Parsing Dependency Paths to Identify Event-Argument Relations

Seung-Cheol Baek
CS Dept., KAIST
291 Daehak-ro, Yuseong-gu, Daejeon,
305-701, Republic of Korea
scbaek@nlp.kaist.ac.kr

Jong C. Park
CS Dept., KAIST
291 Daehak-ro, Yuseong-gu, Daejeon,
305-701, Republic of Korea
park@nlp.kaist.ac.kr

Abstract

Mentions of event-argument relations, in particular dependency paths between event-referring words and argument-referring words, can be decomposed into meaningful components arranged in a regular way, such as those indicating the type of relations and the others allowing relations with distant arguments (e.g., coordinate conjunction). We argue that the knowledge about arrangements of such components may provide an opportunity for making event extraction systems more robust to training sets, since unseen patterns would be derived by combining seen components. However, current state-of-the-art machine learning-based approaches to event extraction tasks take the notion of components at a shallow level by using n-grams of paths. In this paper, we propose two methods called pseudo-count and Bayesian methods to semi-automatically learn PCFGs by analyzing paths into components from the BioNLP shared task training corpus. Each lexical item in the learned PCFGs appears in 2.6 distinct paths on average between event-referring words and argument-referring words, suggesting that they contain recurring components. We also propose a grounded way of encoding multiple parse trees for a single dependency path into feature vectors in linear classification models. We show that our approach can improve the performance of identifying event-argument relations in a statistically significant manner. [1]

1 Introduction

Event extraction tasks can be viewed as identifying event-argument relations between tokens by mapping events onto tokens, to be called henceforth *triggers*, even though events may have other events as arguments in contrast to average relation extraction tasks, leading to interdependencies between events. On looking into mentions of event-argument relations, in particular the shortest dependency path between triggers and arguments, one may find that they can be decomposed into intuitively meaningful components arranged in a regular way, such as *core components* indicating the type of relations and *subordinate components* making it possible for events to take arguments further away from triggers (e.g., coordinate conjunction). We anticipate that the knowledge about arrangements of components provides an invaluable opportunity for making event extraction systems more robust to the choice of training sets, for example by assembling seen components into unseen patterns. Towards this goal, we propose in this paper a way of automatically learning and exploiting internal structures of dependency paths for a robust extraction of biological events from the biological literature with the corpora provided by a series of BioNLP shared tasks (Kim et al., 2009 and Kim et al., 2011).

For example, the following sentence has annotated positive regulation events, including the induction of IP-10 by IFN, in the training corpus.

(1) IL-10 preincubation resulted in the inhibition of gene expression for several IFN-induced genes, such as IP-10, ISG54, and intercellular adhesion molecule-1. (PMID: 10029571)

From this sentence, we may formulate the pattern "X-induced genes, such as Y" with slots X and Y to detect the THEMEs of positive regulation events based on the underlined expression. This pattern can also be decomposed into a core component "X-induced Y" and a subordinate component "genes such as Y". These two components have different roles. That is, the core component

[1] All the datasets and codes used in this study are available at http://www.biopathway.org/ijcnlp2013

International Joint Conference on Natural Language Processing, pages 699–705,
Nagoya, Japan, 14-18 October 2013.

alone can be used to detect the THEMEs of positive regulation events (e.g., "IFN-induced IP-10"), but the subordinate component alone cannot. Core components may not appear together in a pattern, but subordinate components may (e.g., there are two other involved subordinate components "genes such as Y" and "[PROTEIN] and Y", where "[PROTEIN]" would be replaced with any protein or gene name, in "IFN-induced genes, such as IP-10, ISG54 and intercellular adhesion molecule-1"). From this observation it is possible to come up with an unseen pattern "X-induced Y and Z".

However, current state-of-the-art machine learning-based approaches exploit the notion of components of patterns only at a shallow level using n-grams encoding partial structures of dependency graphs (including unigrams used in bag-of-words models), not to mention the notion of regularity in arrangements of components (e.g., Björne et al., 2009; Miwa et al., 2010; Riedel et al., 2011). Therefore, their approaches would be biased towards dependency paths that contain a number of components even overlapping with one another, even though such paths may have undesired meanings due to the arrangements of components.

