Artificial Intelligence in Medicine 51 (2011) 163–173

Contents lists available at ScienceDirect

Artificial Intelligence in Medicine

journal homepage: www.elsevier.com/locate/aiim

Multiple kernel learning in protein–protein interaction extraction from biomedical literature

Zhihao Yang[a,*], Nan Tang[a], Xiao Zhang[a], Hongfei Lin[a], Yanpeng Li[a], Zhiwei Yang[b]

[a] *Department of Computer Science and Engineering, Dalian University of Technology, Dalian 116024, China*
[b] *Department of Ultrasound, Oil Field Hospital of Daqing, Heilongjiang 163001, China*

ARTICLE INFO

Article history:
Received 26 September 2009
Received in revised form
25 November 2010
Accepted 7 December 2010

Keywords:
Support vector machines
Multiple kernel learning
Text mining
Information extraction
Protein–protein interaction

ABSTRACT

Objective: Knowledge about protein–protein interactions (PPIs) unveils the molecular mechanisms of biological processes. The volume and content of published biomedical literature on protein interactions is expanding rapidly, making it increasingly difficult for interaction database administrators, responsible for content input and maintenance to detect and manually update protein interaction information. The objective of this work is to develop an effective approach to automatic extraction of PPI information from biomedical literature.
Methods and materials: We present a weighted multiple kernel learning-based approach for automatic PPI extraction from biomedical literature. The approach combines the following kernels: feature-based, tree, graph and part-of-speech (POS) path. In particular, we extend the shortest path-enclosed tree (SPT) and dependency path tree to capture richer contextual information.
Results: Our experimental results show that the combination of SPT and dependency path tree extensions contributes to the improvement of performance by almost 0.7 percentage units in *F*-score and 2 percentage units in area under the receiver operating characteristics curve (AUC). Combining two or more appropriately weighed individual will further improve the performance. Both on the individual corpus and cross-corpus evaluation our combined kernel can achieve state-of-the-art performance with respect to comparable evaluations, with 64.41% *F*-score and 88.46% AUC on the AImed corpus.
Conclusions: As different kernels calculate the similarity between two sentences from different aspects. Our combined kernel can reduce the risk of missing important features. More specifically, we use a weighted linear combination of individual kernels instead of assigning the same weight to each individual kernel, thus allowing the introduction of each kernel to incrementally contribute to the performance improvement. In addition, SPT and dependency path tree extensions can improve the performance by including richer context information.

1. Introduction

Protein–protein interactions (PPIs) play a key role in several aspects of the structural and functional cell organization. Thus understanding PPI enables unveiling the molecular mechanisms of biological processes. A number of databases, such as MINT [1], BIND [2], and DIP [3], have been created to store protein interaction information in structured and standard formats. However, the biomedical literature regarding protein interactions is expanding rapidly, making it difficult for interaction database administrators to detect and manually input or update the content. Thus, most of the protein interaction information remains hidden within biomedical literature, making its automatic extraction from biomedical literature an extremely important research area.

The existing PPI methodologies rely on different approaches, broadly divided into three main categories: manual pattern engineering, grammar engineering, and machine learning.

Manual pattern engineering approaches define a set of rules for possible textual relationships, called patterns, which encode similar structures in expressing relationships. The SUISEKI system uses regular expressions, with probabilities that reflect the experimental accuracy of each pattern to extract interactions into predefined frame structures [4]. Ono et al. manually defined a set of rules based on syntactic features to preprocess complex sentences, making further use of negation structures [5]. The BioRAT system uses manually engineered templates that combine lexical and semantic information to identify protein interactions [6]. Such manual pattern engineering approaches for information extraction are very hard to scale up to large document collections since they require labor-intensive and skill-dependent pattern engineering.

* Corresponding author at: Department of Computer Science and Engineering, Dalian University of Technology, No. 2 LingGong Road, Dalian, Liaoning 116023, China. Tel.: +86 411 84706009x3926; fax: +86 411 84708116.
E-mail address: yangzh@dlut.edu.cn (Z. Yang).

0933-3657/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.artmed.2010.12.002

Grammar engineering approaches use manually generated specialized grammar rules that perform a deep parse of the sentences. Sekimizu et al. used shallow parser, EngCG, to generate syntactic, morphological, and boundary tags [7]. Based on the tagging results, subjects and objects were recognized for the most frequently used verbs in a collection of abstracts that were believed to express the interactions between proteins. Fundel et al. proposed RelEx based on the dependency parse trees to extract relations [8].

