A Tree Kernel-Based Method for Protein-Protein Interaction Mining from Biomedical Literature

Jae-Hong Eom, Sun Kim, Seong-Hwan Kim, and Byoung-Tak Zhang

Biointelligence Laboratory,
School of Computer Science and Engineering,
Seoul National University, Seoul 151-744, South Korea
{jheom, skim, shkim, btzhang}@bi.snu.ac.kr

Abstract. As genomic research advances, the knowledge discovery from a large collection of scientific papers becomes more important for efficient biological and biomedical research. Even though current databases continue to update new protein-protein interactions, valuable information still remains in biomedical literature. Thus data mining techniques are required to extract the information. In this paper, we present a tree kernel-based method to mine protein-protein interactions from biomedical literature. The tree kernel is designed to consider grammatical structures for given sentences. A support vector machine classifier is combined with the tree kernel and trained on predefined interaction corpus and set of interaction patterns. Experimental results show that the proposed method gives promising results by utilizing the structure patterns.

1 Introduction

Since protein-protein interactions play key roles in various biological processes [1], detail analysis of these interactions would significantly contribute to the understanding of the biological phenomena. As genomic research advances, the knowledge discovery from a large collection of scientific papers becomes more important to support biological and biomedical research. Thus, how to extract protein interactions from biomedical literature has been an active research subject over recent years.

There are many accomplishments in literature data mining for biological data analysis and in most cases they focus on protein interaction extraction. But, the protein interaction data is still accumulated manually in biological databases. Furthermore, scientists sometime continue to publish their discoveries on new protein interactions and modifying previous results in scientific papers without submitting to the public databases [2]. Therefore, a lot of interaction data still exist only in text materials.

Protein interaction extraction systems widely adopt natural language processing (NLP) techniques. The NLP approaches can be regarded as parsing-based methods and both full and shallow parsing strategies have been performed in previous studies. Yakushiji et al. [3] used a general full parser with grammars for biomedical domain to extract interaction events by filling sentences into augmented structures. Park et al. [4] proposed bidirectional incremental full parsing with combinatory categorical grammar (CCG) which localizes target verbs and then scans the left and right neighborhood of the verb respectively to find interaction events in the sentences. Temkin et al. [5] also utilized a lexical analyzer and context-free grammar (CFG) to extract gene, protein, and small molecule interactions with recall rate of 63.9% and precision rate of 70.2%. Similarly, preposition-based parsing to generate templates also proposed by Leroy et al. [6] and they achieved precision of 70% for biomedical literature abstract processing. For a partial parsing method, Pustejovsky et al. [7] used the relational parsing for the inhibition relation with recall rate of 57%. But, these methods are inherently complicated, requiring many resources, and the performance is not satisfactory as yet.

In this paper, we extract protein-protein interactions from biomedical literature using a tree kernel-based method, which utilizes grammatical structure of sentences directly. A support vector machine (SVM) with the tree kernel is used to discriminate interaction and non-interaction data from predefined interaction corpus and set of interaction patterns. The proposed approach exploits part-of-speech (POS) tags and text structures to improve extraction performance.

Here, we address how the tree kernel can be used to extract protein interactions and how to the extraction performance can be further improved. This paper is organized as follows. In Section 2, the basic concept of kernel method and its types are described. The tree kernel for protein-protein interaction extraction is explained in Section 3. In Section 4, we show the experimental results of protein interaction extraction. Finally, in Section 5, we present concluding remarks and draw future directions.

2 Kernel Method

An object can be transformed into a collection of features $f_1, \ldots, f_N$, which produce N-dimensional feature vectors. However, it is difficult to express data via features. For example, feature-based representations in NLP problems produce inherently local representations of objects and it is computationally infeasible to generate features involving long-range dependencies.

Kernel methods are an attractive alternative of feature-based approaches. Kernel methods retain the original representation of objects and use the object in algorithms only via computing a kernel function between a pair of objects. A kernel function is a similarity function which has certain properties. That is, kernel function K over the object space X is binary function $K : X \times X \rightarrow [0, \infty]$ mapping a pair of objects $x, y \in X$ to their similarity score $K(x, y)$. This is embedding procedure of data items (e.g. genes, proteins, molecular compounds, etc.) into a vector space F, called feature space, and searching for linear relation in the feature space. This embedding is defined implicitly, by specifying an inner product for the feature space via a symmetric and positive semidefinite kernel function: $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$, where $\Phi(x)$ and $\Phi(y)$ are the embeddings of data items, x and y [8].