In this paper, we propose two methods (called *pseudo-count* and *Bayesian* methods) to semi-automatically learn three types of probabilistic context-free grammars (PCFGs) that assume different internal structures of paths, with the help of which dependency paths will be analyzed into components. All the learned PCFGs contain lexical items covering an average of about 2.6 distinct paths between triggers and arguments in the training corpus, suggesting that the methods successfully identified recurring components. To exploit multiple parse trees derived from a single path, we also propose a linear classification model whose output score approximates the difference between the log probabilities of the path being derived from positive and negative relations. We find that the use of PCFGs learned by our pseudo-count method improves the performance of classifiers in a statistically significant manner, compared to a baseline classifier with n-grams encoding partial structures of paths.

2 Related Work

The literature on information extraction (IE) contains a number of studies in which dependency paths are found to play a significant role (Johansson and Nugues, 2008; Miwa et al., 2010b; Qiu et al., 2011). Likewise, the biological event extraction research, a branch of information extraction, stresses the importance of the role of dependency paths in identifying event-argument relations due to the resemblance of event-argument relations to dependency relations (Björne et al., 2008). For this reason, most of the event extraction studies have to use dependency path features, such as n-grams (n=1~4) of dependencies and words, the length of dependency paths and so on, in identifying event-argument relations (Björne et al., 2009 and Miwa et al., 2010b).

It is thus not surprising that while there are many studies on dependency paths in the IE literature, most of them focus on identifying the type of dependency graph representations that is most suitable to their problem (cf. Johansson and Nugues, 2008, Miwa et al., 2010a), with few exceptions including Kilicoglu and Berger (2009) and Joshi and Penstein-Rosé (2009). In particular, Kilicoglu and Bergler (2009) manually constructed a total of 27 dependency path patterns by examining dependency paths between triggers and arguments. Joshi and Penstein-Rosé (2009) first generated sequences of triples of dependency relations and the baseform/POS of their tokens and then generalized the sequences by concealing one of the elements of the triples. They find that the use of such generalized sequences improves the performance of the task of identifying opinions from product reviews. However, there are no studies on automatically learning and using the internal structure of the dependency paths that express semantic relations between tokens, as addressed in this paper, to the best of our knowledge.

3 Problem Setting

Our proposal is tested on the event extraction task as defined in the 2009 BioNLP shared task 1 (Kim et al., 2009), which was later renamed as GENIA Event Task 1 and extended to cover full papers in the 2011 Bio-NLP shared task (Kim et al., 2011). Their task is to extract structured information on events from sentences in the biological literature, including their event type and participants encoded with a controlled vocabulary that has nine event types and two role types ("THEME" and "CAUSE"). This task can be considered to consist of two sub-tasks, one of identifying triggers and another of identifying event-argument relations. In this study, we focus on the latter and use the gold-standard annota-

tions of triggers in the training and development corpora (including full papers) to generate dependency paths for training and testing.

In order to identify event-argument relations, we use twelve binary classifiers for all the possible combinations of event and role types. One may argue that multi-class classifiers are more suitable for this setting than binary classifiers, but there is no conclusive evidence for their advantage (cf. Baek and Park, 2012). Note also that our present focus is on assessing the benefit from the use of the knowledge about internal structures of dependency paths and not on assessing the whole event extraction systems.

4 Method

4.1 Preparation of Training Sequences

The shortest dependency paths between triggers and argument candidate words (e.g., "-induced" and "IP-10" in (1)) over basic Stanford dependency graphs[2] (de Marneffe et al., 2006) are first computed, from which the three types of sequences are extracted in turn: *token sequences*, or a sequence of the surface forms of the visited tokens (e.g., "induced gene as IP-10"), *dependency sequences*, or a sequence of the visited dependencies (or more precisely, their type and direction; e.g., "-amod +prep +pobj"), and *combined sequences*, or a sequence of the visited tokens and dependencies (e.g., "induced -amod genes +prep as +pobj IP-10").

Training sequences are derived from the extracted sequences by preprocessing them as follows. First, the last tokens of sequences, namely arguments, are dropped, because of the observation that this makes it easy to convert the components of patterns into sequences and their subsequences in a systematic way. For example, the two components "-induced Y" and "genes, such as Y" of the pattern "-induced genes, such as Y" can be converted into the sequences "-induced -amod" and "genes +prep as +pobj", which are combined into a sequence corresponding to the pattern, namely, "induced -amod genes +prep as +pobj". Second, protein names are replaced with a special token "[PROTEIN]" to help learn generalized patterns, since there are a considerable amount of different types of proteins. Third, the first occurrence of each word in the training corpus is replaced with a special token "[UNKNOWN]" to simulate encounters with unknown words in the test corpus during learning. Its downside is that all the tokens in the first training sequence are replaced with "[UNKNOWN]".

