Automatic Construction of Predicate-argument Structure Patterns for Biomedical Information Extraction

ane Yakushiji[*†] **Yusuke Miyao**[*] **Tomoko Ohta**[*] **Yuka Tateisi**[*‡] **Jun'ichi Tsujii**[*§]
[*] Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 JAPAN
[§] School of Informatics, University of Manchester
POBox 88, Sackville St, MANCHESTER M60 1QD, UK
{akane, yusuke, okap, yucca, tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper presents a method of automatically constructing information *extraction patterns* on predicate-argument structures (PASs) obtained by full parsing from a smaller training corpus. Because PASs represent generalized structures for syntactical variants, patterns on PASs are expected to be more generalized than those on surface words. In addition, patterns are divided into components to improve recall and we introduce a Support Vector Machine to learn a prediction model using pattern matching results. In this paper, we present experimental results and analyze them on how well protein-protein interactions were extracted from MEDLINE abstracts. The results demonstrated that our method improved accuracy compared to a machine learning approach using surface word/part-of-speech patterns.

1 Introduction

One primitive approach to Information Extraction (IE) is to manually craft numerous extraction patterns for particular applications and this is presently one of the main streams of biomedical IE (Blaschke and Valencia, 2002; Koike et al., 2003). Although such IE attempts have demonstrated near-practical performance, the same sets of patterns cannot be applied to different kinds of information. A real-world task requires several kinds of IE, thus manually engineering extraction patterns, which is tedious and time-consuming process, is not really practical.

Techniques based on machine learning (Zhou et al., 2005; Hao et al., 2005; Bunescu and Mooney, 2006) are expected to alleviate this problem in manually crafted IE. However, in most cases, the cost of manually crafting patterns is simply transferred to that for constructing a large amount of training data, which requires tedious amount of manual labor to annotate text.

To systematically reduce the necessary amount of training data, we divided the task of constructing extraction patterns into a subtask that general natural language processing techniques can solve and a subtask that has specific properties according to the information to be extracted. The former subtask is of full parsing (i.e. recognizing syntactic structures of sentences), and the latter subtask is of constructing specific extraction patterns (i.e. finding clue words to extract information) based on the obtained syntactic structures.

We adopted full parsing from various levels of parsing, because we believe that it offers the best utility to generalize sentences into normalized syntactic relations. We also divided patterns into components to improve recall and we introduced machine learning with a Support Vector Machine (SVM) to learn a prediction model using the matching results of extraction patterns. As an actual IE task, we extracted pairs of interacting protein names from biomedical text.

2 Full Parsing

2.1 Necessity for Full Parsing

A technique that many previous approaches have used is shallow parsing (Koike et al., 2003; Yao et al., 2004; Zhou et al., 2005). Their assertion is

Current Affiliation:
† FUJITSU LABORATORIES LTD.
‡ Faculty of Informatics, Kogakuin University

Distance	Count	(%)	Sum (%)
−1	54	5.0	5.0
0	8	0.7	5.7
1	170	15.7	21.4
2–5	337	31.1	52.5
6–10	267	24.6	77.1
11–	248	22.9	100.0

Distance −1 means protein word has been annotated as interacting with itself (e.g. "actin polymerization"). Distance 0 means words of the interacting proteins are directly next to one another. Multi-word protein names are concatenated as long as they do not cross tags to annotate proteins.

Table 1: Distance between Interacting Proteins

that shallow parsers are more robust and would be sufficient for IE. However, their claims that shallow parsers are sufficient, or that full parsers do not contribute to application tasks, have not been fully proved by experimental results.

Zhou et al. (2005) argued that most information useful for IE derived from full parsing was shallow. However, they only used dependency trees and paths on full parse trees in their experiment. Such structures did not include information of semantic subjects/objects, which full parsing can recognize. Additionally, most relations they extracted from the ACE corpus (Linguistic Data Consortium, 2005) on broadcasts and newswires were within very short word-distance (70% where two entities are embedded in each other or separated by at most one word), and therefore shallow information was beneficial. However, Table 1 shows that the word distance is long between interacting protein names annotated on the AImed corpus (Bunescu and Mooney, 2004), and we have to treat long-distance relations for information like protein-protein interactions.

Full parsing is more effective for acquiring generalized data from long-length words than shallow parsing. The sentences at left in Figure 1 exemplify the advantages of full parsing. The gerund "activating" in the last sentence takes a non-local semantic subject "*ENTITY1*", and shallow parsing cannot recognize this relation because "*ENTITY1*" and "activating" are in different phrases. Full parsing, on the other hand, can identify both the subject of the whole sentence and the semantic subject of "activating" have been shared.

