DEEPER: a Full Parsing based Approach to Protein Relation Extraction

Timur Fayruzov[1], Martine De Cock[1], Chris Cornelis[1], Véronique Hoste[2]

[1] Department of Applied Mathematics and Computer Science
Ghent University Association
Krijgslaan 281 (S9), 9000 Gent, Belgium
{Timur.Fayruzov, Martine.DeCock, Chris.Cornelis}@UGent.be
[2] School of Translation Studies
Ghent University Association
Groot-Brittanniëlaan 45, 9000 Gent, Belgium
Veronique.Hoste@hogent.be

Abstract. Lexical variance in biomedical texts poses a challenge to automatic protein relation mining. We therefore propose a new approach that relies only on more general language structures such as parsing and dependency information for the construction of feature vectors that can be used by standard machine learning algorithms in deciding whether a sentence describes a protein interaction or not. As our approach is not dependent on the use of specific interaction keywords, it is applicable to heterogeneous corpora. Evaluation on benchmark datasets shows that our method is competitive with existing state-of-the-art algorithms for the extraction of protein interactions.

1 Introduction

Studying the interactions of proteins is an essential task in biomedical research, so it comes as no surprise that a lot of effort is being devoted to the construction of protein interaction knowledge bases. More and more relevant information is becoming available on the web, in particular in literature databases such as MEDLINE[3], in ontological resources such as the Gene Ontology[4], and in specialized structured databases such as IntAct[5]. The unstructured information in scientific publications poses the biggest challenge to biologists who are interested in specific gene or protein interactions, as they are forced to spend a tremendous amount of time reviewing articles looking for the information they need. Structured knowledge bases are easier to query, but again require a great deal of knowledge and labour intensive maintenance to stay synchronized with the latest research findings in molecular biology. Automation tools can facilitate this task, which is why machine learning techniques for information extraction (IE) in the biomedical domain have gained a lot of attention over the last years.

[3] `http://www.ncbi.nlm.nih.gov/`
[4] `http://www.geneontology.org/`
[5] `http://www.ebi.ac.uk/intact/site/index.jsf`

Relation extraction from texts is one of the most difficult tasks of IE. In natural language, relations can be expressed in different ways, hence no universal set of rules or patterns for mining them can be constructed. Traditional algorithms for relation learning from texts can perform reasonably well (see e.g. [1, 2, 5, 12]), but they typically rely explicitly or implicitly on specific interaction keywords, which limits their applicability to heterogeneous data. The biggest obstacle with heterogeneous datasets is that they describe protein interactions using different lexicons. However, entirely different surface representations for interactions can still exhibit the same syntactic pattern. We therefore propose to abstract from pure lexical data and to concentrate only on more general language structures such as parsing and dependency information. This coarser grained approach allows to cope better with the lexical variance in the data. Indeed, taking the fact into account that lexically different expressions of protein interactions might still bear some resemblance on the syntactic level provides welcome hints for machine learning techniques that commonly thrive on similarities in the data.

The resulting system is a mining tool that facilitates information extraction and knowledge base maintenance by presenting to the user protein interactions identified in scientific texts. The tool aims at supporting biologists in finding relevant information, rather than to exclude them entirely from the data processing flow. After reviewing related approaches in Section 2, we give a detailed description of the proposed method in Section 3. Abstracting from pure lexical data and only relying on syntactic patterns instead bears the risk of overgeneralizing, in the sense that sentences that do not describe protein interactions might exhibit a syntactic structure similar to those that do, and hence they might get incorrectly identified as protein interactions. To verify the reliability of our approach we therefore evaluated it on two benchmark datasets. The experimental results and a comparison with existing algorithms are described in Section 4. Concluding remarks and future work are presented in Section 5.

2 Related Work

The extraction of protein relations has attracted a lot of attention during the last years, resulting in a range of different approaches. The first step is the recognition of the protein names themselves (see e.g. [3, 6, 15]). As the focus of this paper is on the mining of *interactions*, we assume that protein name recognition has already taken place. The recognition of protein interactions is typically treated as a classification problem: the classifier gets as input information about a sentence containing two protein names and decides whether the sentence describes an actual interaction between those proteins or not. The classifier itself is built manually or, alternatively, it is constructed automatically using an annotated corpus as training data. The different approaches can be distinguished based on the information they feed to the classifier: some methods use only shallow parsing information on the sentence while others exploit full parsing information.

