Syntactic features for protein-protein interaction extraction

Rune Sætre*[1] , Kenji Sagae[1] and Jun'ichi Tsujii[1]

[1]Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Japan

Email: Rune Sætre*- rune.saetre@is.s.u-tokyo.ac.jp; Kenji Sagae - sagae@is.s.u-tokyo.ac.jp; Jun'ichi Tsujii - tsujii@is.s.u-tokyo.ac.jp;

*Corresponding author

Abstract

Background: Extracting Protein-Protein Interactions (PPI) from research papers is a way of translating information from English to the language used by the databases that store this information. With recent advances in automatic PPI detection, it is now possible to speed up this process considerably. Syntactic features from different parsers for biomedical English text are readily available, and can be used to improve the performance of such PPI extraction systems.

Results: A complete PPI system was built. It uses a deep syntactic parser to capture the semantic meaning of the sentences, and a shallow dependency parser to improve the performance further. Machine learning is used to automatically make rules to extract pairs of interacting proteins from the semantics of the sentences. The results have been evaluated using the AImed corpus, and they are better than earlier published results. The F-score of the current system is 69.5% for cross-validation between pairs that may come from the same abstract, and 52.0% when complete abstracts are hidden until final testing. Automatic 10-fold cross-validation on the entire AImed corpus can be done in less than one hour on a single server. We also present some previously unpublished statistics about the AImed corpus, and a short analysis of the AImed representation language.

Conclusions: We present a PPI extraction system, using different syntactic parsers to extract features for SVM with Tree Kernels, in order to automatically create rules to discover protein interactions described in the molecular biology literature. The system performance is better than other published systems, and the implementation is freely available to anyone who is interested in using the system for academic purposes. The system can help researchers quickly discover reported PPIs, and thereby increasing the speed at which databases can be populated and novel signaling pathways can be constructed.

Keywords: Protein-Protein Interaction (PPI), BioNLP, Natural Language Processing, Information Extraction (IE)

Background

The task of collecting relevant Protein-Protein Interactions (PPIs) from the thousands of new research papers published every day is too time consuming to be done manually, so automatic approaches using some form of

Natural Language Processing (NLP) are necessary. For example, someone searching the PubMed database for abstracts containing both the words "estrogen" and "cancer" could find 41.372 research paper references on November 12th 2007, and new papers are added on this topic every day. NLP can help reduce the burden on the human researcher, by automatically identifying which papers are truly about the relationship between estrogen and cancer, and which papers just coincidentally mention both of these terms. NLP can also be used to identify the exact locations in the papers that the human researcher should look closer at, and with the help of dictionaries, ontologies and inference mechanisms, relevant information can be found even if the author has used other synonyms instead of "estrogen" and "cancer" to describe the given relation. Many research groups are currently focusing on solving these problems, but the implications of using different parsers have not received much attention yet. In this research, we present a system that uses many existing components and combines them to make a complete working PPI detection system. Especially, we study how the use of a Head-Phrase Structure Grammar (HPSG) parser and a dependency parser helps the machine learning component extracting good rules for PPI discovery. To make direct comparisons to other published approaches possible, the results were evaluated using the AImed corpus. During the evaluation of the results, it became obvious that some standardization is needed to perform fair comparison between PPI systems in the future.

There are mainly two ways of doing evaluation on the AImed corpus in current literature. In most rule-based systems, 10% of the abstracts are kept secret and only used for testing. This can be done 10 times with a different 10% test-set every time, as long as the rule-generation is automatic, and only based on the training set. However, most of the published machine learning evaluations indirectly use the same abstract both for training and testing the systems. The results presented in this paper compare both these types of evaluations, and show that our results are better than other published systems in either case.