2.2 Predicate-argument Structures

We applied Enju (Tsujii Laboratory, 2005a) as a full parser which outputs predicate-argument structures (PASs). PASs are well normalized forms that represent syntactic relations. Enju is based on Head-driven Phrase Structure Grammar (Sag and Wasow, 1999), and it has been trained on the Penn Treebank (PTB) (Marcus et al., 1994) and a biomedical corpus, the GENIA Treebank (GTB) (Tsujii Laboratory, 2005b). We used a part-of-speech (POS) tagger trained on the GENIA corpus (Tsujii Laboratory, 2005b) as a preprocessor for Enju. On predicate-argument relations, Enju achieved 88.0% precision and 87.2% recall on PTB, and 87.1% precision and 85.4% recall on GTB.

The illustration at right in Figure 1 is a PAS example, which represents the relation between "activate", "*ENTITY1*" and "*ENTITY2*" of all sentences to the left. The predicate and its arguments are words converted to their base forms, augmented by their POSs. The arrows denote the connections from predicates to their arguments and the types of arguments are indicated as arrow labels, i.e., ARGn (n $= 1, 2, \ldots$), MOD. For example, the semantic subject of a transitive verb is ARG1 and the semantic object is ARG2.

What is important here is, thanks to the strong normalization of syntactic variations, that we can expect that the construction algorithm for extracting patterns that works on PASs will need a much smaller training corpus than those working on surface-word sequences. Furthermore, because of the reduced diversity of surface-word sequences at the PAS level, any IE system at this level should demonstrate improved recall.

3 Related Work

Sudo et al. (2003), Culotta and Sorensen (2004) and Bunescu and Mooney (2005) acquired substructures derived from dependency trees as extraction patterns for IE in general domains. Their approaches were similar to our approach using PASs derived from full parsing. However, one problem with their systems is that they could not treat non-local dependencies such as semantic subjects of gerund constructions (discussed in Section 2), and thus rules acquired from the constructions were partial.

Bunescu and Mooney (2006) also learned extraction patterns for protein-protein interactions by SVM with a generalized subsequence kernel. Their patterns are sequences of words, POSs, entity types, etc., and they heuristically restricted length and word positions of the patterns. Al-

ENTITY1 recognizes and **activates** *ENTITY2*.
ENTITY2 **activated** by *ENTITY1* are not well characterized.
The herpesvirus encodes a functional *ENTITY1* that **activates** human *ENTITY2*.
ENTITY1 can functionally cooperate to synergistically **activate** *ENTITY2*.
The *ENTITY1* plays key roles by **activating** *ENTITY2*.

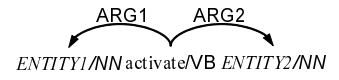

Figure 1: Syntactical Variations of "activate"

though they achieved about 60% precision and about 40% recall, these heuristic restrictions could not be guaranteed to be applied to other IE tasks.

Hao et al. (2005) learned extraction patterns for protein-protein interactions as sequences of words, POSs, entity tags and gaps by dynamic programming, and reduced/merged them using a minimum description length-based algorithm. Although they achieved 79.8% precision and 59.5% recall, sentences in their test corpus have too many positive instances and some of the patterns they claimed to have been successfully constructed went against linguistic or biomedical intuition. (e.g. "*ENTITY1* and interacts with *ENTITY2*" should be replaced by a more general pattern because they aimed to reduce the number of patterns.)

4 Method

We automatically construct patterns to extract protein-protein interactions from an annotated training corpus. The corpus needs to be tagged to denote which protein words are interacting pairs.

We follow five steps in constructing extraction patterns from the training corpus. (1) Sentences in the training corpus are parsed into PASs and we extract raw patterns from the PASs. (2) We divide the raw patterns to generate both *combination* and *fragmental patterns*. Because obtained patterns include inappropriate ones (wrongly generated or too general), (3) we apply both kinds of patterns to PASs of sentences in the training corpus, (4) calculate the scores for matching results of combination patterns, and (5) make a prediction model with SVM using these matching results and scores.

We extract pairs of interacting proteins from a target text in the actual IE phase, in three steps. (1) Sentences in the target corpus are parsed into PASs. (2) We apply both kinds of extraction patterns to these PASs and (3) calculate scores for combination pattern matching. (4) We use the prediction model to predict interacting pairs.