Note that it is a natural extension of our work to additionally generate other types of sequences, for example by replacing the surface forms of tokens with their other attributes (e.g., POSs and surface forms concatenated with POSs) in sequences mentioned above and by dropping functional tokens (e.g., prepositions) within token sequences, even though we do not consider them here.

4.2 PCFG Induction

A PCFG consists of production rules (of the form $A \rightarrow x$), each indicating that a nonterminal symbol A (a *parent symbol*) is replaced with a sequence x of symbols (*child symbols*) with a predefined probability. Our PCFGs have two types of production rules, those that produce a sequence of nonterminal symbols (*non-lexical production rules*) and the others that produce a lexical item (*lexical production rules*). In our PCFGs, non-lexical production rules are crafted manually and lexical production rules are learned. The probability of each rule is determined by maximum-likelihood estimation (MLE), which divides the total number of the occurrences of the rule in training parse trees by the total number of the occurrences of its parent symbol in training parse trees.

We build two sets of non-lexical rules, one generating positive sequences and another generating negative sequences, together with the following two non-lexical rules, where "S" stands for the start symbol and "Positive" and "Negative" symbols are the ones to be expanded into positive and negative sequences, respectively.

(2) S→Positive
(3) S→Negative

We come up with the following three types of non-lexical rules for positive sequences, where the underlined symbols are *lexical symbols*, or the ones to be expanded into single lexical items, and asterisks indicate that the marked symbols may occur zero or more times in a row.

(4) Unigram Rules
Positive→<u>Component</u> <u>Component</u>*
(5) Uni-directionally Growing Rules
Positive→Core <u>Component</u>*

[2] Since arguments and triggers may be hyphenated, we preprocess dependency graphs, so that hyphenated words are separated into their component words.

(6) Bi-drectionally Growing Rules
Positive→Component* Core Component*

These rules assume that sequences consist of components that may appear independently of one another (*independence constraint*), but also that they cannot overlap with one another (*non-overlapping constraint*). The second and third types of rules assume that sequences should have core components as indicated by the "Core" symbols. The independence constraint may not capture the nature of dependency paths, but makes it cheaper to learn lexical rules. We leave the question about the effect of the independence constraint open for future research. The uni-directionally growing rules are most consistent with our observation that triggers and their dependencies play a significant role in determining the type of event-argument relations.

Since lexical items are allowed to span across more than one element in positive sequences but are not annotated on training sequences, we need to make a guess at parse trees for each sequence to count the occurrences of rules. To address this problem, we propose two methods. One is called a *pseudo-count method* that assigns all possible parse trees for each training sequence an equal probability (i.e., one divided by the number of all possible parse trees) of the sequence being generated from them, and accumulates the assumed probability (i.e., pseudo-count) of parse trees containing each rule.

Another is called a *Bayesian method* that converts our non-lexical rules into an adaptor grammar, or a description of non-parametric Bayesian models with Chinese Restaurant Processes (CRP) and Pitman-Yor Processes (PYP) (Johnson et al., 2007), by adding production rules, to be called *lexical item production rules*, that replace lexical symbols with a sequence of terminal symbols, such as tokens and dependencies (e.g., "Tokens→Token Token*"), and by labeling lexical symbols as an *adaptor symbol*, whose expansion to terminal symbols is collected during learning. One advantage of this method is to penalize lengthy lexical items, and as a result, to facilitate analyzing sequences into more than one lexical item, since producing a lengthy lexical item requires the use of many lexical item production rules with a probability below one. In practice, we use the adaptor grammar inference program (Johnson et al., 2007), which samples analyses of input sequences (i.e., sequences of dependency types). We assume that all production rules in our adaptor grammars have the same probability. We ran two thousand iterations of sampling analyses, but ignored samples during the first half, as these may not be significantly different from randomly assigned initial analyses. Afterwards, we counted the occurrences of lexical items and rules. As a result, 1,000 samples are taken for each sequence.

Since negative training sequences can convey a variety of semantic types, it is unlikely that a training corpus contains all possible negative training sequences covering such semantic types, suggesting the risk of over-fitting of learned PCFGs to negative training sequences (cf. Li et al., 2010). To avoid it, we use a simple grammar, which is expected to be able to learn from a relatively small amount of training instances, as shown below, where "NComponent" symbols produce single token and dependency types. In contrast to the positive sequences, it is straightforward to construct parse trees for negative sequences and to count production rules, since all negative sequences have only one possible parse tree.