One way to help verify that an evaluation on a given corpus is correct is to publish the modified data on-line like Erkan et al. did [1]. Ideally, each biomedical corpus should have its own homepage where the users of the corpus could post corrections, modifications, and additions to the existing corpus. These homepages could then also be enhanced with automatic scoring services, tables of existing systems and comparisons. There are some pages like this already, for example for the LLL corpus [2], but this corpus is too small for machine learning systems, because they quickly run into data sparseness problems. Even AImed is quite a small corpus, with only 225 abstract, so bigger corpora are needed, with their own homepages to encourage competition, cooperation and further development or corrections of the corpus. During all the recent experiments with AImed corpus, some small inconsistencies and errors have been discovered. We have made a list of the corrections, to be posted on-line as a cleaned-up version of the corpus.

Related work

The biggest PPI extraction event arranged so far was the BioCreative2 challenge/workshop [3]. It gathered 26 research groups focusing on finding interacting proteins from MEDLINE abstracts, or from full text journal papers in PDF or HTML formats. Since the BioCreative PPI challenge did not separate the task of finding protein names and database identifiers from the task of finding interactions between the given proteins, the total performance scores were all below F=30%. This number cannot be directly compared to any of the published results on finding PPIs in the AImed corpus, since the training data provided was also very different. The AImed corpus is made of plain text abstracts, and human experts identified the protein names, and the exact locations of statements expressing PPI in the text. In the BioCreative training data, the exact protein names used in the text were not known, since only protein identifiers from the UniProt database were provided. Also, it was not clear which part of the full-text articles that actually describe the given protein interaction, and since the articles were provided in PDF and HTML formats only, some information can be lost in the process of recovering the plain text. This is especially true for protein names, which often contain special encoding characters, like the Greek letters α, β and κ.

Another major difference between BioCreative2 training data and the AImed corpus is that BioCreative only focused on PPI that is connected to experimental evidence in the text, without giving many examples of such evidence passages. In the AImed corpus, all PPI-sentences are annotated, so even though the

BioCreative setting may be more similar to the work of database curators, the training data is not complete enough to be used successfully in machine learning applications. Until more real examples can be provided in the form of complete PPI corpora, we focus on the smaller, but complete, AImed corpus. That being said, the system was also used, together with a protein name identifier, to produce well above average results for the BioCreative PPI task.

Many recent studies have used 10-fold cross-validation to compare their results with other studies using the AImed corpus [1,4–9]. Each of these papers report a different number of PPI pairs used from the AImed corpus (Table 3), making direct comparison between different systems very hard. These different numbers came from different ways of doing pre-processing and feature extraction. Unless proper care is taken, some of the interaction pairs in the corpus may be lost (or *new ones gained*) as illustrated in Figures 1 and 2 using two examples from a feature-set that was recently published on-line by Erkan et al [1].

Results and Discussion

The results from the different PPI experiments are given in Table 2a and 2b. Tables 3-5 provides some key numbers for people interested in the AImed corpus. And Table 6 shows the computational time required to run these experiments on one AMD64-bit processor with 32 GBytes of RAM. The discussion is presented in the following sub-sections.

The influence of syntactic features on PPI extraction

Tables 1 and 2 shows the different combinations of feature sets, and the corresponding results. F1 is our baseline, just using lemmatized word-features as Bag-Of-Words (BOW). F2 and F3 show that substituting the baseline BOW-features with features from one single parser, HPSG or Dependency, does not yield much improvement, but F6 shows that just the combined features from two parsers performs significantly better than the baseline. F2, F3 and F6 show the performance of parsing-features alone, but in F4, F5 and F7 we also kept the baseline BOW features. Now, F4 and F5 imply that adding either dependency or predicate-argument relations can contribute to improved PPI extraction performance, and F6 and F7 imply that their contributions are made in different ways. Dependency parsing is much faster than for example HPSG parsing, but can still improve the performance of PPI extraction. HPSG parsing is more expensive, but it also contributes to a bigger improvement compared to dependency parsing.

