Combining Multiple Layers of Syntactic Information for Protein-Protein Interaction Extraction

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

Abstract

Protein-protein interaction extraction is a challenging information extraction task in the BioNLP field. Several kernels focusing on a part of syntactic information have been proposed for the task. In this paper, we propose a method to combine multiple layers of syntactic information by using a combination of multiple kernels based on several different parsers. We evaluated the method using support vector machine and achieved an F-score of 62.0% on the AIMed corpus. Further, we analyzed the performance with or without including self-interaction pairs, and found that there is a danger of confusing classifiers and decreasing the performance when treating self-interaction pairs together with real pairs.

1 Introduction

With the growing number of research papers, researchers have difficulty finding the papers that they need. Biomedical relationships in papers help biomedical researchers to find specific papers. Protein-protein interactions (PPI) are one of relations, and automatic extraction methods can help to extract biomedical relationships based on PPI. PPI is also important in biological processes, and finding them automatically can help construct PPI databases that are usually manually constructed, like BIND (Mathivanan et al., 2006). To achieve this, researchers in the BioNLP field have been examining automatic extraction of PPI from research papers.

One major approach for this task is finding a criteria to judge whether a sentence which contains a pair of proteins actually implies interaction of the pair or not. Detection of PPI was initially tackled by using simple methods based on co-occurrences (Blaschke et al., 1999), while more sophisticated NLP techniques have been used later (Bunescu et al., 2005). For example, NLP tools were used to lemmatize surface words and tag them by their parts of speech (POS). Dependency relations in sentences can also be revealed by syntactic parsers. While NLP techniques make this information explicit, appropriate techniques should be applied to use the information collectively for judging the relevance of a sentence for PPI. For this purpose, several kernels have been proposed, including subsequence kernels (Bunescu and Mooney, 2005b), tree kernels (Moschitti, 2006; Sætre et al., 2007), shortest path kernels (Bunescu and Mooney, 2005a), and graph kernels (Airola et al., 2008). Each kernel utilizes a portion of the structures to calculate useful similarity. The kernel cannot retrieve the other important information that may be retrieved by other kernels.

In this paper, we propose a way of combining kernels based on several syntactic parsers for PPI extraction. In order to retrieve the widest range of important information in a given sentence, it is important to extract as much information as possible from the sentence and its parse graphs. Using a support vector machine (SVM) with much useful information from the combination, we achieved an F-score of 62.0% on the AIMed corpus.

We also analyzed the performance changes with or without self-interaction pairs (self-pairs). From this analysis, we found that the self-pairs can have confuse the classifiers and decrease the performances. To avoid this and make the performance better, predicting the self-pairs and the binary-interaction pairs (binary-pairs) indepen-

XPG$_{p1}$ protein interacts with multiple subunits of **TFIIH**$_{prot}$ and with **CSB**$_{p2}$ protein.

Figure 1: A sentence including an interacting protein pair (p1-p2). (AIMed PMID 8652557, sentence 9, pair 3)

before	–
middle	PROT, and, interact, multiple, of, protein, subunit, with
after	protein

Table 1: BOW features

dently could be a better option.

2 The Combination of Kernels Based on Syntactic Parsers

In recent years, parsing technologies have improved rapidly and many different types of parsers have been proposed. Some of them are retrained using biomedical corpora and adapted to biomedical texts (Miyao et al., 2008). Kernel methods applicable to structured data have also been researched. Several kernels are adapted to the parsers' outputs and applied to PPI extraction. The parsers produce different types of structures providing different information regarding the target sentence. Each kernel uses different aspects to extract and utilize some portion of information from the outputs of their parsers. We combine the kernels for utilizing the multiple layers of information that the parsers and the kernels extracted. To realize our method, the PPI extraction method (Sætre et al., 2007; Miyao et al., 2008) is extended. We adopt two types of parsers and three kernels.

2.1 Syntactic Parsers

There are many types of parsers that output different layers of syntactic structures. The structures have different types of useful information. We focus on two types of parsers.

2.1.1 Dependency Parser

The task of a dependency parser is to take a sentence as a sequence of words, and to construct a dependency tree consisting of dependency links between words. Figure 2 is a parse tree produced by a dependency parser.