(7) Negative→NComponent NComponent*

Finally, we filter out infrequent and lengthy lexical items, which may have the same form as the sequences from which they are learned, to prevent models from memorizing training sequences as they are (e.g., "induced -amod genes +prep as +pobj"), that is, assigning a high weight to them and to teach instead models ways of analyzing positive sequences into relatively small lexical items (e.g., "induced –amod" and "genes +prep as +pobj"). For each lexical symbol, we remove the least probable lexical items whose occurrences form a predefined percentage[3] of the occurrences of the lexical symbol. Note that it is apparently a more reasonable option to learn PCFGs and linear classification models on two different disjoint subsets of randomly selected sequences. We leave this option for future work.

4.3 Linear Classification Model

Using the CKY algorithm with beam search, we generate the most probable *k* parse trees for three types of sequences extracted from a dependency path with the help of the learned PCFGs, each of which explicitly has a favorite label. One way to

[3] The predefined percentage is 1% if the ratio of the number of distinct sequences to the number of sequences is below a third, 5% if the ratio is between a third and two third, and 10% otherwise.

combine their opinions is to let respective classifiers *S* for the types of sequences vote for their favorite label $z_s(x)$ ($+1$ or -1) of a path *x* and to count their vote with a different weight proportionate to their reported confidence $w_s(x)$ and their credibility c_s, as follows:

$$y = \sum_S c_s z_s(x) w_s(x) = \sum_S c_s y_s(x)$$

If the output score y is positive, our classifier makes a final decision of labeling *x* as being positive. The term $z_s(x)w_s(x)$ can be regarded as the output score $y_s(x)$ given by a classifier *S*.

We define $y_s(x)$ as follows, where the capital letters stand for random variables:

$$\mathrm{y_s}(x) = \log\left(\frac{\mathrm{P}(Z=+1, X=x)}{\mathrm{P}(Z=-1, X=x)}\right)$$

The log probability $\log(\mathrm{P}(Z,X))$ is written in terms of the probability $\mathrm{P}(Z,T)$ of our PCFGs generating T_z parse trees supporting a value *z* of Z:

$$\log(\mathrm{P}(z,x)) = \log\left(T_z \times \overline{\mathrm{P}(z,T_x)}\right)$$

where $\overline{\mathrm{P}(z,T_x)}$ is the average of the probability of parse trees generating *x* and supporting *z*. Using Jensen's inequality, it is easy to show that its lower bound $l(z,x)$ is:

$$\log\left(\mathrm{P}(z,x)\right) \geq l(z,x) = \overline{\log \mathrm{P}(z,T_x)} + \log T_z$$

where the first term is the average of the log probability of parse trees under consideration. One thing to note is that the equality always holds for $\mathrm{P}(Z=-1, X=x)$, since our PCFGs for negatively labeled sequences produce at most one parse tree for each sequence. For this reason, the lower bound $y_s^{low}(x)$ of $y_s(x)$ is:

$$y_s^{low}(x) = \sum_z \left(\sum_{t \to x|z} \frac{z \log \mathrm{P}(T=t)}{T_z} + z \log T_z \right)$$

Instead of $y_s(x)$, we use $y_s^{low}(x)$ at risk of the deterioration of the performance of the resulting model, since it is apparently easier to handle than $y_s(x)$.

In the worst case, the difference between $\log \mathrm{P}(Z,X)$ and $l(z,x)$ can be:

$$\left|\log\left(\mathrm{P}(z,x)\right) - l(z,x)\right| \leq \log \frac{\overline{\mathrm{P}(z,T_x)}}{\mathrm{P}_{min}(z,T_x)}$$

where the denominator is the least probability of parse trees under consideration. It indicates that with a wide beam width the estimated value of $y_s^{low}(x)$ may be significantly lower than the true value of $y_s(x)$, while with a narrow beam width the estimated value of $y_s^{low}(x)$ is likely to be similar to the estimated value of $y_s(x)$, which may be significantly higher than its true value. Thus the success of the use of $y_s^{low}(x)$ is dependent on the beam width.