Shallow parsing information includes part-of-speech (POS) tags and lexical information such as lemmas (the base form of words occuring in the sentence) and orthographic features (capitalization, punctuation, numerals etc.). In [2], a support vector machine (SVM) model is used to discover protein interactions. In this approach each sentence is split into three parts — before the first protein, between the two proteins and after the second protein. The kernel function between two sentences is computed based on common sequences of words and POS tags. In another approach [5], this kernel function is modified to treat the same parts of the sentence as bags-of-words and called a global context kernel. It is combined with another kernel function called a local context kernel, that represents a window of limited size around the protein names and considers POS, lemmas, and orthographic features as well as the order of words. The resulting kernel function in this case is a linear combination of the global context kernel, and the local context kernel.

A completely different approach is presented in [12], where very high recall and precision rates are obtained by means of hand-crafted rules for sentence splitting and protein relation detection. The rules are based on POS and keyword information, and they were built and evaluated specifically for *Escherichia coli* and yeast domains. It is questionable, however, whether comparable results could be achieved in different biological domains and how much effort would be needed to adapt the approach to a new domain. In another approach reported on in [8], a system was built specifically for the LLL challenge (see Section 4). First, training set patterns are built by means of pairwise sentence alignment using POS tags. Next, a genetic algorithm is applied to build several finite state automata (FSA) that capture the relational information from the training set.

Besides the methods described above, approaches have been proposed that augment shallow parsing information with full parsing information, i.e. syntactic information such as full parse and dependency trees. In [4] for instance, for every sentence containing two protein names a feature vector is built containing terms that occur in the path between the proteins in the dependency tree of the sentence. These feature vectors are used to train an SVM based classifier with a linear kernel. More complex feature vectors are used in [10], where the local contexts of the protein names, the root verbs of the sentence, and the parent of the protein nodes in the dependency tree are taken into account by a BayesNet classifier. In [7], syntactic information preprocessing, hand-made rules, and a domain vocabulary are used to extract gene interactions. The approach in [17] uses predicate-argument structures (PAS) built from dependency trees. As surface variations may exhibit the same PAS, the approach aims at tackling lexical variance in the data. It is tailored towards the AImed dataset (see Section 4) for which 5 classes of relation expression templates are predefined manually. The classes are automatically populated with examples of PAS patterns and protein interactions are identified by matching them against these patterns.

To the best of our knowledge, all existing work either uses only shallow parsing information (including lexical information) or a combination of shallow and full parsing information. Furthermore, approaches of the latter kind typically use dependency trees only as a means to e.g. detect chunks or to extract relevant keywords. The goal of this paper is to investigate what can be achieved using *only* full parsing information. In other words, the full parsing information is not used as a means to select which further (lexical) information to feed to the classifier, but it is used as a direct input itself to the classifier. The fact that such an approach is independent of the use of a specific lexicon makes it worthwhile to investigate.

3 DEEPER: a Dependency and Parse Tree based Relation Extractor

There is an abundance of ways in English to express that proteins stimulate or inhibit one another, and the available annotated corpora on protein interactions cover only a small part of them. In other words, when the interaction mining tool is confronted with a previously unseen text, it is likely for this text to contain protein interactions described in ways for which there is no exact matching example in the training data. However, different surface representations can still exhibit a similar syntactic pattern, as the following example illustrates.

Example 1. Consider the following sentences about the interaction between *sigma F* and *sigma E* in one case and between *GerE* and *cotB* in the other case:

> Sigma F activity regulates the processing of sigma E within the mother cell compartment.
>
> A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.