2.1.2 Deep Parser

A deep parser takes a sentence as a sequence of words like a dependency parser, and constructs graph structures that represent theory-specific syntactic/semantic relations among words. A predicate argument structure (PAS) is often used to represent the semantic structure. It is different from the dependency parser, because it also treats deeper relations and may include reentrant structures. Figure 3 is a parse graph produced by a deep parser.

2.2 Kernels

Syntactic parsers produce useful parse trees or graphs, but the extraction of information from these structures is an open problem. Several kernels are proposed to extract useful information from such structures. Words are also useful features, and several kernels are proposed to treat the combination of the words. We use the following kernels.

2.2.1 Bag-of-words (BOW) kernel

A bag-of-words kernel takes two unordered sets of words as feature vectors, and calculates their similarity. As input, three feature vectors are used. The vectors contain the lemma forms of the words before, inside of, and after the pair (Sætre et al., 2007). The lemmas in the vectors are limited to the top 1,000 most frequent lemmas. Table 1 shows BOW features of the sentence in Figure 1. Polynomial kernels are applied to each feature vector, and their outputs are summed up as the output of the kernel.

2.2.2 Subset tree kernel

A subset tree kernel (Moschitti, 2006) calculates the similarity between two input trees by counting their common subtrees. Subset tree kernels are applied to the shortest path between pairs from a parse tree. The shortest path is calculated including reverse relations to preserve the direction of the parse tree relations. The predicate information in PAS from the deep parser, which was unused in previous works (Sætre et al., 2007; Miyao and Tsujii, 2008), is used to represent the dependency types. An example of shortest path features can be found in Figure 4.

2.2.3 Graph Kernel

A graph kernel (Gärtner et al., 2003; Airola et al., 2008) calculates the similarity between two

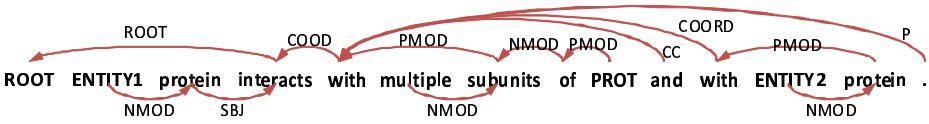

Figure 2: An output example produced by a dependency parser, Kenji and Tsujii's parser. (CoNLL-X dependency tree)

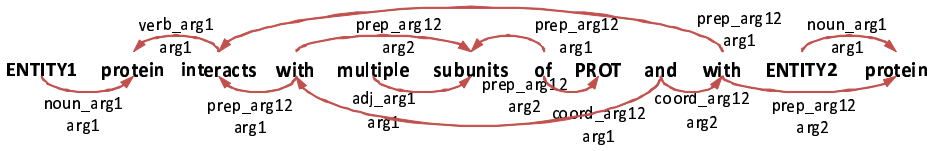

Figure 3: An output example produced by a deep parser, Enju. (Predicate argument structure)

input graphs by comparing the relations between common vertices. This graph kernel is called an all-dependency-paths graph kernel. The weights of the relations are calculated using all walks between each pair of vertices.

The graph consists of two directed subgraphs, a parse graph, and a graph representing a linear order of words. For a vertex in the first subgraph, the dependency and the lemma with POS in the parse graph are used. The dependency and lemma in the shortest path are distinguished from the others. For the PAS structure, the shortest path is calculated by using constituents, and the words in the constituents are distinguished. A vertex in the second subgraph is labeled with the lemma, the POS, and the place information. The place is separated into three by the target pair like BOW features (before, middle, after). Figure 5 is an example of subgraphs made from the parse tree in Figure 2.

For the calculation, two types of matrices are used: a label matrix $\mathbf{L}$, and an edge matrix $\mathbf{A}$. The label matrix is a $N \times L$ matrix, where N is the number of vertices, and L is the number of labels. It represents the correspondence between labels and vertices. $\mathbf{L}_{ij}$ is 1 if the i-th vertex corresponds to the j-th label, and 0 if otherwise. The edge matrix is a $N \times N$ matrix, and represents the relation between pairs of vertices. $\mathbf{A}_{ij}$ is a weight w_{ij} if the i-th vertex is connected to the j-th vertex, and 0 if otherwise. The weight is a predefined constant and the setting is found in the caption of Figure 5. Using the Neumann Series, a graph matrix $\mathbf{G}$ is calculated as:

$$\mathbf{G} = \mathbf{L}^{\mathbf{T}} \sum_{n=1}^{\infty} \mathbf{A}^n \mathbf{L} = \mathbf{L}^{\mathbf{T}}((\mathbf{I}-\mathbf{A})^{-1} - \mathbf{I})\mathbf{L}. \quad (1)$$