Expanding the log probability $\log\mathrm{P}(T=t)$, $y_s^{low}(x)$ is rewritten:

$$\sum_r \sum_z z \log(p_r) \left(\sum_{t \to X|z} \frac{count(r \text{ in } t)}{T_z} \right) + \left(\sum_z z \log T_z \right)$$

where p_r is the probability of rule *r*, which is given by PCFGs, and $count(r \text{ in } t)$ is the number of the occurrences of rule *r* in parse tree *t*. Introducing coefficients w_r, w_1 and w_0 into the equation, $y_s^{low}(x)$ can be generalized to a linear model as shown below.

$$\sum_r w_r \left(\sum_{t \to X|z} \frac{count(r \text{ in } t)}{T_z} \right) + w_1 \left(\sum_z z \log T_z \right) + w_0$$

Being their linear combination of the linear models $y_s^{low}(x)$, the output score *y* is also a linear model. In this paper, we train our linear classifiers using LIBLINEAR (Fan et al., 2008) [4].

Finally, we note that as in the re-ranking parsers (e.g., Charniak and Johnson, 2005), it is possible to use *global features*, or features not allowed in the CKY algorithm, to calculate the log probability $\log\mathrm{P}(T=t)$. In this paper, we leave the effect of the use of such global features for future research.

5 Experiments

We generated labeled training dependency paths for each event-argument relation type from the BioNLP training corpus with the help of the Charniak-Johnson parser (Charniak and Johnson, 2005) with a self-trained biomedical parsing model (McClosky and Charniak, 2008). There are 7,009 positive paths and 10,603 negative paths. The ratio of the number of positive paths

[4] Our linear classifiers are trained using the L2 regularized logistic regression solver with cost constants that are chosen among 0.01, 0.1, 1 and 100 with the help of five-fold cross validation.

to the number of negative paths is 0.66. We found that a majority of the relation types would have a balanced set of training instances, except for a few relation types with the imbalance between positive and negative instances. One way of correcting the imbalance is to give more weight to positive instances, but we leave out the imbalance in this experiment.

We extracted three types of sequences from them. We found that most distinct negative sequences appear once in the training corpus as shown in Table 1, where the bracketed figures are the ratios of the number of distinct sequences to the number of sequences, justifying the use of a simple grammar for negative sequences.

Sequence	Positive	Negative	Total
Combined	3,703 (1.89)	9,781 (1.08)	13,484 (1.31)
Token	3,366 (2.08)	9,270 (1.14)	12,636 (1.39)
Dependency	1,816 (3.86)	7,419 (1.43)	9,235 (1.91)

Table 1. Distinct Training Sequences

We use the pseudo-count and Bayesian methods to learn grammars. The learned PCFGs contain the mentioned example lexical items, "-induced -amod", "genes +prep as +pobj" and "[PROTEIN] +conj". They contain a number of intuitively correct core and subordinate components. The learned subordinate components include "genes +prep like +pobj", "[PROTEIN] +abbrev" and "[PROTEIN] +appos".

With three different beam widths, we parse sequences to generate feature vectors for our linear classification models and evaluate the resulting models in terms of accuracy, as shown below.

Grammar	Beam Width		
	k=1	k=10	k=100
Pseudo-Count			
Unigram	**86.43%**	85.97%	86.07%
Uni-direct	86.94%	**87.05%**	87.03%
Bi-direct	**86.48%**	86.43%	86.25%
Bayesian			
Unigram	82.72%	**83.39%**	83.27%
Uni-direct	82.95%	**83.70%**	82.88%
Bi-direct	82.70%	**83.45%**	83.26%

Table 2. Accuracy of Our Classifiers

For each grammar, the best reported accuracy is set in bold. With PCFGs learned by the pseudo-count method, the use of multiple parse trees does not affect or even decrease the performance of classifiers. One possible explanation is that the wider the beam is the more erroneous parse trees are likely to affect the final decision of classifiers. In contrast, the classifiers with PCFGs learned by the Bayesian model would slightly benefit from the use of multiple parse trees, even though their performance also drops when using the widest beam. To explain that we get only a slight benefit from a wide beam width, we looked at feature vectors, noticing that many positive training sequences have only a small number of possible parse trees. We also observed that as expected, classifiers with the uni-directionally growing PCFGs outperform the other classifiers, with one exception of classifiers with the widest beam and the ones learned by the Bayesian method.

To compare with our classifiers, we implement *linear baseline classifiers* that use as features all n-grams (n=1~4) of token, dependency and combined sequences extracted from the training instances. They first replace unknown words in an input sequence with a special token "[UNKNOWN]" and count the occurrence of n-grams in the sequence. Like our classifiers, they are also trained by LIBLINEAR (Fan et al., 2008).