Kernel functions implicitly calculate the inner product of feature vectors in high-dimensional feature spaces by projecting all objects from their original low-dimensional spaces. That is, there exist features $f(\cdot) = (f_1(\cdot), f_2(\cdot), \ldots)$, $f_i : X \to R$, so that $K(x, y) = \langle f(x), f(y) \rangle$. Conversely, given features $f(\cdot) = (f_1(\cdot), f_2(\cdot), \ldots)$, a function defined as a dot product of the corresponding feature vectors is necessarily a kernel function [8].

In many cases, it is possible to compute the dot product of certain features without enumerating all the features. One good example is the subsequence kernels. In the subsequence kernels, the inputs are strings of characters, and the kernel function computes the number of common subsequences between two strings, where each subsequence match is additionally decreased by the factor reflecting how spread out the matched subsequence in the original sequences [9]. Despite of an exponential increase in number of features (subsequences), it is possible to compute kernel matrix in polynomial time. Therefore, one can exploit long-range features without enumerating the features explicitly.

There are a number of learning algorithms that operate only by using the dot product of examples. The models produced by the learning algorithms are also expressed by dot product of examples. Substituting a particular kernel functions in place of dot product defines a specific instantiation of such learning algorithms. The algorithms which process examples only via computing their dot products are sometimes called dual learning algorithms. The SVM [10] is known as the learning method that not only allows for a dual formulation, but also provides a rigorous rationale for resisting over-fitting. For the kernel-based algorithms working in extremely rich feature spaces, it is crucial to deal with the problem of over-fitting problems. Many experimental results indicate that the SVM is able to generalize classification boundary very well and avoid over-fitting in high dimensional feature spaces. Thus, we use the SVM method with the tree kernel to extract protein interactions.

3 Tree Kernel for Protein-Protein Interaction Extraction

3.1 Tree Kernel

There have been many approaches for text classification using kernel methods. The BOW (bag-of-word) kernel, which uses word frequency vectors as features and calculates inner products to get their similarities, is a typical form of kernel classification methods [11].

Kernel-based approaches using simple word distribution of documents cannot make use of the grammatical structure, however, new kernel method which utilizes the structural information have been proposed in Collins [12]. The sequence kernel considers the data as a sequence of characters and the common subsequences as attributes. It calculates kernel value by counting these common subsequences. The kernels which calculate structural similarity in a recursive manner are called ‘convolution kernel’ [13].

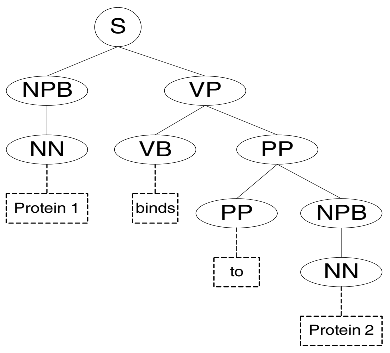


Fig. 1. An example of parsing tree

The tree kernel is a convolution kernel and naturally uses the grammatical structure. The tree kernel represents a structure with a tree form and numbers the child nodes of parent node by its order. A parsing tree represents the text and the structural information. Figure 1 shows the parsing tree of "protein1 binds to protein2." In the tree kernel, kernel value is evaluated by summing up the number of common subtrees between two trees. Consequently, the tree kernel can be used to calculate the structural similarity effectively.

A tree represents as a vector of subtree consisting of corresponding tree itself through high dimensional feature mapping:

$$\Phi(\text{Tree } T) = (subTree(\text{type } 1), \ldots, subTtree(\text{type } n)), \tag{1}$$

where $subTree(\text{type } n)$ is the number of subtree of node type n.

The kernel function is defined as follows:

$$K(T_1, T_2) = \langle \Phi(T_1) \cdot \Phi(T_2) \rangle = \sum_{l} \Phi(T_1)[i] \times \Phi(T_2)[i] \tag{2}$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_{i} I_i(n_1) \times I_i(n_2), \tag{3}$$

where N_1 and N_2 represent the set of all possible nodes of tree T_1 and T_2, and $I_i(n)$ is an indicator function which has 1 if subtree of type i is started from root node n, 0 otherwise.

The number of subtrees with type i in T is calculated by $\Phi(T)[i] = \sum_{n \in N} I_i(n)$. It means that the total number of nodes in tree T which have subtrees with type i. The inner product between two trees, having its features as the all possible sub-trees, is computed by the following recursive way and it is known to be calculated in polynomial time.