The Protein-Protein Interaction system

Another important result is the implementation of the PPI system itself. It brings together all the components from the PPI pipeline, including sentence splitting, part-of-speech (POS) tagging, protein name recognition, protein database-identifier mapping, syntactic parsing with different language models and machine learning with tree kernels, to perform the PPI extraction. To make all these systems work together, the stand-off notation defined by the UIMA project was adopted [10]. This allows us to keep the original text unchanged, and let each module in the pipeline add a new file with its own type of annotations. The new files only specify the range of characters that each tag is spannning in the original text file. The original text-file and the stand-off files and can easily be combined into an XML document, and operations like import, export, sort, merge and clip can be done very efficiently with a tool called the Stand-Off Manager (SOM).

The three most important inputs to the PPI module are the stand-off files with tags for Named Entities (proteins) and the syntactic tags from the two parsers we want to evaluate. In case of the AImed corpus, the protein tags are already provided, but when a prototype of this system was applied to the BioCreative2 PPI challenge data, the protein tags also had to be discovered. This was done with state-of-the-art precision using tools like MedT-NER [11], which already supported the UIMA stand-off (SO) format. As for parsing, we wanted to test two very different parsers, to see how their different outputs affect the performance of

machine learning on the AImed corpus. The first parser is a publicly available deep syntactic parser called Enju. It has been adapted to the domain of molecular biology by re-training it with the GENIA corpus [12], and it supports SO-output. The second parser is a word surface forms dependency parser made by Kenji Sagae [13].

The PPI module reads all the SO input-files, together with the original text file, and performs two tasks on the complete data. First, it extracts features from the parsers, and then it runs the support vector machine with tree kernels (SVM-TK [14, 15]) algorithm, either in learning, classification or cross-validation mode. Unless you want to do cross-validation, the data used for training and learning should be different. For example, the prototype of the system that was used in the BioCreative2 challenge was first trained on the AImed corpus abstracts, and the resulting support vectors were stored in a model file. Later, that model file was used to predict interactions in the BioCreative2 articles. Some post-processing was also necessary, since BioCreative only request the unique identifiers for the two interacting proteins, whereas our system outputs all the sentences where two protein names interact. Even though the prototype system used for BioCreative2 did not include the SVM-TK machine learning functionality, it achieved an F-score of 18.7% on the SwissProt IPS test set, making it the 6th best system among 16 participants.

In the cross-validation mode, the expected input is abstracts which contain protein and interaction tags. This input can be either in AImed style XML format, which will then be processed by the stand-off manager, or in the form of two files containing already processed plaintext and corresponding entities in the SO-format. The data is split into 10 groups with the same number of abstracts in each group, and then training and testing are done 10-fold, with one new group used for testing each time. The results from some of these cross-validations are presented in the next subsection.

Results from 10-fold cross-validation on AImed Corpus

There are at least two very different ways of doing 10-fold cross-validation using the AImed corpus. The experiments in [4, 5, 7] all split the abstracts into 10 groups before doing any feature extraction. In order to compare our results with them we did the same, and the results from this type of 10-fold cross-validations for all the systems are presented in Table 2a.

In other experiments, pre-processing and feature extraction was first done on the whole corpus, before randomly splitting the resulting pair-features into 10 groups, not minding which abstract the pair was originally coming from. These numbers will naturally be 10-20% higher, because a single sentence with more than one protein pair will usually produce many *similar* features that will be used both for training and testing the system. Some experiments also count multiple mentions of the same pair within one abstract as just one mention (see Figure 1). In this case the f-score will also be higher, since recall is perfect as soon as just one of the mentions have been discovered.

The second way of doing 10-fold cross-validation is very popular when using machine learning techniques. This is especially true when a parser is used to create the features for ML, since it is a waste of time to parse the same text 10 times, when the output will be the same every time. And the creation of features for each protein-pair will often be the same too, so these two tasks can often be done collectively for all the 225 abstracts in the corpus just one time. However, not all parts of the pre-processing can be done in this one-time fashion. For example, we included the 1000 most common lemmatized word forms as features, which mean that we need to count all the words in the corpus. This frequency count could be generated from the whole corpus during the one-time parsing, but that would mean that test data is used to create the training set. A better way is to generate a new frequency list for each of the 10 folds, based only on the 90% of the data that is available for training.