This matrix sums up the weights of all the walks between a pair of vertices, so as a result, each entry represents the strength of the relation between a pairs of vertices. Using these two graph matrices, the graph kernel k is defined as:

$$k(\mathbf{G}, \mathbf{G}') = \sum_{i=1}^{L} \sum_{j=1}^{L} \mathbf{G}_{ij} \mathbf{G'}_{ij}. \quad (2)$$

This kernel sums up the products of the common relations' weights.

For fast calculation and performance, the graph kernels of two subgraphs are calculated separately and the normalized outputs of the graph kernels are summed up. In the evaluation, we calculated the matrices of the weights beforehand, and entered the sparse feature vector of the weights into a linear SVM.

2.3 Combination of Kernels

The parsers treat different layers of relations. The dependency parsers ignore some deep information, and conversely, the deep parsers do not output certain shallow relations. Every kernel has different aspects, and has different advantages and disadvantages. The BOW kernels can combine the words easily, but they ignore the internal word order and word relations. The subset tree kernels can calculate the similarity of two shortest paths, but they ignore the words, the paths outside of the shortest path, and cycles in the parsed graphs. The graph kernels can treat the parser's output and word features at the same time. However, they cannot treat them properly without tuning kernel parameters. They may also miss some distant words, and similarities of multiple paths.

(*DEPENDENCY* (NMOD (ENTITY1 protein)) (SBJ (protein interact)) (rCOOD (interact with)) (rCOORD (with with)) (rPMOD (with protein)) (rNMOD (protein ENTITY2)))
(*DEEP* (noun_arg1_arg1 (ENTITY1 protein)) (rverb_arg1_arg1 (protein interact)) (rprep_arg12_arg1 (interact with)) (prep_arg12_arg2 (with protein)) (rnoun_arg1_arg1 (protein ENTITY2)))

Figure 4: Shortest path features

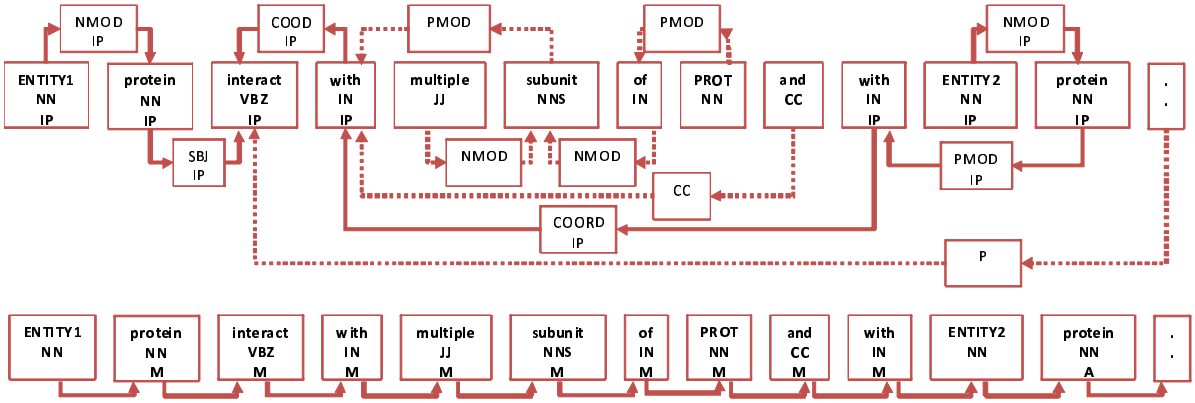

Figure 5: Two directed subgraphs features. One subgraph is made from a parse graph. As a label of a vertex in the subgraph, the relation with the place (IP:In shortest Path) and the lemma with the POS and the place are used. The other graph represents the linear order of words. As a label of a vertex in the subgraph, the lemma with the POS and the place (B:Before, M:Middle, A:After) is used. Weights are assigned to the edges in the subgraphs (Airola et al., 2008), 0.9 for the edges in the shortest paths and the second graph (represented with full lines), and 0.3 for the other edges (represented with dotted line).