With the advent of ever more complex and powerful technologies, machine learning techniques for extracting protein interaction information are increasingly becoming the subject of scientific research. In most recent work on machine learning for PPI extraction, the PPI extraction is performed by a decision function that determines for each unordered candidate pair of protein names occurring together in a sentence whether the two proteins interact or not. Xiao et al. used Maximum Entropy models to combine diverse lexical, syntactic and semantic features for PPI extraction [9]. In a similar study, Zhou et al. employed a semantic parser using the Hidden Vector State (HVS) PPI model, which can be trained using only lightly annotated data while simultaneously retaining sufficient ability to capture the hierarchical structure [10]. In their attempt to identify protein interactions, Yang et al. used support vector machines to combine rich feature sets, including word, keyword, protein names distance, link path and link grammar extraction result feature [11]. Finally, Miyao et al. used the PPI extraction method based on support vector machines with sub-set tree kernels, while using different parsers and parse representations [12].

2. Related works

Kernel-based methods are an effective alternative to explicit feature extraction [13]. They retain the original representation of objects and use the objects to compute a kernel function between a pair of objects. Formally, a kernel function is a mapping K: $X \times X \to [0.\infty)$: from input space X to a similarity score $K(x, y) = \phi(x) \cdot \phi(y) = \sum_i \phi_i(x)\phi_i(y)$, where $\phi_i(x)$ is a function that maps X to a higher dimensional space without the need to know its explicit representation. Such a kernel function makes it possible to compute the similarity between objects without enumerating all the features.

Several kernels have been proposed, including subsequence kernels [14], tree kernels [15,16], shortest path kernels [17], and graph kernels [18]. Each kernel utilizes a portion of the structures to calculate useful similarity, whereby any one kernel cannot retrieve other important information that may be retrieved by other kernels.

In recent years researches have proposed the use of multiple kernels to retrieve the widest range of important information in a given sentence. Kim et al. suggested four kernels, namely, predicate, walk, dependency and hybrid kernels to adequately encapsulate information required for a relation prediction based on the sentential structures involved in two entities [19]. They obtained a 77.5% *F*-score with the walk kernel on the LLL corpus [20]. Miwa et al. proposed a method that combines kernels based on several syntactic parsers, in order to retrieve the widest possible range of important information from a given sentence [21]. Their method, which combines bag-of-words (BOW) kernel, sub-set tree kernel and graph kernel, obtained an *F*-score of 60.8% and an area under the receiver operating characteristics curve (AUC) of 86.8% on the AImed corpus [22].

However, the methods in [19,21] assign the same weight to each individual kernel and their combined kernel fails to achieve the best performance: the performance of the hybrid kernel in [19] is inferior to that of walk kernel, one of the individual kernels. In [21], the graph kernel outperforms the other individual kernels, and combined with the sub-set tree kernel, it achieves improved performance. However, when further combined with BOW kernel, the performance deteriorates. In fact, the performance of BOW kernel and graph kernel combination degrades below that of graph kernel alone.

In this paper, we propose a weighted multiple kernel learning-based approach to extracting PPI from biomedical literature. The approach combines several appropriately weighed kernels, namely feature-based, tree, graph and part-of-speech (POS) path kernel, whereby the kernel with better performance is assigned higher weight. Experimental results show that the introduction of each individual kernel contributes to the performance improvement. The other innovative features of our approach include: (a) in addition to the commonly used word feature, our feature-based kernel includes the protein name distance feature as well as the keyword feature. The introduction of keyword feature allows utilizing domain knowledge and improves the performance effectively. (b) Using our tree kernel allows for extending the shortest path-enclosed tree (SPT) and dependency path tree to capture richer contextual information.

The remaining part of this paper is organized as follows: Section 3 describes our method, Section 4 presents and discusses the experimental results and finally Section 5 offers some concluding remarks.

3. Methods

A kernel can be thought of as a similarity function for pairs of objects. Different kernels calculate the similarity with different aspects between two sentences. Combining the similarities can reduce the risk of missing important features and thus produces a new useful similarity measure. In this work, we combine several distinctive types of kernels to extract PPI: feature-based, tree, graph and POS path kernel.