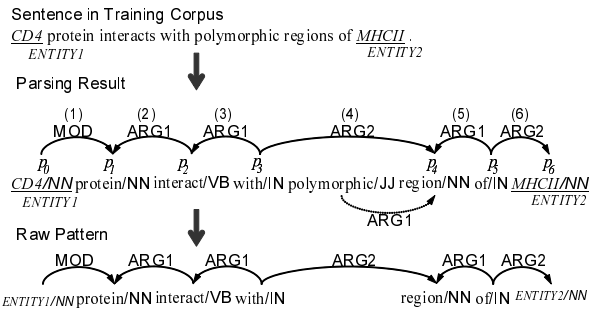

Figure 2: Extraction of Raw Pattern

4.1 Full Parsing and Extraction of Raw Patterns

As the first step in both the construction phase and application phase of extraction patterns, we parse sentences into PASs using Enju.[1] We label all PASs of the protein names as protein PASs.

After parsing, we extract the smallest set of PASs, which *connect* words that denote interacting proteins, and make it a raw pattern. We take the same method to extract and refine raw patterns as Yakushiji et al. (2005). *Connecting* means we can trace predicate-argument relations from one protein word to the other in an interacting pair. The procedure to obtain a raw pattern $(p_0, \ldots, p_n)$ is as follows:
$predicate(p)$: PASs that have p as their argument
$argument(p)$: PASs that p has as its arguments

1. $p_i = p_0$ is the PAS of a word correspondent to one of interacting proteins, and we obtain candidates of the raw pattern as follows:
   1-1. If p_i is of the word of the other interacting protein, $(p_0, \ldots, p_i)$ is a candidate of the raw pattern.
   1-2. If not, make pattern candidates for each $p_{i+1} \in predicate(p_i) \cup argument(p_i) - \{p_0, \ldots, p_i\}$ by returning to 1-1.
2. Select the pattern candidate of the smallest set as the raw pattern.

[1] Before parsing, we concatenate each multi-word protein name into the one word as long as the concatenation does not cross name boundaries.

3. Substitute variables (*ENTITY1*, *ENTITY2*) for the predicates of PASs correspondent to the interacting proteins.

The lower part of Figure 2 shows an example of the extraction of a raw pattern. "*CD4*" and "*MHCII*" are words representing interacting proteins. First, we set the PAS of "*CD4*" as p_0. $argument(p_0)$ includes the PAS of "protein", and we set it as p_1 (in other words, tracing the arrow (1)). Next, $predicate(p_1)$ includes the PAS of "interact" (tracing the arrow (2) back), so we set it as p_2. We continue similarly until we reach the PAS of "*MHCII*" (p_6). The result of the extracted raw pattern is the set of $p_0, \ldots, p_6$ with substituting variables *ENTITY1* and *ENTITY2* for "*CD4*" and "*MHCII*".

There are some cases where an extracted raw pattern is not appropriate and we need to refine it. One case is when unnecessary coordinations/parentheses are included in the pattern, e.g. two interactions are described in a combined representation ("*ENTITY1* binds this protein and *ENTITY2*"). Another is when two interacting proteins are connected directly by a conjunction or only one protein participates in an interaction. In such cases, we refine patterns by unfolding of coordinations/parentheses and extension of patterns, respectively. We have omitted detailed explanations because of space limitations. The details are described in the work of Yakushiji et al. (2005).

4.2 Division of Patterns

Division for generating combination patterns is based on observation of Yakushiji et al. (2005) that there are many cases where combinations of verbs and certain nouns form IE patterns. In the work of Yakushiji et al. (2005), we divided only patterns that include only one verb. We have extended the division process to also treat nominal patterns or patterns that include more than one verb.

Combination patterns are not appropriate for utilizing individual word information because they are always used in rather strictly combined ways. Therefore we have newly introduced fragmental patterns which consist of independent PASs from raw patterns, in order to use individual word information for higher recall.

4.2.1 Division for Generating Combination Patterns

Raw patterns are divided into some components and the components are combined to construct combination patterns according to types of the division. There are three types of division of raw patterns for generating combination patterns. These are:

(a) Two-entity Division
  (a-1) Entity-Main-Entity Division
  (a-2) Main-Entity-Entity Division
(b) Single-entity Division, and
(c) No Division (Naive Patterns).