1. If the form of the children nodes of n_1 and n_2 are different,

$$NumComSubt(n_1, n_2) = 0 \tag{4}$$

2. If the form of the children nodes of n_1 and n_2 are identical (including their order) and they are leaf nodes (POS tag),

$$NumComSubt(n_1, n_2) = \lambda \tag{5}$$

3. For all other cases,

$$NumComSubt(n_1, n_2) = \prod_j (1 + NumComSubt(ch(n_1)_j, ch(n_2)_j)), \quad (6)$$

where $ch(n_1)_j$ is the j-th child of node n_1, $ch(n_2)_j$ is the j-th child of node n_2, and $NumComSubt(n_1, n_2) = \lambda \sum_i I_i(n_1) \times I_i(n_2)$. The parameter λ, $0 < \lambda \leq 1$, is used to consider the relative importance of tree fragment according to its length and is set to '1' when the size of tree fragments is not considered.

3.2 Applying Tree Kernel

We can improve some degree of extraction performance simply by using a set of patterns or rules because the protein interactions are represented by typical forms in many cases [2][14]. Generally, text sentences are generated from specific rules such as grammar, then they form grammatical structures. Thus we can utilize the structural properties from texts.

In this paper, we use a tree kernel which calculates tree similarity implicitly without explicit rules or templates. By using the tree kernel we can compute the similarity between two parse trees without modifying their structures. On the other hand, it does not necessarily need to analyze full tree to extract protein-protein interactions because the result can be decided by only sub-structure including the interactions. Therefore, we only use the minimum subtrees that have two proteins from full trees. This work helps to improve computational efficiency and accurate extraction.

3.3 Adding Semantic Information by Tag-Transformation

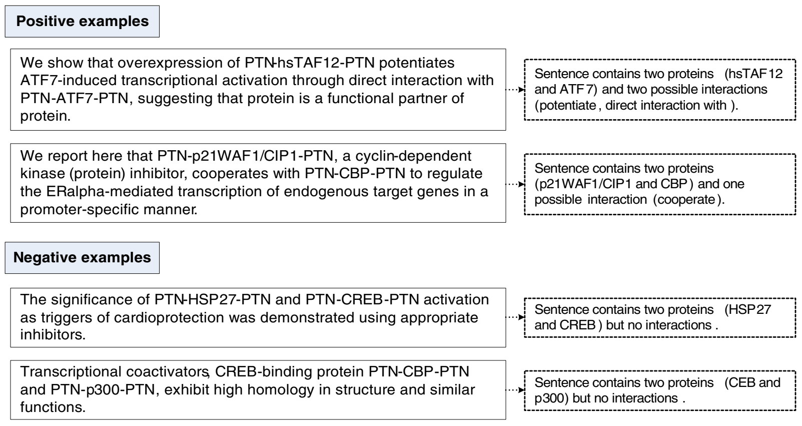
In the parsing tree, the tag information at leaf nodes plays an important role to identify structure patterns, then we can add simple semantic information for protein or interaction by modifying their POS tags. Possible two candidates are 'NN' tag for protein and 'VB' tag for interaction. Firstly, we can modify 'NN' tag of protein to 'PTN' to represent explicitly which is a protein in tree structure level. This tag modification would not have any advantage when the structure of positive sentence (which contains valid protein interaction) and negative sentence (which does not contain any protein interaction) are totally different. But, if both sentences from positive and negative examples have similar structure and the modified POS tags are used, we could discriminate protein interactions more easily.

Secondly, we can modify interaction-related verbs and nouns. A list of interaction verbs representing protein interactions are well studied in many researches. We have chosen interaction-related verbs determined by referencing these resources and by human experts. A POS tag for interaction-related verbs is transformed by adding '-I' (from 'VB' to 'VB-I') to differentiate protein interaction verbs from general verbs. A POS tag for interaction-related nouns is also transformed in the same way, from 'NN' to 'NN-I', to distinguish deverbal nouns (e.g., a noun 'regulation' formed from a verb 'regulate') from other normal nouns. These modified tags are used to give rich information when we calculate kernel value to compare similarity of subtree structures. In the experiments, we examine the impact of using the semantic information.