The results in Table 2b shows that the pair-level evaluation gives higher f-scores for our system, and presumably also for other systems using this evaluation style. The two last entries in this table (Mitsumori2006 and Yakushiji2006) are slightly different from the others, because they only counted two mentions of the same interaction pair in the same abstract one time, both during training and testing. We notice from Tables 2a and 2b that even though each parser alone cannot produce results much better than lemmatized word

features, the combination of features from two parsers, together with word features, is many percent-points better than previous state-of-the-art performance.

The final results are from timing the experiments, to see how expensive they are in terms of computational power. Table 6 shows the time used for the different parts of the pipeline.

Conclusions

We made a PPI extraction system, combining features from two parsers with very different language models, to extract good features for the tree kernels in SVM-light, in order to automatically create rules to discover protein interactions described in molecular biology literature. We experimented with features from a dependency parser, and from a HPSG parser. The results show that the combination of multiple parsers is very effective, and the resulting system performs better than the other published PPI systems. We also note that comparison between different systems is complicated by the fact that pre-processing in different systems removes different parts of the corpus, and that cross-validation without keeping test-abstracts completely hidden from the training phase produces performance scores 10-20% higher than what can be expected in a real-world application.

The implementation is freely available to anyone who might be interested in using this system for academic purposes. Because of the license agreement for SVM-TK, the system cannot be used for commercial purposes without a written license agreement. The system was first implemented in Perl, but later also translated to C++, and wrapped in Java as a UIMA component. Please contact the first author if you are interested in using the program, or download it directly from our homepage (http://www-tsujii.is.s.u-tokyo.ac.jp/∼satre/akane/). The C++ implementation of the system is tightly connected to the SVM-TK [14,15], but the Perl implementation can produce feature files in different formats compatible with many existing machine learning systems.

Methods

This section gives a more detailed description about the methods used to produce the results presented in this paper. The following subsections describe each pipeline step sequentially.

System Input

The input to the system can be plain text as in the PubMed database of research abstracts, or HTML versions of full papers provided on-line by many publishers like BioMed Central (BMC). In the case of HTML input, the text is converted to plain text by a separate module, mainly by stripping away all the HTML tags, and converting HTML entities to corresponding ASCII representations (e.g. the greek letter α is changed into the string "alpha"). In addition to the plain text, the system also needs to know the position of the proteins/genes mentioned in the text. This is called Named Entity Recognition (NER), and can be done by many freely available existing tools, like ABNER [16], MedT-NER [11] or PowerBioNE [17].

In the next subsections, we will use the following sentence (third sentence from abstract with PMID 10074428) as an example sentence, to explain what happens in each module in the pipeline system. The sentence, with its proteins and interacting pairs, is shown graphically in Figure 2, and duplicated here for easy reference:

We have identified a new TNF-related ligand , designated human GITR ligand (hGITRL) , and its human receptor (hGITR) , an ortholog of the recently discovered murine glucocorticoid-induced TNFR-related (mGITR) protein [4] .

Parsing

Enju is a Head-driven Phrase Structure Grammar (HPSG) parser, and it is capable of producing output in the SO-format [18]. This means that it does not modify the original input text. Instead, the output from the parser is stored in a separate file, with references to the positions of words, phrases and sentences in the original text file.

The dependency parser we experimented with was originally used for the CoNLL-X 2007 Shared Task [19]. Later it was re-trained to perform better in the biomedical domain, by using the GENIA Treebank corpus. Even though the dependency parser does not output stand-off format annotations, this can easily be created automatically from the CoNLL-X shared task format, as long as the (white-space) tokenization is done consistently.