The kernels calculate the similarity with different aspects between the two sentences. Combining the similarities can reduce the danger of missing important features, and can produce a new useful similarity measure. To realize the combination of the different types of kernels based on different parse structures, we sum up the normalized output of several kernels k_{ij} as:

$$k(X, X') = \sum_{i}^{K} \sum_{j}^{P} k_{ij}(X, X'), \qquad (3)$$

where K represents the number of types of kernels and P represents the number of parsers. This is a very simple combination, but the resulting kernel function contains all of the kernels' information.

3 Experiments

3.1 Experimental Settings

In the following experiments, we used AIMed (Bunescu and Mooney, 2004)[1], which is a major corpus for the evaluation of PPI extraction methods. We pre-processed AIMed for the named-entity tokenization in the following way. First, we converted spaces in protein names into "_", to group named entities. Then, we put a space between the end of a protein tag and the beginning of another protein tag when they were contiguous. Finally, we converted " to " or " according to their portions. AIMed consists of 225 abstracts (1970 sentences), and we extracted 5,648 binary-pairs including 1,005 positive pairs and 4,233 self-pairs including only other 54 positive pairs. Two examples of self-pairs are shown in Figure 6. Because of the pre-processing, the number of extracted pairs differs from other reported PPI extraction methods. Pre-processing can affect the performance and make it difficult to compare the result. The protein names in a sentence were converted to ENTITY1, ENTITY2, or PROT according to which pair was being processed. Examples are

Suramin induced high-affinity trimerization of *C8*$_{self\text{-}pair}$ (Kd = 0.10 microM at 20 degrees C) and dimerization of *C9*$_{self\text{-}pair}$ (Kd = 0.86 microM at 20 degrees C) .

Figure 6: Self-interaction pair examples. Each protein interacts with itself. (AIMed PMID 10346902, sentence 6, pair 4 and 5)

[1] ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/

shown in Figures 2 and 3.

Our system is based on the AKANE++ PPI system[2] (Sætre et al., 2007). We utilized Sagae and Tsujii's dependency parser[3] (Sagae and Tsujii, 2007) as the dependency parser, and the Enju parser[4] (Miyao and Tsujii, 2008) as the deep parser. Both parsers were retrained using the GENIA Treebank corpus[5] (Kim et al., 2003). As word features, we used the lemma information from the Enju parser, which originally output from the GENIA Tagger[6]. We used SVM for the evaluation. The performance was measured in an abstract-wise 10-fold cross validation (CV), and a one-answer-per-occurrence criterion, which were used for the evaluation of other PPI extraction methods before (Giuliano et al., 2006). We controlled the separating hyperplane of the SVM by varying the threshold and calculated the average of the results for each threshold. We fixed the other parameters, and we set the regularization parameter C to 1. We report the best f-value for each SVM in the following tables.

3.2 Effects of Self-pairs

We extracted 54 self-pairs from AIMed. The number of the self-pairs is much smaller than the number of the binary-pairs. Most previous results were obtained without the self-pairs. We evaluated the performance of our method in three different ways:

1. Evaluation without including any self-pairs
2. Evaluation without trying to predict any self-pairs
3. Evaluation with prediction of self-pairs

The first way ignores the self-pairs in prediction and evaluation. The result is shown in Table 2. We also showed the performance of the co-occurrence (or all-true) baseline. The result looks better than others, but it is too optimistic to assume that self-pairs can be classified with the same performance as the binary-pairs at the same threshold. However, this way is useful for comparing our method with other reported methods.

[2]http://www-tsujii.is.s.u-tokyo.ac.jp/~satre/akane/
[3]http://www.cs.cmu.edu/~sagae/parser/
[4]http://www-tsujii.is.s.u-tokyo.ac.jp/enju/
[5]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/
[6]http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/

	K	E	K+E
C (baseline)	30.2	–	–
B	53.9	–	–
T	55.5	54.6	57.0
G	59.2	58.4	60.6
T+B	60.1	59.6	60.4
G+B	60.0	59.5	61.0
T+G	62.1	61.6	62.6
T+G+B	62.6	62.7	64.3

Table 2: F-score without including any self-pairs (K:Sagae and Tsujii's parser, E:Enju, C:Co-occurrence, B:BOW, T:Subset Tree, G:Graph, +:Summation of kernels)

