulation events as one event class for the edge detection and the following complex event detection; 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 Section 3.3) are used, and we create positive and negative examples from the edges among the triggers and named entities. We extract features of an event edge candidate for edge detection as shown in Table 3. For the features, in addition to the features for the edge detection in Turku System, we use the confidences of terminal nodes by trigger detection. We also add the shortest path features between the argument trigger in trigger-trigger edge node and the closest proteins to add the information of supporting proteins to the features.

For trigger-trigger edges, the latter terminal node must be in another edge. The edges were checked recursively, and unacceptable edges were removed until all 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 Section 3.4. To find the

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

Table 3: The features of an Event Edge Candidate for an Edge Detector

Type	Features
Each event edge	Edge features in Table 3
All pairs among arguments	Edge features in Table 3 except shortest paths between an argument trigger and the closest proteins
All edges including event trigger outside of events	Edge features in Table 3
All pairs between argument proteins and their closest proteins in binding	Edge features in Table 3

Table 4: The features of a Complex Event Candidate for a Complex Event Detector

appropriate combinations, we construct a complex event detection system. Turku System combined 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 approach can automatically construct models from the training data, and can find combinations missed by the 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, Regulations) for four complex event classes. We treat all regulations as one event class (like the edge detection). For each problem, the event class and the connected terminal node types (event or protein) are used as the labels of complex events (e.g. Regulations: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 in consideration of the edges inside and outside of the events; the arguments of inside edges should interact each other, and the arguments of outside edges should not interact with the arguments of inner edges. We extract features of a complex event candidate for complex event detection as shown in Table 4, using the feature extractor for the edge detection in Section 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 3 for several edges, and the features 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 our system by using the evaluation script[2] for the development data set and the evaluation system[3] for the test data set. The script and the system are provided by the shared task organizers. Errors were also analyzed on the development data set.

Liblinear-java[4] (Fan et al., 2008) was used as the one-vs-rest SVMs explained in Section 3.1. Our system contains many classification problems, and tun-

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

[3]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/eval-test.shtml

[4]http://www.bwaldvogel.de/liblinear-java/

	Development Data Set			Test Data Set			Turku (Test Data Set)		
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 5: Approximate Span Matching/Approximate Recursive Matching on Development Data Set, Test Data Set, and Test Data Set with Turku System.

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

Table 6: Comparison of our result with the result by Turku System on Test Data Set in F-score. F-scores in parentheses show the result of the Event Decomposition/Approximate Span Matching/Approximate Recursive Matching.

ing thresholds for each problem takes up much computational cost. We used the same settings for all of the problems. The one-vs-rest SVMs need to solve many unbalanced classification problems. To ease the problem, we balanced the positive and negative examples by putting more weight on the negative examples. To have a selection of as many confident examples as possible, we selected examples with the confidences more than 0.5, in addition to the examples with the most confident labels. The C-values were set to 1.0. Please note that this setting is different from Turku System, which tuned the C-values and thresholds for all their detectors. This parameter tuning is left as future work, and this will be discussed in Section 5.

4.2 Performance

Table 5 shows the performance of our system on the development data set produced by the evaluation script, the performance on the test data set produced by using the evaluation system, and the performance of Turku System on the test data set. Table 6 summarizes the comparison of our result with the result by Turku System.

Our system is comparable to Turku System in finding simple events and regulations, and our system performed much better than the system in finding binding events. In overall performance, our system outperformed the system in the shared task.

In the complex event detection, our classification approach is better than the rules in Turku System as indicated in Table 6; the loss in event composition by our approach is less than that by the rules.

For binding events, the F-score is much better than other systems that were submitted to the 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 do not consider the “equiv” annotation as shown in Section 3.2. The script can output lower score than the evaluation system. The event decomposition results implies that the additional features is useful for finding binding events (Sætre et al., 2009). The loss in event composition shows that our complex event detector is useful for finding complex binding events. The correct combinations of arguments are selected by using our rich feature vector.

For regulation events, the F-score on the development data set decreased about 2% when we set the threshold for the complex events to zero. This setting without the threshold is the same as the rules in Turku System. The result of this setting indicates

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

Table 7: Error Classification among 100 Missing False Negatives on the Development Data Set. 61 triggers were missing, and 39 events were missing.

that, in finding regulation events, our system could not produce the comparable results with the system in the level of the edge detection. This weakness in finding edges is also seen in the event decomposition results in Table 6. Our system could perform comparable to Turku System in the level of the event detection, and the complex event detector may also improve the system. However, the complex event detector did not drastically affect finding regulation events. This is the case because most of the causal events could not be found by the edge detector, and because 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 we need to resolve the hierarchy of events for finding causes.

4.3 Error Analysis

For the further improvement and practical use of the event extraction system, an improvement in the recall is necessary. Table 7 summarizes the analysis of 100 missing false negatives.