Protein-Protein Interaction (PPI) detection

The stand-off files from NER and parsing are combined using a software component called the Stand-Off Manager (SOM). SOM is optimized to run in linear time, and it can do jobs like exporting/importing between XML and stand-off format. It also supports functions like sorting, merging, uniting and clipping specific tags from stand-off files. The combined tags from the SOM are processed by the PPI module in several stages, including creation of Predicate-Argument Structure (PAS) paths between all pairs of proteins within single sentences. These PAS-paths remove many of the different ways that natural language can be used to express the same fact, and creates a general representation of the relation between Protein1 and Protein2 for each pair in the sentence. We created such PAS-paths for all the co-occurring (within a sentence) protein pairs in the AImed corpus, and then we used machine learning to automatically create rules that say which parts of the patterns that best can separate interacting protein pairs from non-interacting pairs.

AImed Corpus

The AImed corpus was created by Bunescu and Mooney [20], and contains 177 Medline abstracts with interactions, and 48 abstracts without any PPI within single sentences (to create negative training examples). 25 of the "no interaction" abstracts are not supposed to contain interactions, while the remaining 23 should contain interactions (but probably not within a single sentence), and therefore they were not annotated by the creators of the AImed corpus.

The abstract files were cleaned to remove information like "PG-1377-AB", address-lines like "AD - ..." and citations (which should not be used in abstracts, since abstracts are often distributed without the corresponding reference list) like "[4]". The cleaned version of our example sentence is given in Figure 2.

Predicate Argument Structure (PAS) Paths

The PAS paths are constructed by finding the shortest path of PAS relations that connect Protein1 with Protein2 in the parse tree. Some examples showing the different feature-trees that were used for machine learning are presented in the *Features* subsection. The ML module itself is further described in the next sub-section.

Machine Learning

The ML tool used is called SVM-light with Tree-Kernels (SVM-TK) [14, 15]. Tree-kernels give us an easy way to represent the predicate argument structre or Dependency paths, or other syntactic features of the sentence. One feature-line can also be a combination of trees and classical SVM feature sets. The features extracted from interacting and non-interacting protein pairs are described in the following sub-section.

Features

The most basic feature set is simply "which words occur before, between and after the two protein names" of any pair. This idea has been used by many others already [4–6]. The set of words used are restricted to the 1000 most common words in the training part of the corpus, not including 322 common stop-words, like "and", "not", "a" etc. Figure 3 shows an example, where the SVM numbers (1:1 13:1 256:1 etc.) have been substituted with actual words to make it more readable. There are three vectors for each pair, corresponding to the "before, between and after" word-groups. Figure 3 also shows the shortest PAS and Dependency paths, extracted between the proteins in the interacting pair.

F-score

The comparisons between the different PPI systems are done based on the F-score measure. It is defined as $F = \frac{2*P*R}{P+R}$, where P (Precision) is the fraction of predicted pairs that are correct, and R (Recall) is the fraction of corpus pairs that were found.

List of abbreviations

BMC BioMed Central

BOW Bag-Of-Words

DEP Dependency

ID Identifier

HTML Hypertext Markup Language

SVM-TK Support Vector Machines with Tree Kernels

NEN Named Entity Normalization

NER Named Entity Recognition

PDF Portable Document Format

PPI Protein-Protein Interaction

Authors contributions

RS implemented the system and wrote the paper. KS provided dependency parsing input. JT supervised the research.

Acknowledgements

This work was supported by Grant-in-Aid for Scientific Research on Priority Areas, and Systems Genomics and Genome Network Project (MEXT, Japan). Prof. Yusuke Miayo and Dr. Jin-Dong Kim provided valuable input and guided the research. Takashi Tsunakawa, Kazuhiro Yoshida and prof. Takuya Matsuzaki helped solving some problems during the C++ implementation of the system. Fabio Rinaldi provided excellent feedback, together with two anonymous reviewers.