Most raw patterns, where entities are at both ends of the patterns, are divided into Entity-Main-Entity. Main-Entity-Entity are for the cases where there are PASs other than entities at the ends of the patterns (e.g. "interaction between *ENTITY1* and *ENTITY2*"). Single-entity is a special Main-Entity-Entity for interactions with only one participant (e.g. "*ENTITY1* dimerization").

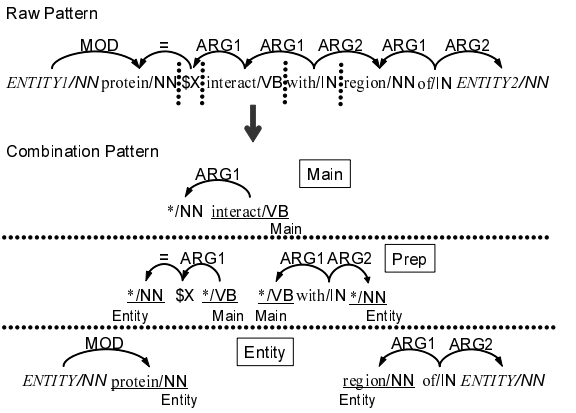

Figure 3: Division of Raw Pattern into Combination Pattern Components (Entity-Main-Entity)

There is an example of Entity-Main-Entity division in Figure 3. First, the *main component* from the raw pattern is the syntactic head PAS of the raw pattern. If the raw pattern corresponds to a sentence, the syntactic head PAS is the PAS of the main verb. We underspecify the arguments of the main component, to enable them to unify with the PASs of any words with the same POSs. Next, if there are PASs of prepositions connecting to the main component, they become *prep components*. If there is no PAS of a preposition next to the main component on the connecting link from the main component to an entity, we make the *pseudo PAS* of a *null preposition* the prep component. The left prep component ($X) in Figure 3 is a pseudo PAS of a null preposition. We also underspecify the arguments of prep components. Finally, the remaining two parts, which are typically noun phrases, of the raw pattern become *entity components*. PASs

corresponding to the entities of the original pair are labeled as only unifiable with the entities of other pairs.

Main-Entity-Entity division is similar, except we distinguish only one prep component as a *double-prep component* and the PAS of the coordinate conjunction between entities becomes the *coord component*. Single-entity division is similar to Main-Entity-Entity division and the difference is that single-entity division produces no coord and one entity component. *Naive patterns* are patterns without division, where no division can be applied (e.g. "*ENTITY1/NN* in/IN complexes/NN with/IN *ENTITY2/NN*").

All PASs on boundaries of components are labeled to determine which PAS on a boundary of another component can be unified. Labels are represented by subscriptions in Figure 3. These restrictions on component connection are used in the step of constructing combination patterns.

Constructing combination patterns by combining components is equal to reconstructing original raw patterns with the original combination of components, or constructing new raw patterns with new combinations of components. For example, an Entity-Main-Entity pattern is constructed by combination of any main, any two prep and any two entity components. Actually, this construction process by combination is executed in the pattern matching step. That is, we do not off-line construct all possible combination patterns from the components and only construct the combination patterns that are able to match the target.

4.2.2 Division for Generating Fragmental Patterns

A raw pattern is splitted into individual PASs and each PAS becomes a fragmental pattern. We also prepare underspecified patterns where one or more of the arguments of the original are underspecified, i.e., are able to match any words of the same POSs and the same label of protein/not-protein. We underspecify the PASs of entities in fragmental patterns to enable them to unify with any PASs with the same POSs and a protein label, although in combination patterns we retain the PASs of entities as only unifiable with entities of pairs. This is because fragmental patterns are designed to be less strict than combination patterns.

4.3 Pattern Matching

Matching of combination patterns is executed as a process to match and combine combination pattern components according to their division types (Entity-Main-Entity, Main-Entity-Entity, Single-entity and No Division). Fragmental matching is matching all fragmental patterns to PASs derived from sentences.

4.4 Scoring for Combination Matching

We next calculate the score of each combination matching to estimate the adequacy of the combination of components. This is because new combination of components may form inadequate patterns. (e.g. "*ENTITY1* be *ENTITY2*" can be formed of components from "*ENTITY1* be *ENTITY2* receptor".) Scores are derived from the results of combination matching to the source training corpus.