The accuracy of the baseline classifiers is 85.76%, which is lower than that of the pseudo-count classifiers with any beam width in use, but higher than that of the Bayesian classifiers with any used beam width. The superiority of the pseudo-count classifiers with any beam width over the baseline classifiers is statistically significant at the 10% significance level in terms of their accuracy (p-value=5.6~8.4%), according to the one-sided paired Student's *t*-test with the accuracy of classifiers for each relation type.

6 Conclusion

In this paper we proposed a way of exploiting internal structures of dependency paths for the extraction of biological events from the biological literature with the BioNLP shared task corpora. We proposed pseudo-count and Bayesian methods to learn three types of PCFGs that assume different internal structures of paths from dependency paths. To use multiple parse trees for a single path, we also developed a linear classification model whose output score approximates the difference between the log probabilities of the path being derived from positive and negative relations. Finally, we have shown that our approach can improve the performance of identifying event-argument relation in a statistically significant manner.

Acknowledgments

This work was supported by the National Research Foundation (NRF) of Korea funded by the Ministry of Education, Science and Technology (MEST) (No. 20110029447). We are also grateful to the anonymous reviewers who helped improve the clarity of the paper. All remaining errors are of course ours.

References

Baek, S. C. and Park, J. C. (2012, September) Use of Clue Word Annotations as the Silver-standard in Training Models for Biological Event Extraction. In *Proceedings of the SMBM 2012* (pp. 34-41).

Björne, J., Heimonen, J., Ginter, F., Airola, A., Pahikkala, T., & Salakoski, T. (2009, June). Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task* (pp. 10-18). Association for Computational Linguistics.

Björne, J., Pyysalo, S., Ginter, F., & Salakoski, T. (2008, September). How complex are complex protein-protein interactions. In *Proceedings of the SMBM 2008* (pp. 125-128).

Charniak, E., & Johnson, M. (2005, June). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the ACL 2005* (pp. 173-180). Association for Computational Linguistics.

de Marneffe, M. C., MacCartney, B., and Manning, C. D. (2006, May). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC* (Vol. 6, pp. 449-454).

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., & Lin, C. J. (2008). LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, *9*, 1871-1874.

Johansson, R., and Nugues, P. (2008, August). The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (pp. 393-400). Association for Computational Linguistics.

Johnson, M., Griffiths, T. L., & Goldwater, S. (2007). Adaptor grammars: A framework for specifying compositional nonparametric Bayesian models.*Advances in neural information processing systems*, *19*, 641.

Joshi, M., and Penstein-Rosé, C. 2009. Generalizing dependency features for opinion mining. In *Proceedings of the ACL-IJCNLP 2009 Conference* (pp. 313-316). Association for Computational Linguistics.

Kilicoglu, H. and Bergler, S. 2009. Syntactic Dependency Based Heuristics for Biological Event Extraction. In *Proceedings of the BioNLP Shared Task 2009 Workshop* (pp. 119-127).

Kim, J. D., Ohta, T., Pyysalo, S., Kano, Y., and Tsujii, J. I. (2009, June). Overview of BioNLP'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task* (pp. 1-9).

Kim, J. D., Wang, Y., Takagi, T., and Yonezawa, A. (2011, June). Overview of GENIA event task in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop* (pp. 7-15).

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., & Steedman, M. (2010, October). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 1223-1233). Association for Computational Linguistics.

Li, X. L., Liu, B., & Ng, S. K. (2010, October). Negative training data can be harmful to text classification. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 218-228). Association for Computational Linguistics.

McClosky, D., and Charniak, E. (2008). Self-training for biomedical parsing. In *Proceedings of the ACL 2008* (pp. 101-104). Association for Computational Linguistics.

Miwa, M., Pyysalo, S., Hara, T., and Tsujii, J. I. (2010a). A comparative study of syntactic parsers for event extraction. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing* (pp. 37-45). Association for Computational Linguistics.

Miwa, M., Sætre, R., Kim, J. D., & Tsujii, J. I. (2010b). Event extraction with complex event classification using rich features. *Journal of bioinformatics and computational biology*, *8*(01), 131-146.

Qiu, G., Liu, B., Bu, J., and Chen, C. (2011). Opinion word expansion and target extraction through double propagation. *Computational linguistics*, *37*(1), 9-27.

Riedel, S., & McCallum, A. (2011, June). Robust biomedical event extraction with dual decomposition and minimal domain adaptation. In *Proceedings of the BioNLP Shared Task 2011 Workshop* (pp. 46-50). Association for Computational Linguistics.