3.1. Feature-based kernel

The following features are used in our feature-based kernel:

3.1.1. Word feature

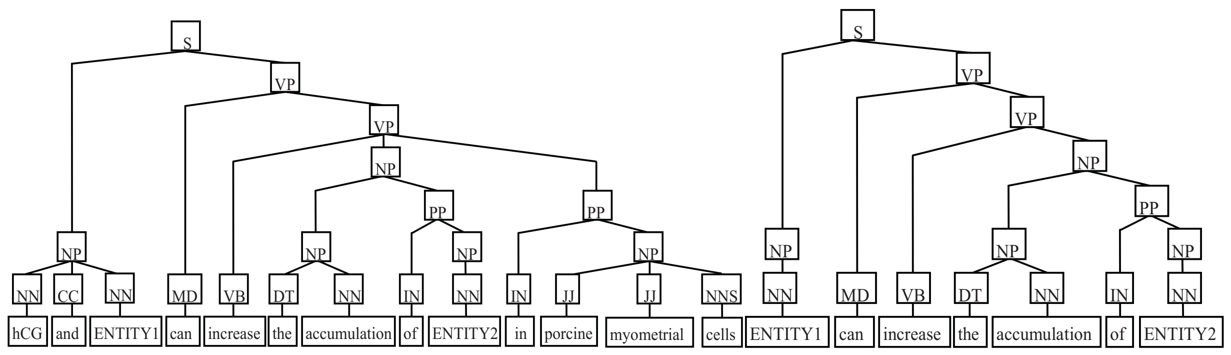
Our feature-based kernel takes two unordered sets of words as feature vectors and calculates their similarity. It is simple and efficient. There are two sets of word features used in our method.

- Words between two protein names. These features include all words that are located between two protein names.
- Words surrounding two protein names, which include *N* words to the left of the first protein name and *N* words to the right of the second protein name. *N* is set to be three in our experiment.

Our feature-based kernel is similar to the BOW kernel in [21], whereby the frequency of words is replaced by two new features – protein name distance feature and keyword feature.

3.1.2. Protein name distance feature

Under the assumption that the shorter the distance (the number of words) between two protein names is, the more likely the two proteins have interaction relation, the distance is chosen as a feature. If there are fewer than three words between two proteins, the feature value is set to “DISLessThanThree;” if there are more than or equal to three words but fewer than six words between two proteins, the feature value is set to “DISBetweenThreeSix.” The other feature values include “DISBetweenSixNine,” “DISBetween-NineTwelve” and “DISMoreThanTwelve.”



S
VP
VP
NP
PP
PP
NP
NP
NP
NP
NN CC NN MD VB DT NN IN NN IN JJ JJ NNS
hCG and ENTITY1 can increase the accumulation of ENTITY2 in porcine myometrial cells
S
VP
VP
NP
PP
NP
NP
NP
NN MD VB DT NN IN NN
ENTITY1 can increase the accumulation of ENTITY2

Fig. 1. MCT (left part) and SPT (right part) presentation of a relation instance in the example sentence "The present study demonstrates that hCG and ENTITY1 can increase the accumulation of ENTITY2 in porcine myometrial cells".

3.1.3. *Keyword feature*

The existence of an interaction keyword (the verb expressing protein interaction relation such as "bind," "interact," "inhibit," etc.) between two protein names or among the words surrounding two protein names often implies the existence of the PPI. Therefore, the existence of keywords is chosen as a binary feature. To identify the keywords in texts; we manually create an interaction keyword list of about 500 entries; which includes the interaction verbs and their variants (for example; interaction verb "bind" has variants "binding," "bound," etc. The list can be provided upon request).

3.2. *Tree kernel*

A convolution kernel aims to capture structured information in terms of substructures. As a specialized convolution kernel, convolution tree kernel $K_C(T_1, T_2)$ (where '*C*' denotes convolution) counts the number of common sub-trees (sub-structures) as the syntactic structure similarity between two parse trees T_1 and T_2 [23,24]:

$$K_C(T_1,\ T_2) = \sum_{n_1 \in N_1, n_2 \in N_2} \Delta(n_1, n_2) \tag{1}$$

where N_j is the set of nodes in tree T_j, and $\Delta(n_1, n_2)$ evaluates the common sub-trees rooted at n_1 and n_2 and is computed recursively as follows:

(1) If the context-free productions – Context-Free Grammar (CFG) rules – at n_1 and n_2 are different, $\Delta(n_1, n_2) = 0 = 0$; Otherwise go to 2.
(2) If both n_1 and n_2 are POS tags, $\Delta(n_1, n_2) = 1 \times \lambda$; Otherwise go to 3.
(3) Calculate $\Delta(n_1, n_2)$ recursively as:

$$\Delta(n_1, n_2) = \lambda \prod_{k=1}^{\#ch(n_1)} (1 + \Delta(ch(n_1, k),\ ch(n_2, k))) \tag{2}$$

where $\# ch(n)$ is the number of children of node n, $ch(n, k)$ is the kth child of node n and $\lambda\,(0 < \lambda < 1)$ is the decay factor in order to make the kernel value less variable with respect to different sub-tree sizes.