4 Experimental Results

4.1 Data Set

In order to generate the data set, we have first found the abstracts with the keyword, 'protein interaction' or 'protein-protein interaction' in PubMed. Next, five proteins (TRPC, CREB, ERK, FOS, and EGR) have been selected as queries among proteins that have over 2,000 relevant abstracts. More than 10,000 abstracts were retrieved by the five queries, and segmented into sentences, where the number of sentences was about 100,000. To make the problem more difficult, we discarded any sentence which contains less than two protein names or no interaction-related word. Note that the sentences which contain at least two protein names and one interaction-related word may not include protein-protein interactions at all. Finally, the sentences which include protein-protein interactions were labeled as 'positive', otherwise labeled as 'negative'. The protein names and interaction-related words were predetermined by human experts and all sentences are manually classified. Consequently, we have got total 1,135 sentences of 'positive' and 569 sentences of 'negative'. Figure 2 shows the examples from positive and negative sentences.



Positive examples

We show that overexpression of PTN-hsTAF12-PTN potentiates ATF7-induced transcriptional activation through direct interaction with PTN-ATF7-PTN, suggesting that protein is a functional partner of protein.

Sentence contains two proteins (hsTAF12 and ATF7) and two possible interactions (potentiate, direct interaction with).

We report here that PTN-p21WAF1/CIP1-PTN, a cyclin-dependent kinase (protein) inhibitor, cooperates with PTN-CBP-PTN to regulate the ERalpha-mediated transcription of endogenous target genes in a promoter-specific manner.

Sentence contains two proteins (p21WAF1/CIP1 and CBP) and one possible interaction (cooperate).

Negative examples

The significance of PTN-HSP27-PTN and PTN-CREB-PTN activation as triggers of cardioprotection was demonstrated using appropriate inhibitors.

Sentence contains two proteins (HSP27 and CREB) but no interactions.

Transcriptional coactivators, CREB-binding protein PTN-CBP-PTN and PTN-p300-PTN, exhibit high homology in structure and similar functions.

Sentence contains two proteins (CEB and p300) but no interactions.

Fig. 2. The examples of 'positive' and 'negative' sentences

4.2 Evaluation Measure

Table 1 presents the labels given as a result of the relationship between a system's output and an answer. Based on the table, the system performance measured as follows:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \cdot 100\%$$
$$precision = \frac{TP}{TP + FP} \cdot 100\%$$
$$recall = \frac{TP}{TP + FN} \cdot 100\%$$
$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

Table 1. The labels given as a result of the relationship between a system's output and an answer

		Answer	
		Positive	*Negative*
Test Results	*Positive*	*TP*	*FP*
	Negative	*FN*	*TN*

4.3 Protein Interaction Extraction

For experiments, we have used Brill tagger [15] and Collins parser [16] to construct parsing trees. 10-fold cross-validation is performed to evaluate the system performance because the data set is not large enough for a comparative study. We also have generated two different data sets from the original examples. One set contains original sentences with unchanged tags, but the other set is additionally tagged by using new keyword 'PTN' and '-I', which is explained in Section 3.3. The 'PTN' indicates 'protein name' and the '-I' indicates 'interaction-related word'.

We first performed the protein-protein interaction extraction using all 1,704 sentences, which unbalances positive and negative examples. Figure 3 depicts the performance results using tree kernel methods. When $\lambda \leq 0.5$, the tree kernel approach gives high performance in accuracy and F1-score for both data sets. Note that λ in tree kernels has been introduced to scale the relative importance

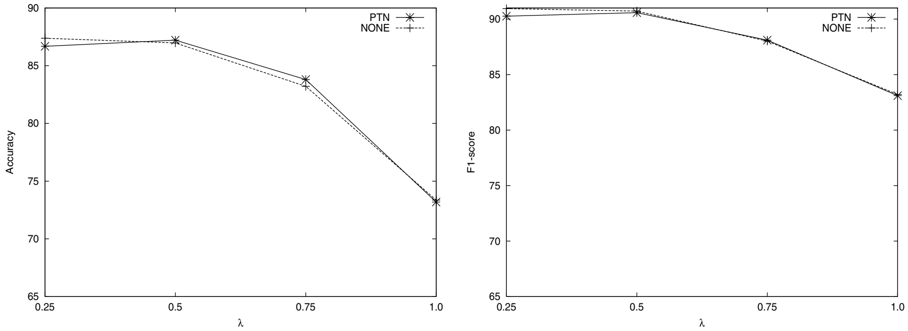


Fig. 3. The performance comparison of the tree kernel methods using all examples

Table 2. The performance comparison of the proposed approach and other methods

Method	Tree Kernel	BOW Kernel	naïve Bayes
Accuracy	87.21	83.96	81.77
F1-score	90.58	88.61	86.71

of tree fragments with their size. Using λ, we can adjust the degree of down-weight according to tree fragments' size. Therefore we can conclude that it is more important to look at the overall structure rather than the detail of the structure, especially for long and complex sentences. However, both data sets do not provide any difference for the tree kernel method. We infer that it occurs from the unbalanced data. The SVM classifier is learned more focusing on positive examples because positive examples are twice more than negative ones, and it causes relatively high recall and blurs other performance factors.