References

1. Erkan G, Ozgur A, Radev DR: **Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing**. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing (EMNLP '07)*, Prague, Czech Republic 2007.
2. Nédellec C: **Learning Language in Logic – Genic Interaction Extraction Challenge**. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*. Edited by Cussens J, Nédellec C, Bonn 2005:31–37, [http://www.cs.york.ac.uk/aig/lll/lll05/lll05-nedellec.pdf].
3. Krallinger M: **The Interaction-Pair and Interaction Method Sub-Task evaluation**. In *Proceedings of the Second BioCreative Challenge Workshop*. Edited by Hirschman L, Krallinger M, Valencia A 2007.
4. Bunescu RC, Mooney RJ: **Subsequence Kernels for Relation Extraction**. In *NIPS* 2005.
5. Mitsumori T, Murata M, Fukuda Y, Doi K, Doi H: **Extracting Protein-Protein Interaction Information from Biomedical Text with SVM**. *IEICE - Trans. Inf. Syst.* 2006, **E89-D**(8):2464–2466.
6. Giuliano C, Lavelli A, Romano L: **Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature**. In *EACL* [21] 2006.
7. Yakushiji A, Miyao Y, Tateishi Y, Tsujii J: **Biomedical Information Extraction with Predicate-Argument Structure Patterns**. In *the Proceedings of the First International Symposium on Semantic Mining in Biomedicine*, Hinxton, Cambridgeshire, UK 2005:60–69.
8. Yakushiji A, Yusuke M, Ohta T, Tateishi Y, Tsujii J: **Automatic Construction of Predicate-argument Structure Patterns for Biomedical Information Extraction**. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, Sydney, Australia 2006:284–292.
9. Katrenko S, Adriaans PW: **Learning Relations from Biomedical Corpora Using Dependency Trees**. In *KDECB*, *Volume 4366 of* Lecture Notes in Computer Science. Edited by Tuyls K, Westra RL, Saeys Y, Nowé A, Springer 2006:61–80.
10. Ferrucci D, Lally A: **Building an example application with the Unstructured Information Management Architecture**. *IBM Systems Journal* 2004, **43**(3):455–575.
11. Sætre R, Yoshida K, Yakushiji A, Miyao Y, Matsubyashi Y, Ohta T: **AKANE System: Protein-Protein Interaction Pairs in BioCreAtIvE2 Challenge, PPI-IPS subtask**. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*. Edited by Hirschman L, Krallinger M, Valencia A, Spain: CNIO 2007:209–212.
12. Hara T, Miyao Y, Tsujii J: **Evaluating Impact of Re-training a Lexical Disambiguation Model on Domain Adaptation of an HPSG Parser**. In *Proceedings of IWPT 2007*, Prague, Czech Republic 2007.
13. Sagae K, Tsujii J: **Dependency parsing and domain adaptation with LR models and parser ensembles**. In *the CoNLL 2007 Shared Task, Joint Conferences on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*, Prague, Czech Republic 2007.
14. Joachims T: *Advances in Kernel Methods: Support Vector Learning*, MIT Press 1999 chap. 11 - Making Large-scale SVM Learning Practical.
15. Moschitti A: **Making Tree Kernels Practical for Natural Language Learning.** In *EACL* [21] 2006.
16. Settles B: **ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text**. *Bioinformatics* 2005, **21**(14):3191–3192.
17. Zhou G, Zhang J, Su J, Shen D, Tan C: **Recognizing names in biomedical texts: a machine learning approach**. *Bioinformatics* 2004, **20**(7):1178–1190.
18. **Enju - A practical HPSG parser** [http://www-tsujii.is.s.u-tokyo.ac.jp/enju/].
19. Nivre J, Hall J, Kübler S, McDonald R, Nilsson J, Riedel S, Yuret D: **The CoNLL 2007 Shared Task on Dependency Parsing**. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* 2007:915–932, [http://www.aclweb.org/anthology/D/D07/D07-1096].
20. **"AImed" (Protein Interaction Corpus)** [ftp.cs.utexas.edu/pub/mooney/bio-data/interactions.tar.gz].
21. *EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, April 3-7, 2006, Trento, Italy*, The Association for Computer Linguistics 2006.

Figure 1: AImed sentence with protein pairs example 2: an example of AImed tagging the same entity name multiple times in one sentence, without indicating that it is the same name. In Erkan et al. [1] two of the interaction pairs are “missing” because of this fact.