Although the surface representations are very different, the underlying syntactic pattern, which represents part of a dependency tree, is the same in both cases:

$$protein \xleftarrow{nn} noun \xleftarrow{nsubj} verb \xrightarrow{dobj} noun \xrightarrow{prep\_of} protein$$

We exploit this deeper similarity between training instances by using dependency and parsing information to build abstract representations of interactions. Such representations have less variance than the initial lexical data, hence sensible results can be obtained from smaller training datasets. The approach is fully automatical and consists of three stages: after a text preprocessing stage, for every sentence containing two tagged protein names, we construct a feature vector summarizing relevant information on the parse tree and the dependency tree. In the third stage a classifier is trained to recognize whether the sentence describes an actual interaction between the proteins. The novelty of the approach w.r.t. existing work is that we do not use dependency data to detect keywords, but we consider dependencies as features themselves. In the next section we show that using only this kind of syntactic information without any lexical data allows to obtain reasonably good results.

Text preprocessing This step is intended to simplify the sentence structure and hence increase the parser reliability. It includes sentence splitting as well as the detection and the substitution of complex utterances (e.g. chemical formulas or constructions with many parentheses) with artificial strings, which in some cases can otherwise significantly reduce the quality of parsing. Furthermore, we expand repeating structures, turning e.g. 'sigA- and sigB-proteins' or 'sigA/sigB-proteins' into 'sigA-proteins and sigB-proteins'. All substitutions are done automatically by means of regular expressions; hence the same kind of preprocessing can be applied to an arbitrary text. Moreover, tagged protein names in the text may include more than one word; in order to treat them as a single entity in further processing stages, we replace them in the same manner as formulas. Finally, we take all possible pairwise combinations of proteins in each sentence and create one sentence for each combination where only this combination is tagged. Part of this process is shown in Example 2.

Example 2. Below is the result after text preprocessing for the second sentence from Example 1:

> A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.
>
> ...
>
> A low GerE concentration, which was observed during the experiment, activated the transcription of cotB by final sigmaK RNA polymerase, whereas a higher concentration was needed to activate transcription of cotX or cotC.

Feature vector construction After the text preprocessing stage, for each sentence we build a feature vector that summarizes important syntactic information on the parse tree and the typed dependency tree, which are both ways of representing sentence structure. A parse tree is a tree (in terms of graph theory) that represents the syntactical structure of a sentence. Words from the sentence are leaves of the parse tree and syntactical roles are intermediate nodes, so a parse tree represents the nesting structure of multi-word constituents. A dependency tree on the other hand represents interconnections between individual words of the sentence. Hence, all nodes of the dependency tree are words of the sentence, and edges between nodes represent syntactic dependencies. In typed dependency trees, edges are labeled with syntactic functions (e.g., subj, obj). Figure 1 depicts the typed dependency tree and parse tree for the first sentence of Example 1.

During the feature extraction phase we parse each sentence with the Stanford Parser[6]. For each tagged pair of proteins (recall that each sentence has only one such pair), we extract a linked chain [14] from the dependency tree. The dependency tree is unordered w.r.t. the order of the words in the sentence; hence to produce patterns uniformly, we order the branches in the linked chain based

[6] `http://nlp.stanford.edu/downloads/lex-parser.shtml`

Fig. 1. Dependency and parse trees and linked chain for the first sentence of Ex. 1.

on the position of the words in the initial sentence. Thus the left branch contains the word that occurs earlier in the sentence and the right branch the word that occurs later. The absolute majority of the branches in the linked chains from the datasets we examined contain no more than 6 edges, and those which contain more are negative instances, so we choose feature vectors with 6 features for each branch to cover all positive examples from the training set. Therefore, we use the first 6 dependencies from the left branch as the first 6 features in the feature vector. Likewise, the first 6 dependencies from the right branch correspond to features 7 through 12. Moreover, to better describe the structure of the relation we incorporate information from the parse tree as well, namely the length of the path from the root of the parse tree to each protein as the 13th and the 14th feature, and the number of nested subsentences in these paths as the 15th and the 16th feature.

Example 3. Below is the feature vector of the first sentence from Example 1:

nsubj	*nn*					*dobj*	*prep_of*					4	7	0	0

We extract a linked chain between the two proteins, as shown in Figure 1. It is already ordered, i.e. *Sigma F* precedes *Sigma E* in the sentence, so we do not need to swap these branches. We take the left branch and fill the first 6 features of the feature vector. As the branch contains only 2 dependencies — *nsubj* and *nn*, 4 slots in the vector remain empty. Features 7-12 for the right branch are filled in the same manner. *Sigma F* is at depth 4 in the parse tree while *Sigma E* is at depth 7, and the parse tree in Figure 1 does not contain subsentences. All this information is reflected in the last 4 features of the vector. Note that the resulting feature vector contains several empty fields; only the most complicated sentences will have a value for each feature in the vector.