61 triggers were missing from among 100 errors. 10 errors include coreferences/exemplification problems between the trigger and the theme or cause; and the distance from the trigger to protein was far in these cases. The problems include pronoun, 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 no training instances. Some triggers did not appear in the training data set. We may be able to dig some of the triggers by using other resources, like variations of terms, to find such instances. Seven errors were caused by the 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 answered them as different types. These errors also include a few difficult cases, which can be ambiguous for the annotators (like the ambiguity between regulation and positive/negative regulation). Three errors include self interactions, like "transfection," which can be regulation events without other triggers or proteins in the shared task data set. Three errors are caused by parse problems, with PP-attachment problems. Three errors contain inference problems. The other 19 errors were mostly caused by the threshold.

In cases where events are missing, 39 missing errors were found. Seven errors were caused by coreferences. 10 regulation events were not found by 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 by our system. Two regulation events, including the causes, were missed because our system had some difficulty in resolving the event hierarchy explained in Section 4.2.

5 Discussion

A drawback of our system is in the dependencies among the parameters to be tuned. A parameter tuning considering for the event edge detector and the trigger detector was done in Turku System, but the parameter tuning for our systems is more complex. In our system, the complex event detector depends on the event edge detector, and the event edge detector depends on the trigger detector. In addition to these dependencies, since some regulation events depend on other events, it is difficult to find optimal parameters for regulations. Regulations are different from other events; they can have the other events as arguments, and can have causes. For treating regulations more appropriately, one possible approach is to find the regulation events after finding other events and simple regulation events. Thresholds are also the parameters in our system. The cut uncertain triggers and edges include many events to be found, and some of them can be found by the complex event detection. To improve the recall, one possible approach is to use confidence effectively without thresholds, although how to use confidence appropriately in machine learning is not trivial.

For the event extraction, the analysis in Section 4.3 illustrated many types of NLP problems. From the analysis, 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 part of the missing trigger problems. Coreferences increase the distance between a trigger and the arguments. Coreferences also cause errors in finding PPIs (Miwa et al., 2009), and we need to focus on resolving them for finding biomedical relations.

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

One problem involves the selection of the target proteins and events (Kim et al., 2009). The shared task data sets were made from the GENIA corpus (Kim et al., 2008). By the selection, some meaningless or incomplete events were left in the shared task data sets. Some events were missing; the events should be between regulation events and the causal events for representing the hierarchy of events. This problem may be reduced by using the GENIA corpus instead of the shared task data sets, or by removing these events from the shared task data set, by using some 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 (Kim et al., 2008). The evaluation scripts ease 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 what event classes involved the proteins. We can try to evaluate the performance of our system considering the event classes without the triggers for a more general analysis of our event extraction system. This evaluation can then make clear the issues that need more focus. Considering this evaluation policy, we can think of other approaches to find events; finding event types including proteins first instead of triggers is a possible approach.

For the shared task evaluation system, we observed that the recall tended to be lower in the test data set than in the development data set. To avoid tuning the recall for the system and compare the event extraction systems from different points of view, we need more evaluation criteria than just the F-score. We can use AUC (area under the ROC [receiver operating characteristic curve) used in the evaluations of recent PPI extraction systems (Airola et al., 2008; Miwa et al., 2009).

6 Conclusion

In this paper, we proposed an event extraction system which mainly focuses on extracting complex events. Our complex event detector performed better than the rule-based detector in the top system, and the proposed system performed better than the other systems in the BioNLP'09 Shared Task data set. We also analyzed false negatives on the development data set, and we showed that the missing triggers caused 71% of errors, and that coreferences were a major problem. Our system will be integrated into U-compare (Kano et al., 2009).

Based on the error analysis, we need to integrate a coreference resolution system. The coreferences here are the combination of many problems, including the pronoun, anaphora, and apposition. We will continue to analyze errors further, to determine the problems that need to be focused on to improve the general event extraction system; further, we need to determine the problems caused by the shared task setting. 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

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross corpus learning. *BMC Bioinformatics*.

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'09 Shared Task on Event Extraction*, pages 10–18.

Hong-Woo Chun, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, and Jun'ichi Tsujii. 2006. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. The Pacific Symposium on Biocomputing (PSB), pages 4–15.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Yoshinobu Kano, William Baumgartner, Luke McCrohon, Sophia Ananiadou, Kevin Cohen, Larry Hunter, and Jun'ichi Tsujii. 2009. U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, 25(15):1997–1998. in press.

Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. 2008. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9:10.

Jin-Dong Kim, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. 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*, pages 121–130, August.

Yusuke Miyao, Rune Sætre, Kenji Sagae, Takuya Matsuzaki, and Jun'ichi Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08:HLT)*.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Rune Sætre, Makoto Miwa, Kazuhiro Yoshida, and Jun'ichi Tsujii. 2009. From protein-protein interaction to molecular event extraction. In *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pages 103–106.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007*.