Figures

Figure 1 - SentencePairs1.pdf

Figure 2 - SentencePairs2.pdf

Figure 3 - Features.pdf

Tables

Table 1 - Language features used for the machine learning

Table 1a - Single machine learning features

Feature	Description
After	The set of tokens after the last mention of the last protein in the pair
Before	The set of tokens before the first mention of the first protein in the pair
Between	The set of tokens between the end of the first protein and the beginning of the last protein in the pair
Head	The final semantic head token (head of head of...) is used to represent a constituent
Lemma	The base form of Words are used, e.g. singular form for nouns, and the infinite form for verbs (taken from Enju when possible, and using surface forms in case of parse failure)
DEP	Word-Dependencies from the Kenji Parser
PAS	Predicate Argument Structure information from Enju
Path	The smallest set of DEP or PAS head-lemmas connecting a pair of proteins

Table 1b - Feature sets used for machine learning

Feature set	Description
1 feature type	
F1	Word-features: Before, Between, After
F2	Path: Kenji Dependency between Enju Lemmas
F3	Path: Enju PAS between Enju Lemmas
2 feature types	
F4	Dependency Parsing + Word-features
F5	Enju + Word-features
F6	2 Paths: Enju & Dependency Parsing
All feature types	
F7	Enju & Dependency Parsing + Word-features

Figure 2: AImed sentence with protein pairs example 1: an example of the tagging done in the AImed Corpus and the corresponding automatic converted data from Erkan et al. (CLAIR) [1]: One protein name is changed, one protein name is deleted, and two of the interaction pairs are mixed up when the annotation is mapped into this format.

Word Features

AIMED PMID 10074428, Sentence 3, Pair 2
We have identified a new TNF-related ligand , designated human **GITR ligand** (**hGITRL**) , and its human receptor (**hGITR**) , an ortholog of the recently discovered murine **glucocorticoid induced TNFR-related (mGITR) protein** [4] .

Word Features	1000 most common lemmatized words (unordered, uncounted)
Before	**identify new ligand , designate human**
Between	**(PROT) , human receptor**
After	**) , recently murine (PROT protein**

AIMED PMID 10074428, Sentence 3, Non-Interacting Pair 3
We have identified a new TNF-related ligand , designated human **GITR ligand** (**hGITRL**) , and its human receptor (**hGITR**) , an ortholog of the recently discovered murine **glucocorticoid induced TNFR-related (mGITR) protein** [4] .

Word Features	1000 most common lemmatized words (unordered, uncounted)
Before	**identify new ligand , designate human**
Between	**(PROT) , human receptor recently murine**
After	**<empty>**

Predicate and Dependency Paths

AIMED PMID 10074428, Sentence 3, Pair 2
We have identified a new TNF-related ligand , designated human **GITR ligand** (**hGITRL**) , and its human receptor (**hGITR**) , an ortholog of the recently discovered murine **glucocorticoid induced TNFR-related (mGITR) protein** [4] .

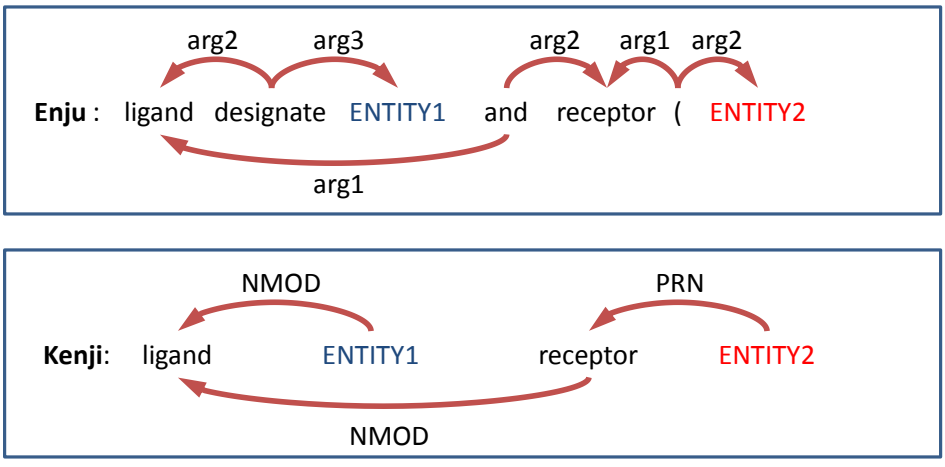

Figure 3: Word and Tree-Kernel features extracted from example sentence 1.