We apply the combination patterns to the training corpus, and count pairs of True Positives (TP) and False Positives (FP). The scores are calculated basically by the following formula:

$$Score = TP/(TP + FP) + \alpha \times TP$$

This formula is based on the precision of the pattern on the training corpus, i.e., an estimated precision on a test corpus. α works for smoothing, that is, to accept only patterns of large TP when $FP = 0$. α is set as 0.01 empirically. The formula is similar to the Apriori algorithm (Agrawal and Srikant, 1995) that learns association rules from a database. The first term corresponds to the confidence of the algorithm, and the second term corresponds to the support.

For patterns where $TP = FP = 0$, which are not matched to PASs in the training corpus (i.e., newly produced by combinations of components), we estimates TP' and FP' by using the confidence of the main and entity components. This is because main and entity components tend to contain pattern meanings, whereas prep, double-prep and coord components are rather functional. The formulas to calculate the scores for all cases are:

$$Score = \begin{cases} TP/(TP + FP) + \alpha \times TP \\ \quad (TP + FP \neq 0) \\ TP'/(TP' + FP') \\ \quad (TP = FP = 0, TP' + FP' \neq 0) \\ 0 \quad (TP = FP = TP' = FP' = 0) \end{cases}$$

Combination Pattern	
(1)	Combination of components in combination matching
(2)	Main component in combination matching
(3)	Entity components in combination matching
(4)	Score for combination matching (SCORE)
Fragmental Pattern	
(5)	Matched fragmental patterns
(6)	Number of PASs of example that are not matched in fragmental matching
Raw Pattern	
(7)	Length of raw pattern derived from example

Table 2: Features for SVM Learning of Prediction Model

$$TP' = \begin{cases} TP'_{main} + TP'_{entity1}(+TP'_{entity2}) \\ \quad \text{(for Two-entity, Single-entity)} \\ 0 \quad \text{(for Naive)} \end{cases}$$

$$FP' = (\text{similar to } TP' \text{ but } TP'_x \text{ is replaced by } FP'_x)$$

$$TP'_{main} = \begin{cases} TP_{main:two}/(TP_{main:two} + FP_{main:two}) \\ \quad \left(\begin{array}{l} TP_{main:two} + FP_{main:two} \neq 0, \\ \text{for Two-entity} \end{array} \right) \\ TP_{main:single}/(TP_{main:single} + FP_{main:single}) \\ \quad \left(\begin{array}{l} TP_{main:single} + FP_{main:single} \neq 0, \\ \text{for Single-entity} \end{array} \right) \\ 0 \quad \text{(other cases)} \end{cases}$$

$$TP'_{entityi} = \begin{cases} TP_{entityi}/(TP_{entityi} + FP_{entityi}) \\ \quad \left(TP_{entityi} + FP_{entityi} \neq 0 \right) \\ 0 \quad \text{(other cases)} \end{cases}$$

$$FP'_x = \left(\begin{array}{l} \text{similar to } TP'_x \text{ but } TP'_y \text{ in the} \\ \text{numerators is replaced by } FP'_y \end{array} \right)$$

- TP: number of TPs by the combination of components
- $TP_{main:two}$: sum of TPs by two-entity combinations that include the same main component
- $TP_{main:single}$: sum of TPs by single-entity combinations that include the same main component
- $TP_{entityi}$: sum of TPs by combinations that include the same entity component which is not the straight entity component
- FP_x: similar to TP_x but TP is replaced by FP

The entity component "*ENTITY/NN*", which only consists of the PAS of an entity, adds no information to combinations of components. We call this component a *straight entity component* and exclude its effect from the scores.

4.5 Construction of Prediction Model

We use an SVM to learn a prediction model to determine whether a new protein pair is interacting. We used SVM^{light} (Joachims, 1999) with an rbf kernel, which is known as the best kernel for most tasks. The prediction model is based on the features of Table 2.