Training a classifier By the above process, we obtain a set of feature vectors for sentences which can be divided into two classes — those that describe real protein interactions and those that do not. Therefore, we can use a standard classification algorithm to distinguish between these two classes. To build the classifier, we use a decision tree algorithm (C4.5 implementation [13]) and the BayesNet classifier [9]. These two algorithms represent classical instances of two branches of machine learning — rule induction and statistical learning — which employ different approaches to data processing. Decision trees consist of internal nodes which represent conditions on feature values, and leaves which represent classification decisions that conform to the feature values in nodes on the way to this leaf. The BayesNet classifier is represented as a directed graph with a probability distribution for each feature in the nodes and with the edges denoting conditional dependencies between different features. When we use the BayesNet classifier we apply a conditional independence assumption, which means that probabilities of node values depend only on probabilities of values of their immediate parents, and do not depend on higher ancestors. This corresponds to the reasonable assumption that the syntactic role of a node in the linked chain depends on the syntactic role of its parent only.

To overcome the problem of missing values (which occur frequently in the feature vectors), in the BayesNet classifier we simply change them by a default value. With C4.5, to classify an instance that have a missing value for a given node, the instance is weighted proportionally to the number of instances that go down to each branch, and recursively processed on each child node w.r.t. to assigned weight. This process is described in more detail in [16].

4 Experimental Evaluation

To verify the reliability of our approach, we evaluated it on two datasets. The first dataset [11] originates from the Learning Language in Logic (LLL) relation mining challenge on Genic Interaction Extraction[7]. This dataset contains annotated protein/gene interactions concerned with *Basilicus subtilis* transcription. The sentences in the dataset do not make up a full text, but they are individual sentences taken from several abstracts retrieved from Medline. The proteins involved in the interactions are annotated with agent and target roles; because our current approach is not aimed at mining the direction of interactions, we ignore this annotated information and treat the interactions as symmetrical relations.

[7] `http://genome.jouy.inra.fr/texte/LLLchallenge/`

The AImed dataset [1] is compiled from 197 abstracts extracted from the Database of Interacting Proteins (DIP) and 28 abstracts which contain protein names but do not contain interactions. Since we are interested in retrieving protein interactions, in this paper we use only the former set of abstracts. The connection between a full name of a protein and its abbreviation, e.g. *tumor necrosis factor (TNF)*, is annotated as an interaction in the AImed dataset. Since such an annotation is not concerned with an actual interaction between different proteins, we omit this kind of data from our experiments. Furthermore we removed nested protein annotations, which wrap around another protein or interaction annotation. Finally, TI- and AD- sections as well as PG- prefixes, which are Medline artifacts, were removed.

More information about the datasets is listed in Table 1. From this table, it is clear that the AImed dataset is highly imbalanced, as there is a strong bias to negative examples. To the best of our knowledge, these are the only two publicly available datasets containing annotations of protein interactions and hence suitable to evaluate our approach. In the evaluation we used 10-fold cross validation for both the AImed and the LLL05 dataset; furthermore we ran experiments with AImed as training set and LLL05 as test set. We used Weka [16] for the implementation of the machine learning methods.

Table 1. Datasets used in the experiments

Dataset	# sentences	# positive instances	# negative instances
AImed	1978	816	3204
LLL'05	77	152	233

The difference in the datasets requires different parameters to achieve optimal performance. As we have mentioned above, the AImed dataset is imbalanced and using it for training tends to lead to a bias towards classifying examples as negative (independently of the training scheme). For this reason, we use cost-sensitive learning [16] to decrease the bias when AImed is used as a training set. Moreover, in the C4.5 implementation for the AImed dataset, we build a binary decision tree, i.e. at each node the algorithm tests only one value of one feature. Otherwise, the algorithm would decide that the empty tree classifies the dataset in the best way, and all examples would be classified as negative (again, because of the biased dataset).