Table 2 presents the performance comparison of our approach and other methods in accuracy and F1-score. It compares with the BOW kernel and the naïve Bayes classifier. The tree kernel performance is the results obtained from 'PTN' and '-I' tag transformation when $\lambda = 0.5$. Our approach shows 87.21% of accuracy and 90.58% of F1-score, while the BOW kernel method shows 83.96% of accuracy and 88.61% of F1-score, and the naïve Bayes classifier shows 81.77% of accuracy and 86.71% of F1-score. Here, we find that the tree kernel methods can provide better performance than typical approaches.

Because of the unbalance issue between the number of positive and negative examples, we have measured the extraction performance using balanced data. Each 569 sentences were randomly chosen from positive and negative examples. Figure 4 presents the performance comparison of our methods using the balanced data. It shows the data set tagged by 'PTN' and '-I' provides better performance than original data set over all λ. It means that the extraction system can improve its performance by using extra tags if protein names and interaction-related words are properly detected and tagged.

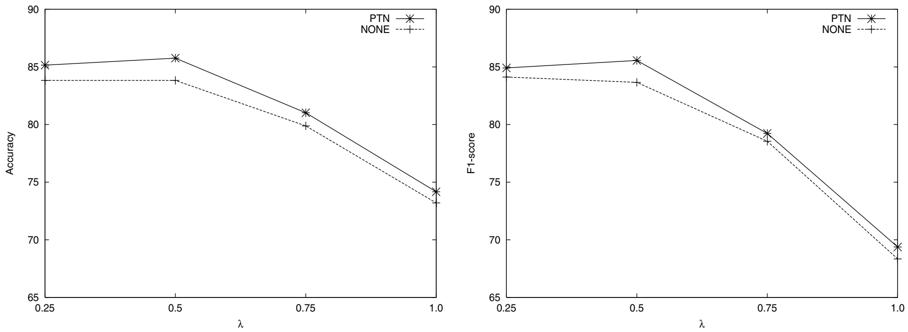


Fig. 4. The performance comparison of the tree kernel methods using randomly selected data

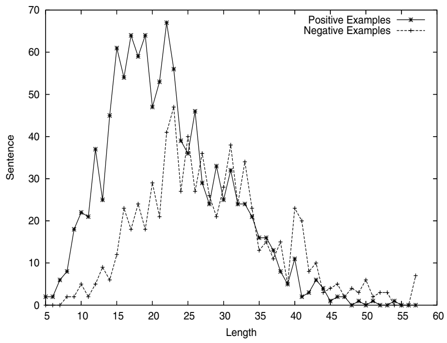
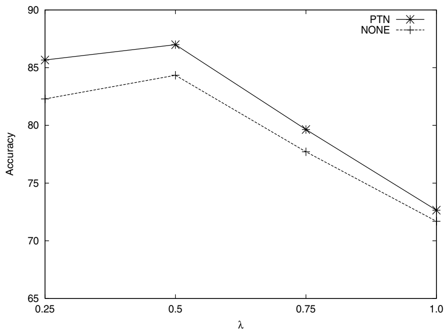


Fig. 5. Number of examples for the sentence length



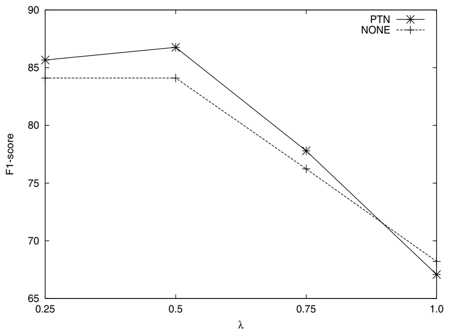


Fig. 6. The performance comparison of the tree kernel methods using randomly selected data, which have similar length

Further, we analyze the number of examples for the sentence length, which shown in Figure 5. It describes positive examples are relatively short compared than negative examples. Under same condition of protein and interaction-related word counts, negative examples have high possibility to be long sentences because the protein-protein interaction is not involved in the examples. Thus we equally selected about 440 sentences with similar length from positive and negative examples. Figure 6 presents the performance results using similar length examples. Like previous results, the best performance is achieved when $\lambda = 0.5$. We can also think that normalization of word length can be useful to identify protein-protein interaction although the tree kernel already considers the sentence length by the parameter λ. It is well known that document length incorporates into the system performance in IR (Information Retrieval) task.