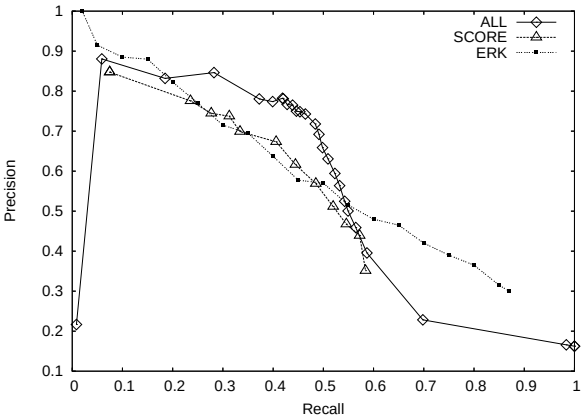

Figure 4: Results of IE Experiment

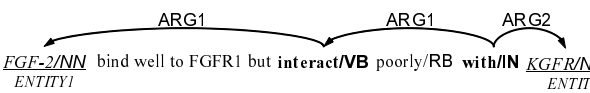

Figure 5: Example Demonstrating Advantages of Full Parsing

5 Results and Discussion

5.1 Experimental Results on the AImed Corpus

To evaluate extraction patterns automatically constructed with our method, we used the AImed corpus, which consists of 225 MEDLINE (U.S. National Library of Medicine, 2006) abstracts (1969 sentences) annotated with protein names and protein-protein interactions, for the training/test corpora. We used tags for the protein names given.

We measured the accuracy of the IE task using the same criterion as Bunescu and Mooney (2006), who used an SVM to construct extraction patterns on word/POS/type sequences from the AImed corpus. That is, an extracted interaction from an abstract is correct if the proteins are tagged as interacting with each other *somewhere* in that abstract (document-level measure).

Figure 4 plots our 10-fold cross validation and the results of Bunescu and Mooney (2006). The line ALL represents results when we used all features for SVM learning. The line SCORE represents results when we extracted pairs with higher combination matching scores than various threshold values. And the line ERK represents results by Bunescu and Mooney (2006).

The line ALL obtained our best overall F-measure 57.3%, with 71.8% precision and 48.4% recall. Our method was significantly better than Bunescu and Mooney (2006) for precision be-

tween 50% and 80%. It also needs to be noted that SCORE, which did not use SVM learning and only used the combination patterns, achieved performance comparable to that by Bunescu and Mooney (2006) for the precision range from 50% to 80%. And for this range, introducing the fragmental patterns with SVM learning raised the recall. This range of precision is practical for the IE task, because precision is more important than recall for significant interactions that tend to be described in many abstracts (as shown by the next experiment), and too-low recall accompanying too-high precision requires an excessively large source text.

Figure 5 shows the advantage of introducing full parsing. "FGF-2" and "KGFR" is an interacting protein pair. The pattern "*ENTITY1* interact with *ENTITY2*" based on PASs successfully extracts this pair. However, it is difficult to extract this pair with patterns based on surface words, because there are 5 words between "FGF-2" and "interact".

5.2 Experimental Results on Abstracts of MEDLINE

We also conducted an experiment to extract interacting protein pairs from a large amount of biomedical text, i.e. about 14 million titles and 8 million abstracts in MEDLINE. We constructed combination patterns from all 225 abstracts of the AImed corpus, and calculated a threshold value of combination scores that produced about 70% precision and 30% recall on the training corpus. We extracted protein pairs with higher combination scores than the threshold value. We excluded single-protein interactions to reduce time consumption and we used a protein name recognizer in this experiment[2].

We compared the extracted pairs with a manually curated database, Reactome (Joshi-Tope et al., 2005), which published 16,564 human protein interaction pairs as pairs of Entrez Gene IDs (U.S. National Library of Medicine, 2006). We converted our extracted protein pairs into pairs of Entrez Gene IDs by the protein name recognizer.[3] Because there may be pairs missed by Reactome or pairs that our processed text did not include, we excluded extracted pairs of IDs that are not included in Reactome and excluded Reactome pairs of IDs that do not co-occur in the sentences of our processed text.

[2]Because protein names were recognized after the parsing, multi-word protein names were not concatenated.

[3]Although the same protein names are used for humans and other species, these are considered to be human proteins without checking the context. This is a fair assumption because Reactome itself infers human interaction events from experiments on model organisms such as mice.

Total	89
Parsing Error/Failure (Related to coordinations)	35 (14)
Lack of Combination Pattern Component	33
Requiring Anaphora Resolution	9
Error in Prediction Model	8
Requiring Attributive Adjectives	5
Others	10

More than one cause can occur in one error, thus the sum of all causes is larger than the total number of False Negatives.

Table 3: Causes of Error for FNs

After this postprocessing, we found that we had extracted 7775 human protein pairs. Of them, 155 pairs were also included in Reactome ([a] pseudo TPs) and 7620 pairs were not included in Reactome ([b] pseudo FPs). 947 pairs of Reactome were not extracted by our system ([c] pseudo False Negatives (FNs)). However, these results included pairs that Reactome missed or those that only co-occurred and were not interacting pairs in the text. There may also have been errors with ID assignment.