The results below are described in terms of the sentences that are (in)correctly identified by the system as describing a protein interaction, as these are exactly the instances that the system will present to the biologist. The *relevant instances* are the sentences that should have been identified as describing protein interactions; this includes the true positives, i.e. the positive instances that are correctly identified by the system, but also the false negatives, i.e. the positive instances that are overlooked by the system. The *retrieved instances* are the sentences that are identified by the system as describing protein reactions. This includes the true positives but may also include false positives, i.e. sentences incorrectly identified by the system as describing a protein interaction. Using TP, FN, and FP to denote the number of true positives, false negatives, and false positives respectively, recall and precision are defined as

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (also referred to as coverage) indicates how many of the relevant instances are retrieved. Precision (also referred to as accuracy) indicates how many of the retrieved instances are relevant.

To study the trade-off between recall and precision we use a confidence threshold p between 0 and 1 such that an instance is retrieved iff the classifier has a confidence of at least p that the instance describes a real protein interaction. The BayesNet classifier provides such a confidence value naturally, because its output is a class distribution probability for each instance. Decision trees can also be easily adapted to produce a probabilistic output by counting training examples at the leaf nodes. If a sentence that is being classified ends up at a leaf node, the confidence of the classifier that it is a positive instance, is the proportion of positive training examples to all training examples at that leaf node. When p is set to 1, a sentence is only retrieved if the classifier has absolute confidence that it describes a protein interaction. In this case typically the precision is high while the recall is low. Decreasing the threshold p allows to increase the recall at the cost of a drop in the precision. Figure 2 shows recall-precision curves obtained by varying p from 1 to 0.

As the first picture depicts, both classifiers allow to obtain similarly nice results for the LLL05 dataset, which is a first indication that we can make reasonable predictions about the occurrence of protein interactions in sentences based solely on full parsing information. Several authors present results of their protein relation extraction methods on the LLL05 dataset. However, since our current approach is not aimed at identifying agent and target roles in the interactions, we can only compare our results with those methods that treat the interactions as symmetrical relations. The first picture in Figure 2 shows a result from [7] depicted by a $*$ and corresponding to a recall of 85% and a precision of 79%. One should keep in mind that the method from [7] uses hand-made rules and a domain vocabulary, while our approach does not employ any prespecified knowledge. However, results show that our approach with a C4.5 classifier achieves results which are very close to the ones obtained by RelEx.

Whereas the LLL05 dataset contains only selected sentences from Medline abstracts, the AImed dataset contains full abstracts, posing a bigger challenge to our approach. The second picture in Figure 2 shows that C4.5 and BayesNet allow to obtain comparable results in terms of recall and precision. They both outperform the PAS-approach for which a recall of 33.1% for a precision of 33.7% is reported in [17].

Finally, we performed a cross dataset experiment using the AImed dataset for training the classifier and the LLL05 dataset for testing. The corresponding

Fig. 2. Recall-precision charts

recall-precision curves for C4.5 and the BayesNet classifier are shown in the third picture in Figure 2. While both datasets are independent (built for different biological subdomains and by different people), our approach shows good results. This indicates that the current approach is applicable to different domains without alterations, although further investigation is needed to back up this claim. The third picture also shows the recall-precision curve for the subsequence kernel method from [2] which is a state-of-the-art shallow parsing based approach for relation extraction. Since the approach was evaluated on a different dataset in [2], we used the implementation provided by the authors[8] and our datasets to perform the experiment. The training is done with LibSVM[9] on the AImed dataset and testing is done on the LLL05 dataset. The results show that the three methods are comparable, with a slight preference for our approach with the BayesNet classifier, as it can keep up a very high precision of 84% for a recall of up to 60%.

5 Conclusions and Future Work

Whereas existing approaches for protein interaction detection typically rely on shallow parsing information, sometimes augmented with full parsing information, we presented an approach based solely on full parsing information. More in particular, we proposed a clean and generally applicable approach in which for each sentence a feature vector is constructed that contains 12 features with information on the dependency tree and 4 features with information on the parse tree of the sentence. Next we fed these feature vectors as inputs to a C4.5 and a BayesNet classifier, as representatives of a rule induction and a statistical learning algorithm. Using these standard data mining algorithms and no shallow parsing or lexical information whatsoever, we were able to obtain results which are comparable with state-of-the-art approaches for protein relation mining. This result is promising since a method that uses only full parsing information does not depend on specific interaction keywords and is less affected by the size and/or the heterogenity of the training corpus.