	K	E	K+E
CF (baseline)	30.0	–	–
B	52.6	–	–
T	54.0	53.2	55.7
G	57.8	57.1	59.2
T+B	58.7	58.2	59.0
G+B	58.5	58.1	59.4
T+G	60.4	60.2	61.1
T+G+B	61.1	61.1	62.7

Table 3: F-score without trying to predict any self-pairs (CF:Co-occurrence on binary-pairs and all-false on self-pairs)

	K	E	K+E
CF (baseline)	30.0	–	–
B	51.1	–	–
T	54.0	53.2	55.7
G	58.2	56.7	59.2
T+B	58.0	57.2	58.7
G+B	59.0	57.8	59.1
T+G	60.7	60.5	61.4
T+G+B	60.2	60.1	61.2

Table 4: F-score with prediction of self-pairs

The second way ignores the self-pairs in prediction, but adds the positive self-pairs as false negatives during evaluation. This way is the same as applying an all-false method to the self-pairs. The result is shown in Table 3. The results are always lower than the first one because of the self-pairs. The baseline was also calculated using the all-false method to the self-pairs. The third way considers the self-pairs in prediction and evaluation. The result is shown in Table 4. This is the right way when we want to make the PPI extraction system simple or we place much trust in machine learning methods. The baseline method is the same as the second one.

Some of the results using the graph kernels in the third way performed better than those in the

second way. The graph kernel can acquire the features on the self-pairs, and the graph kernel is one of the hopeful approaches in treating both pairs together. In other cases, however, the self-pairs confused the classifiers and the scores were low especially with the BOW kernels. It should be noted that the best threshold in Table 2 was different from the one in Table 4. This indicates that there is a suited threshold for each type of interaction pair. We can judge whether an interaction pair is a self-pair or not beforehand, and we do not need to treat both types of pairs together. It is better to predict the self-pairs and the binary-pairs independently to ease the task and improve the performance.

3.3 Accuracy Improvements by Combinations

In Table 3, we can find that the performance increased with an increase in information. This shows that both the combination of kernels and the combination of parsers are effective for PPI extraction. Our method performed best when all information was extracted. This is surprising, because AIMed is very small and there are much relevant information between the graph kernel and the other kernels, which can cause over-fitting.

The comparison with the results of related PPI methods is summarized in Table 5. The averages of Precision, Recall, and the F-score were calculated independently. For the fair comparison, we performed a 10-fold CV in each training data to tune the threshold for each fold, the result of which is shown as the F-scores in parentheses in Table 5. The F-scores in parentheses show that our evaluation method overestimated by only 0.8%, so it is not too optimistic . Additionally, we also calculated AUC (area under the ROC [receiver operating characteristic] curve) and standard deviations provided for the F-score and AUC (Airola et al., 2008). The ROC curve is a plot of TPR (true positive rate) vs FPR (false positive rate) for different thresholds. The F-score is the best point from a Precision-Recall (PR) curve (a plot of Precision vs Recall as we vary the threshold) . Because of these different points of view, the best result in AUC differs from the best result in the F-score. Which result is better depends on the given task; we have thus reported both results.

The results cannot be compared directly, because of the differences in data preprocessing, the different number of target protein pairs, and different evaluation methods. We compare our method with other methods based on the evaluation proposed in other PPI papers. We use the F-score for all the comparisons except for the comparison with (Airola et al., 2008), which used AUC for the first time in PPI extraction.

Our method outperformed all the PPI extraction methods evaluated with the abstract-wise 10-fold CV even though some of them ignored the self-pairs in their prediction and evaluation. The result of our method was 0.8% lower than the result of (Miyao et al., 2008) in the same condition, which is K+E and T+B in Table 4. This is because they did not use the predicate information in the PAS structure. On the other hand, this information increased the performance of the graph kernel. We may need to evaluate our method more precisely in order to decide the optimal input structures. Our method is different from (Airola et al., 2008) in that they performed the leave-one-document-out CV on the training data to tune the parameter. We compare their results with our result without including any self-pairs, because they ignored the self-pairs. Our method performed 7.1% better than theirs in F-score and 0.03 better than theirs in AUC. (Giuliano et al., 2006) performed a different evaluation method. As reported in (Airola et al., 2008), its result was an F-score of 52.4%. (Bunescu and Mooney, 2005b), (Erkan et al., 2007), and (Katrenko and Adriaans, 2006) also performed different evaluation methods. Their methods with our evaluation method is expected to give a lower performance (Sætre et al., 2007; Airola et al., 2008).