5 Conclusion

In this paper, we proposed a tree kernel-based method to mine protein-protein interactions. Our approach transforms each sentence to a tree structure by using grammatical information, and a support vector machine based on the tree kernel is used to extract the protein-protein interactions. The SVM is learned from the given predefined interaction corpus and interaction patterns. Then it extracts the protein-protein interactions from new sentences.

For experiments, we performed 10-fold cross-validation for 1,135 positive examples and 569 negative examples obtained from PubMed. The experimental results show our approach provides better performance than other methods in accuracy and F1-score. Also, the data set using 'PTN' and '-I' tag transformation supports more accurate extraction, which means that the detection of 'protein names' or 'interaction-related words' is important to improve the performance. It is interesting that the accuracy and the F1-score achieve the highest score when $\lambda \leq 0.5$. It suggests the overall form of tree structure is one of the key points for the extraction performance rather than the detail of structure. However, the tree kernel can be modified to consider sentence length or protein-protein word distance to capture informative factors, which remains as future works.

Acknowledgements

This research was supported in part by the National Research Laboratory Program of the Korea Ministry of Science and Technology and by the Korean Ministry of Education under the BK21-IT Program. The ICT at Seoul National University provided research facilities for this study.

References

[1] Deng, M., Mehta, S., Sun, F., and Chen, T.: Inferring domain-domain interactions from protein-protein interactions. *Genome Research* **12** (2002) 1540–1548

[2] Huang, M., Zhu, X., Hao, Y., Payan, D.G., Qu, K., and Li, M.: Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics* **20(18)** (2004) 3604–3612

[3] Yakushiji, A., Tateisi, Y., and Miyao, Y.: Event extraction from biomedical parsers using a full parser. In *Proceedings of the 6th Pacific Symposium on Biocomputing* (2001) 408–419

[4] Park, J.C., Kim, H.S., and Kim, J.J.: Bidirectional incremental parsing for automatic pathway identification with combinatory categorical grammar. In *Proceedings of the 6th Pacific Symposium on Biocomputing* (2001) 396–407

[5] Temkin, J.M. and Gilder, M.R.: Extraction of protein interaction information from unstructured text using a content-free grammar. *Bioinformatics* **19(16)** (2003) 2046–2053

[6] Leroy, G. and Chen, H.: Filling preposition-based templates to capture information from medical abstracts. In *Proceedings of the 7th Pacific Symposium on Biocomputing* (2002) 350–361

[7] Pustejovsky, J., Castano, J., Zhang, J., Kotecki, M., and Cochran, B.: Robust relational parsing over biomedical literature: extracting inhibit relations. In *Proceedings of the 7th Pacific Symposium on Biocomputing* (2002) 362–373

[8] Zelenko, D., Aone, C., and Richardella, A.: Kernel methods for relation extraction. *Journal of Machine Learning Research* **3** (2003) 1083–1106

[9] Lodhi, H., Shawe-Taylor, J., Cristianini, N., and Watkins, C.: Text classification using string kernels. *Journal of Machine Learning Research* **2** (2002) 419–444

[10] Vapnik, V.: *The Nature of Statistical Learning Theory*, Springer, 1995

[11] Cancedda, N., Gaussier, E., Goutte, C., and Renders, J.M.: Word-sequence kernels. *Journal of Machine Learnign Research* **3(6)** (2003) 1059–1082

[12] Collins, M.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of 40th Conference of the Associations for Computational Linguistics* (2002) 625–632

[13] Collins, M. and Duffy, N.: Convolution kernels for natural languages. In *Proceedings of the 15th Annual Conference on Neural Information Processing Systems* **14** (2001) 625–632

[14] Hao, Y., Huang, M., and Li, M.: Discovering patterns to extract protein-protein interactions from full texts - part II. *Bioinformatics* **21(15)** (2005) 3294–3300

[15] Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics* **21(4)** (1995) 543–565

[16] Collins, M.: Head-driven statistical models for natural language parsing. *Doctoral Dissertation*, Dept. of Computer and Information Science, University of Pennsylbania, Philadelphia (1999)