3.2.1. *Parse tree kernel*

A relation instance between two entities is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in the tree kernel calculation.

In order to focus on the information most relevant to relations, a standard tree kernel is often defined on the minimum complete tree (MCT) that contains both entities in a parse tree. It is the complete sub-tree rooted by the node of the nearest common ancestor of the two entities under consideration (an example is shown in Fig. 1). However, the MCT introduces superfluous left and right context information, which may include random noise features.

Zhang et al. explored five tree spans in relation extraction and found that the SPT (an example is shown in Fig. 1) performed best [25]. SPT is the smallest common sub-tree including the two entities. In other words, the sub-tree is enclosed by the shortest path linking the two entities in the parse tree. But in some cases, the information contained in SPT is not sufficient to determine the relationship between the two entities. For example, "interact" is critical to determine the relationship between "ENTITY1" and "ENTITY2" in the sentence "ENTITY1 and ENTITY2 interact with each other" as shown in Fig. 2. However, it is not contained in the SPT (dotted circle in Fig. 2). By analyzing the experimental data, we found that in these cases the SPTs usually have fewer than four leaf nodes and, therefore, include little information except the two entity names.

Here we employ a simple heuristic rule to expand the SPT span. By default, we adopt SPT as our tree span. When the number of leaf nodes in a SPT is smaller than four, the SPT is expanded to a higher level, i.e. the parent node of the root node of the original SPT is used as the new root node. Thus the new SPT (solid circle in Fig. 2) will include richer context information comprising the original SPT. In the above example, the flat SPT string is extended from "(NP (NN PROTEIN1) (CC and) (NN PROTEIN2))" to "(S (NP (NN PROTEIN1) (CC and) (NN PROTEIN2)) (VP (VBP interact) (PP ((IN with) (NP (DT each) (JJ other)))))" and includes richer context information.

3.2.2. *Dependency path tree kernel*

The other type of tree structure information included in our tree kernel is provided by the parser dependency analysis output. For dependency-based parse representations, a dependency path is encoded as a flat tree as depicted as follows: (DEPENDENCY (NSUBJ (interacts ENTITY1)) (PREP (interacts with)) (POBJ (with ENTITY2))) corresponding to the sentence "ENTITY1 interacts with ENTITY2." As tree kernel measures the similarity of trees by counting common sub-trees, it is expected that the system will identify effective subsequences of dependency paths.

Similar to SPT parse tree, in some cases, dependency path tree also needs extension. Taking the sentence "The expression of rsfA is under the control of both ENTITY1 and ENTITY2" as an example (its dependency parse is shown in Fig. 3), the path tree between ENTITY1 and ENTITY2 is given by "(DEPENDENCY (CONJ (ENTITY1, ENTITY2))." Obviously, the information in this path tree is insufficient to determine the relationship between the two entities. Thus, when the length of the dependency path between two proteins is shorter than three, it is extended. In such cases, if two edges are present to the left of the first protein in the entire dependency parse path, they will be included into the dependency path. Otherwise, the two edges to the right of the second protein will be included into the dependency path. In the above example, the path tree between ENTITY1 and ENTITY2 is extended from "(DEPENDENCY

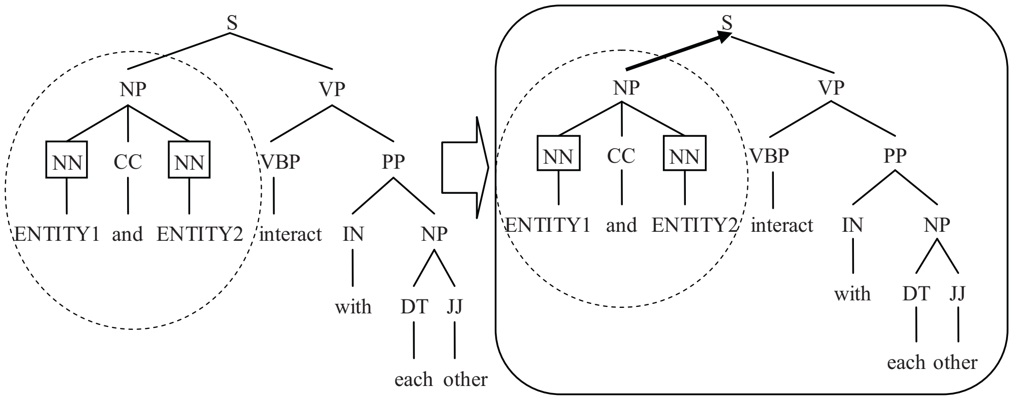


Fig. 2. An example of the extension of SPT (the original SPT is in dotted circle and extended SPT in solid circle).