3.4 Error Analysis

Figures 7 and 8 show some false positive examples and some false negative examples. Some false positives and false negatives were caused by uncertainty and negation of interactions (Pyysalo et al., 2008). The kernels we used may not be able to distinguish these interactions from the others, because they do not extract modal information related to the interactions. The words representing the interactions may not exist in the shortest path in the subset tree kernel, and the information for the interactions may need relations among more than three words which are not retrieved in the graph kernel. Some synonyms and abbreviations

	positive	all binary	P	R	F	σ_F	AUC	σ_{AUC}
without including any self	1,005	5,648	60.4	69.3	64.3 (63.5)	4.3	0.879	0.026
without trying to predict any self	1,059	5,648	60.4	65.6	62.7 (62.0)	3.5	0.834	0.032
with the prediction of self	1,059	5,648	57.8	66.1	61.4 (60.7)	3.9	0.914	0.020
(Miyao et al., 2008)	1,059	5,648	54.9	65.5	59.5			
(Airola et al., 2008)	1,000	5,834	52.9	61.8	56.4	5.0	0.848	0.023
(Sætre et al., 2007)	1,068	5,631	64.3	44.1	52.0			
(Mitsumori et al., 2006)	1,107	5,476	54.2	42.6	47.7			
(Yakushiji et al., 2005)			33.7	33.1	33.4			

Table 5: Comparison with previous results of the PPI extraction methods with the abstract-wise 10-fold CV on the AIMed corpus. (The F-scores in parentheses were obtained using the 10-fold CV in each training data to tune the threshold for each fold.)

Uncertainty Interaction of p85$_{\text{prot}}$ subunit of *PI 3-kinase$_{prot}$* with *insulin$_{prot}$* and IGF-1$_{\text{prot}}$ receptors analysed by using the two-hybrid system .

Synonyms The catalytic domain of activated *collagenase_I$_{prot}$* (MMP-1$_{\text{pair1, pair2}}$) is absolutely required for interaction with its specific inhibitor , *tissue_inhibitor_of_metalloproteinases-1$_{pair1}$* (TIMP-1$_{\text{pair2}}$) .

Other Misclassification A 51-residue region from the conserved C-terminal region of TBP$_{\text{pair12, pair13, pair14}}$, previously shown to be the binding site for the viral activator protein *E1A$_{pair12}$* , interacts with *c-Fos$_{pair13}$* and c-Jun$_{\text{pair14}}$ proteins.

Figure 7: False positive examples. Mis-detected relations are shown in italic.

Need Information of Context We screened proteins for interaction with *presenilin_(_PS_)_1$_{pair1}$*, and cloned the full-length cDNA of human *delta-catenin$_{pair1}$*, which encoded 1225 amino acids.

Need Information of Context We have engineered and purified recombinant *K5$_{pair5}$* head and *DPI$_{pair5}$* tail , and we demonstrate direct interaction in vitro by solution-binding assays and by ligand blot assays .

Negation This demonstrates that the C-terminal hemopexin domain of MMP-1$_{\text{prot}}$, in contrast to the corresponding regions of *gelatinase_A$_{pair4}$* and *gelatinase_B$_{pair5}$* , does not interact with *TIMP-1$_{pair4, pair5}$* .

Other Misclassification Using the cytoplasmic domain of Fas in the yeast two-hybrid system , we have identified a novel interacting protein , *FADD*$_{\text{pair2, }pair3\text{, pair4}}$, which binds Fas$_{\text{pair2}}$ and *Fas$_{pair3}$* - FD5$_{\text{pair4}}$, a mutant of Fas$_{\text{prot}}$ possessing enhanced killing activity , but not the functionally inactive mutants Fas$_{\text{prot}}$ - LPR$_{\text{prot}}$ and Fas$_{\text{prot}}$ - FD8$_{\text{prot}}$.

Figure 8: False negative examples. Undetected interactions are shown in italic.

were also among the false positives. They cannot be distinguished from the true positives because the protein names are hidden. In false negatives, there were interactions that needed more information of the words and the context for the extraction. Some of the interactions may be extracted by using the incidental features in many texts, but other interactions will not be detected by the current sentence-based approach.