To determine such cases, a biologist investigated 100 pairs randomly selected from pairs of pseudo TPs, FPs and FNs retaining their ratio of numbers. She also checked correctness of the assigned IDs. 2 pairs were selected from pseudo TPs, 88 pairs were from pseudo FPs and 10 pairs were from pseudo FNs. The biologist found that 57 pairs were actual TPs (2 pairs of pseudo TPs and 55 pairs of pseudo FPs) and 32 pairs were actual FPs of the pseudo FPs. Thus, the precision was 64.0% in this sample set. Furthermore, even if we assume that all pseudo FNs are actual FNs, the recall can be estimated by actual TPs / (actual TPs + pseudo FNs) $\times$ 100 = 83.8%.

These results mean that the recall of an IE system for interacting proteins is improved for a large amount of text even if it is low for a small corpus. Thus, this justifies our assertion that a high degree of precision in the low-recall range is important.

5.3 Error Analysis

Tables 3 and 4 list causes of error for FNs/FPs on a test set of the AImed corpus using the prediction model with the best F-measure with all the

Total	35
Requiring Attributive Adjectives	13
Corpus Error	11
Error in Prediction Model	5
Requiring Negation Words	2
Parsing Error	1
Others	3

Table 4: Causes of Error for FPs

features. Different to Subsection 5.1, we individually checked each occurring pair of interacting proteins. The biggest problems were parsing error/failure, lack of necessary patterns and learning of inappropriate patterns.

5.3.1 Parsing Error

As listed in Table 3, 14 (40%) of the 35 parsing errors/failures were related to coordinations. Many of these were caused by differences in the characteristics of the PTB/GTB, the training corpora for Enju, and the AImed Corpus. For example, Enju failed to obtain the correct structure for "the *ENTITY1* / *ENTITY1* complex" because words in the PTB/GTB are not segmented with "/" and Enju could not be trained on such a case. One method to solve this problem is to avoid segmenting words with "/" and introducing extraction patterns based on surface characters, such as "*ENTITY1*/*ENTITY2* complex".

Parsing errors are intrinsic problems to IE methods using parsing. However, from Table 3, we can conclude that the key to gaining better accuracy is refining of the method with which the PAS patterns are constructed (there were 46 related FNs) rather than improving parsing (there were 35 FNs).

5.3.2 Lack of Necessary Patterns and Learning of Inappropriate Patterns

There are two different reasons causing the problems with the lack of necessary patterns and the learning of inappropriate patterns: (1) the training corpus was not sufficiently large to saturate IE accuracy and (2) our method of pattern construction was too limited.

Effect of Training Corpus Size To investigate whether the training corpus was large enough to maximize IE accuracy, we conducted experiments on training corpora of various sizes. Figure 6 plots graphs of F-measures by SCORE and Figure 7 plots the number of combination patterns on training corpora of various sizes. From Figures 6 and 7, the training corpus (207 abstracts at a maximum)

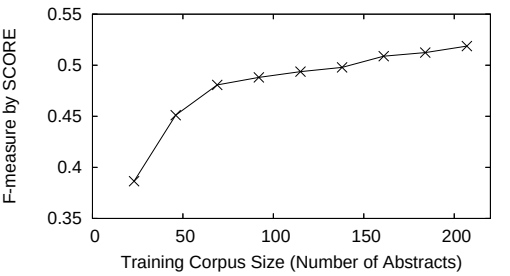

Figure 6: Effect of Training Corpus Size (1)

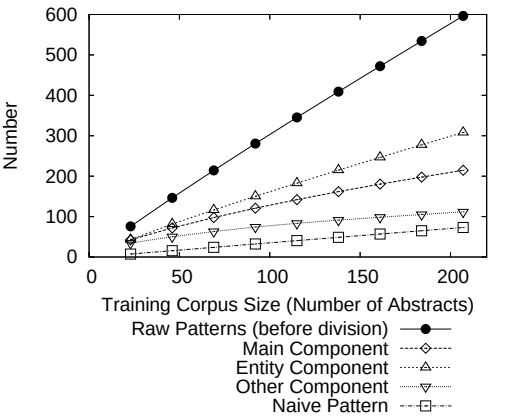

Figure 7: Effect of Training Corpus Size (2)

is not large enough. Thus increasing corpus size will further improve IE accuracy.