As this paper presents work in progress, quite some ground remains to be covered, including a more complete comparison with existing methods. Among other things, it would be interesting to build an SVM model with our feature vectors and compare the results with those of shallow and combined parsing based approaches that rely on kernel methods as well. Furthermore, we intend to look into detecting the agents and the targets of interactions, which would allow us to do an independent evaluation on the LLL05 dataset as intended by the LLL challenge. A final intriguing question is whether an augmentation with shallow parsing information could increase the performance of our approach.

[8] `http://www.cs.utexas.edu/~razvan/code/ssk.tar.gz`
[9] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Acknowledgement

Chris Cornelis would like to thank the Research Foundation–Flanders for funding his research.

References

1. R. Bunescu, R. Ge, J.R. Kate, M.E. Marcotte, R.J. Mooney, K.A. Ramani and W.Y.Wong *Comparative Experiments on Learning Information Extractors for Proteins and their Interactions*, Artificial Intelligence in Medicine 33(2), 2005.
2. R.Bunescu and J.R.Mooney *Subsequence Kernels for Relation Extraction*, in: Proc. 19th Conf. on Neural Information Processing Systems (NIPS), 2005.
3. N.Collier, C.Nobata and J.Tsujii. *Extracting the Names of Genes and Gene Products with a Hidden Markov Model*, in: Proc. 17th Int. Conf. on Computational Linguistics, 2000.
4. G. Erkan, A. Ozgur and D.R. Radev *Extracting Interacting Protein Pairs and Evidence Sentences by using Dependency Parsing and Machine Learning Techniques*, in: Proc. 2nd BioCreAtIvE Challenge Workshop — Critical Assessment of Information Extraction in Molecular Biology, 2007.
5. C. Giuliano, A. Lavelli and L. Romano *Exploiting Shallow Linguistic Information for Relation Extraction From Biomedical Literature*, in: Proc. 11th Conf. of the European Chapter of the Association for Computational Linguistics (EACL 2006).
6. K.Fukuda, A.Tamura, T.Tsunoda, T.Takagi *Toward Information Extraction: Identifying Protein Names from Biomedical Papers*, Proc. Pacific Symp. on Biocomputing, 1998.
7. K. Fundel, R. Küffner, and K. Zimmer *RelEx—Relation extraction using dependency parse trees*, Bioinformatics 23(3) 2007.
8. J. Hakenberg, C. Plake, U. Leser, H. Kirsch and D. Rebholz-Schuhmann *LLL'05 Challenge: Genic Interaction Extraction — Identification of Language Patterns Based on Alignment and Finite State Automata*, in: Proc. ICML05 Workshop: Learning Language in Logic, 2005.
9. F.V. Jensen. *An Introduction to Bayesian Networks* Springer, 1996.
10. S. Katrenko and P. Adriaans *Learning Relations from Biomedical Corpora Using Dependency Tree Levels*, in: Proc. BENELEARN conference, 2006
11. C. Nédellec *Learning Language in Logic - Genic Interaction Extraction Challenge*, Proc. ICML05 Workshop: Learning Language in Logic.
12. T. Ono, H. Hishigaki, A. Tanigami and T. Takagi *Automated Extraction of Information on Protein-Protein Interactions from the Biological Literature*, Bioinformatics, 17(2) 2001.
13. J.R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Francisco, 1993.
14. M. Stevenson and M.A. Greenwood. *Comparing Information Extraction Pattern Models*, in: Proc. IE Beyond The Document Workshop (COLING/ACL 2006).
15. L.Tanabe and W.J.Wilbur. *Tagging Gene and Protein Names in Biomedical Text*, Bioinformatics, 18(8), 2002.
16. I.H. Witten and E. Frank *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.
17. A. Yakushiji, Y. Miyao, Y. Tateisi and J. Tsujii *Biomedical Information Extraction with Predicate-Argument Structure Patterns*, in: Proc. 1st Int. Symp. on Semantic Mining in Biomedicine, 2005.