Table 2 - Comparison of different systems evaluated on AImed

Table 2a - Results with 10 separate groups of abstracts, and sentence/pair-level evaluation

10-fold cross-validation			
Features / Method	Prec	Rec	F-score
Words only			
F1 Words	55.7%	29.2%	37.8%
Parsers only			
F2 Kenji	67.6%	26.3%	37.1%
F3 Enju	**72.0%**	28.7%	41.0%
F6 K+E	68.9%	37.8%	48.6%
Words and Parsers combined			
F4 Kenji+W	59.9%	39.3%	46.9%
F5 Enju+W	61.3%	40.7%	48.6%
F7 K+E+W	64.3%	**44.1%**	**52.0%**
Bunescu2005 [4]	**73.9%**	35.2%	**47.7%**
Mitsumori2006 [5]	54.2%	**42.6%**	**47.7%**
Yakushiji2005 [7]	33.7%	33.1%	33.4%

Table 2b - Random sampling 10% of pairs, 10 times

False 10-fold cross-validation			
Method	Prec	Rec	F-score
Words only			
F1 Words	68.6%	48.0%	56.2%
Parsers only			
F2 Kenji	76.2%	35.6%	48.4%
F3 Enju	76.0%	39.7%	52.0%
F6 K+E	78.0%	51.6%	62.1%
Words and Parsers combined			
F4 Kenji+W	70.1%	53.1%	60.3%
F5 Enju+W	73.2%	54.6%	62.4%
F7 K+E+W	**78.1%**	**62.7%**	**69.5%**
Erkan2007 [1]	59.6%	60.7%	**60.0%**
Giuliano2006 [6]	60.9%	57.2%	59.0%
Katrenko2007 [9]	45.0%	**68.4%**	54.3%
Mitsumori2006 [5]	55.7%	53.6%	54.3%
Yakushiji2006 [8]	**71.8%**	48.4%	57.3%

Table 3 - Interacting/total number of pairs reported in different papers

AImed Protein Pairs in different studies		
Experiment	Positive	Total
AImed from FTP	1071	5631
This study	1068	5631
Erkan2007 [1]	951	4026
Katrenko2006 [9]	1006	5106
Mitsumori2006 [5]	1107	5476

Table 4 - Distribution of pairs among the AImed abstracts

Pairs per Abstract		
$\frac{Pair}{Abstract}$	$\mid Abstracts \mid$	$\mid Pairs \mid$
0	48	0
0,5	(3)	3
1	12	12
2	26	52
3	18	54
4	18	72
5	12	60
6	24	144
7	17	119
8	12	96
9	11	99
10	5	50
11	4	44
12	4	48
13	4	52
14	3	42
15	2	30
16	1	16
17	3	51
27	1	27
Sum	225	1071

Table 5 - Facts about the AImed corpus

AImed from FTP (the numbers)	
Property	Value
Total Lines	2202
AD - tags	225
AD - lines	222
(Missing AD sentence-splits)	3
Extra AD sentence split lines	17
Total Address lines	239
Remaining lines	1963
Title lines	225
Abstract-body sentences	1738
Missing sentence-splits	15
Extra sentence-splits	7
Total Titles and Sentences	1971

Table 6 - Pipeline processing time

Processing time for different sub-modules	
Task	Run-time (minutes)
Pre-processing Corpus files	1
Dependency parsing, Kenji	4
HPSG parsing, Enju	27
Feature Extraction	1
10-fold SVM cross-validation	10
Total 10-fold cross-validation	43