Limitation of the Present Pattern Construction The limitations with our pattern construction method are revealed by the fact that we could not achieve a high precision like Bunescu and Mooney (2006) within the high-recall range. Compared to theirs, one of our problems is that our method could not handle attributives. One example is "binding property of *ENTITY1* to *ENTITY2*". We could not obtain "binding" because the smallest set of PASs connecting "*ENTITY1*" and "*ENTITY2*" includes only the PASs of "property", "of" and "to". To handle these attributives, we need distinguish necessary attributives from those that are general[4] by semantic analysis or bootstrapping.

Another approach to improve our method is to include local information in sentences, such as surface words between protein names. Zhao and Grishman (2005) reported that adding local information to deep syntactic information improved IE results. This approach is also applicable to IE in other domains, where related entities are in a short

[4]Consider the case where a source sentence for a pattern is "*ENTITY1* is an important homodimeric protein." ("homodimeric" represents that two molecules of "*ENTITY1*" interact with each other.)

distance like the work of Zhou et al. (2005).

6 Conclusion

We proposed the use of PASs to construct patterns as extraction rules, utilizing their ability to abstract syntactical variants with the same relation. In addition, we divided the patterns for generalization, and used matching results for SVM learning. In experiments on extracting of protein-protein interactions, we obtained 71.8% precision and 48.4% recall on a small corpus and 64.0% precision and 83.8% recall estimated on a large text, which demonstrated the obvious advantages of our method.

Acknowledgement

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas "Systems Genomics" (MEXT, Japan) and Solution-Oriented Research for Science and Technology (JST, Japan).

References

R. Agrawal and R Srikant. 1995. Mining Sequential Patterns. In *Proc. the 11th International Conference on Data Engineering*, pages 3–14.

Christian Blaschke and Alfonso Valencia. 2002. The Frame-Based Module of the SUISEKI Information Extraction System. *IEEE Intelligent Systems*, 17(2):14–20.

Razvan Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *Proc. ACL'04*, pages 439–446.

Razvan C. Bunescu and Raymond J. Mooney. 2005. A Shortest Path Dependency Kernel for Relation Extraction. In *Proc. HLT/EMNLP 2005*, pages 724–731.

Razvan Bunescu and Raymond Mooney. 2006. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proc. ACL'04*, pages 423–429.

Yu Hao, Xiaoyan Zhu, Minlie Huang, and Ming Li. 2005. Discovering patterns to extract protein-protein interactions from the literature: Part II. *Bioinformatics*, 21(15):3294–3300.

Thorsten Joachims. 1999. Making Large-Scale SVM Learning Practical. In *Advances in Kernel Methods – Support Vector Learning*. MIT-Press.

G Joshi-Tope, M Gillespie, I Vastrik, P D'Eustachio, E Schmidt, B de Bono, B Jassal, GR Gopinath, GR Wu, L Matthews, S Lewis, E Birney, and Stein L. 2005. Reactome: a knowledgebase of biological pathways. *Nucleic Acids Research*, 33(Database Issue):D428–D432.

Asako Koike, Yoshiyuki Kobayashi, and Toshihisa Takagi. 2003. Kinase Pathway Database: An Integrated Protein-Kinase and NLP-Based Protein-Interaction Resource. *Genome Research*, 13:1231–1243.

Linguistic Data Consortium. 2005. ACE Program. http://projects.ldc.upenn.edu/ace/.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proc. AAI '94*.

Ivan A. Sag and Thomas Wasow. 1999. *Syntactic Theory*. CSLI publications.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. In *Proc. ACL 2003*, pages 224–231.

Tsujii Laboratory. 2005a. Enju - A practical HPSG parser. http://www-tsujii.is.s.u-tokyo.ac.jp/enju/.

Tsujii Laboratory. 2005b. GENIA Project. http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/.

U.S. National Library of Medicine. 2006. PubMed. http://www.pubmed.gov.

Akane Yakushiji, Yusuke Miyao, Yuka Tateisi, and Jun'ichi Tsujii. 2005. Biomedical information extraction with predicate-argument structure patterns. In *Proc. SMBM 2005*, pages 60–69.

Daming Yao, Jingbo Wang, Yanmei Lu, Nathan Noble, Huandong Sun, Xiaoyan Zhu, Nan Lin, Donald G. Payan, Ming Li, and Kunbin Qu. 2004. PathwayFinder: Paving The Way Towards Automatic Pathway Extraction. In *Bioinformatics 2004: Proc. the 2nd APBC*, volume 29 of *CRPIT*, pages 53–62.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proc. ACL'05*, pages 419–426.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proc. ACL'05*, pages 427–434.