4 Conclusion and Future Work

In this paper, we have proposed an approach using a combination of kernels for PPI extraction, which can in turn extract and combine several different layers of information from a sentence and its syntactic structures by using several parsers. Each kernel extracts some information from the sentence with different aspects and loses the other information in it. The combination of the kernels can gather up all the kernels' information and cover some of the lost information. To show the usefulness of the combination of kernels and parsers in the PPI extraction, we evaluated our method using the AIMed corpus. We achieved an F-score of 62.0% using a SVM. This result is better than the results of all the current state-of-the-art PPI extraction methods.

We also analyzed how the performance changed when including the self-pairs. The re-

sults of the graph kernels showed that it may be a good solution to treat both kinds of pairs together. We should, however, predict the binary-pairs and the self-pairs independently to make the task easier and to improve the performance, because the best threshold for the binary-pairs is different from the one for the self-pairs. When we predicted the binary-pairs and the self-pairs at the same time, the classifiers produced worse results than the results of the classifiers classifying all the self-pairs as false. We need a way to detect the self-pairs better than the all-false method.

There is the possibility of improvement by adding (or replacing) new kernels and parsers. This is to remedy the fact that the parsers we used may miss some information in a sentence and the kernels we used may not extract full information of the parsers' outputs. There may be relevant and redundant information in the combination of all the kernels, which can confuse classifiers. How redundant features affect the performance need to be analyzed before adopting new kernels and parsers for evaluating them correctly.

Acknowledgments

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

References

A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of the BioNLP 2008 workshop*, pages 1–9.

C. Blaschke, M. A. Andrade, C. Ouzounis, and A. Valencia. 1999. Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the AAAI Conference on Intelligent Systems in Molecular Biology*, pages 60–67.

R. C. Bunescu and R. J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL 2004*, pages 439–446.

R. C. Bunescu and R. J. Mooney. 2005a. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP*, pages 724–731, Morristown, NJ, USA. Association for Computational Linguistics.

R. C. Bunescu and R. J. Mooney. 2005b. Subsequence kernels for relation extraction. In *NIPS 2005*.

R. C. Bunescu, R. Ge, R. J. Kate, E. M. Marcotte, R. J. Mooney, A. K. Ramani, and Y. W. Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

G. Erkan, A. Ozgur, and D. R. Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *EMNLP 2007*.

T. Gärtner, P. A. Flach, and S. Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and the 7th Kernel Workshop*.

C. Giuliano, A. Lavelli, and L. Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL 2006*.

S. Katrenko and P. Adriaans. 2006. Learning relations from biomedical corpora using dependency trees. In *KDECB*, pages 61–80.

J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus — a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19:i180–i182.

S. Mathivanan, B. Periaswamy, T. Gandhi, K. Kandasamy, S. Suresh, R. Mohmood, Y. L. Ramachandra, and A. Pandey. 2006. An evaluation of human protein-protein interaction data in the public domain. *BMC Bioinformatics*, 7 Suppl 5:S19.

T. Mitsumori, M. Murata, Y. Fukuda, K. Doi, and H. Doi. 2006. Extracting protein-protein interaction information from biomedical text with SVM. *IEICE - Transactions on Information and Systems*, E89-D(8):2464–2466.

Y. Miyao and J. Tsujii. 2008. Feature forest models for probabilistic HPSG parsing. *Computational Linguistics*, 34(1):35–80.

Y. Miyao, R. Sætre, K. Sagae, T. Matsuzaki, and J. Tsujii. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of ACL-08: HLT*, pages 46–54.

A. Moschitti. 2006. Making tree kernels practical for natural language processing. In *EACL 2006*.

S. Pyysalo, A. Airola, J. Heimonen, J. Björne, F. Ginter, and T. Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. In *BMC Bioinformatics*, volume 9(Suppl 3), page S6.

R. Sætre, K. Sagae, and J. Tsujii. 2007. Syntactic features for protein-protein interaction extraction. In *LBM 2007 short papers*.

K. Sagae and J. Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007*.

A. Yakushiji, Y. Miyao, Y. Tateisi, and J. Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *First International Symposium on Semantic Mining in Biomedicine*.