(CONJ (ENTITY1, ENTITY2))" to "(DEPENDENCY (PREP(control, of)) POBJ((of, ENTITY1)) (CONJ(ENTITY1, ENTITY2)))". The example is shown in Fig. 3 (the bold solid edge is the original dependency path and the bold dotted edges are included into the new dependency path). The optimal extension threshold three is determined experimentally to achieve the best performance.

In contrast to our tree kernel, the sub-set tree kernel in [21] relies on the predicate type information (from the deep parser) to represent the dependency types, which was unused in previous works. However, it does not use the syntactic tree structure information.

3.3. Graph kernel

A graph kernel calculates the similarity between two input graphs by comparing the relations between common vertices (nodes). The graph kernel used in our method is the all-paths graph kernel proposed by Airola et al. [18]. The kernel represents the target pair using graph matrices based on two sub-graphs, where the graph features include all non-zero elements in the graph matrices. The two sub-graphs are a parse structure sub-graph (PSS) and a linear order sub-graph (LOS), as shown in Fig. 4. PSS represents the parse structure of a sentence and includes word or link vertices. A word vertex contains its lemma and its POS, while a link vertex contains its link. Additionally, both types of vertices contain their positions relative to the shortest path. The "IP"s in the vertices on the shortest path represent the positions which differentiated them from other vertices, such as "P," "CC," and "and: CC" in Fig. 4. LOS represents the word sequence in the sentence, thus has word vertices, each of which contains its lemma, its relative position to the target pair and its POS.

For the calculation, two types of matrices are used: a label matrix L, and an edge matrix A. The label matrix is a (sparse) $N \times L$ matrix, where N is the number of vertices (nodes), and L is the number of labels. It represents the correspondence between labels and ver-

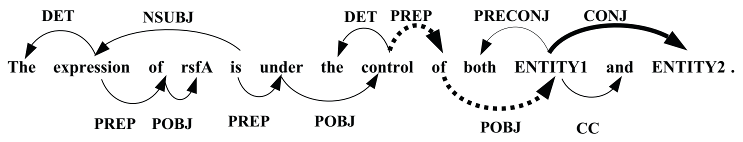

DET
NSUBJ
DET
PREP
PRECONJ
CONJ
The expression of rsfA is under the control of both ENTITY1 and ENTITY2 .
PREP
POBJ
PREP
POBJ
POBJ
CC

Fig. 3. An example of dependency path tree extension. The bold solid edge is the original dependency path and the bold dotted edges are included into the new dependency path.

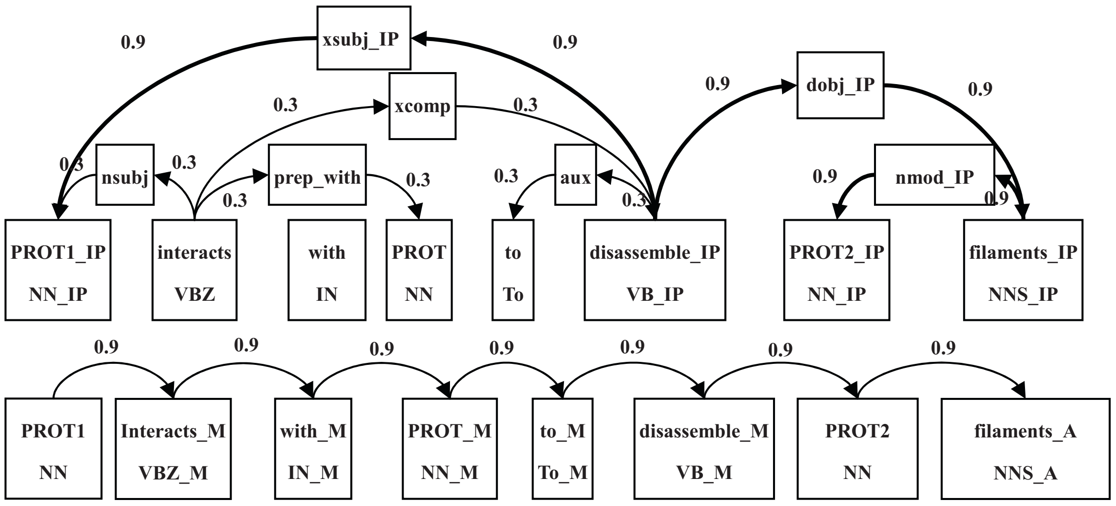


Fig. 4. Graph representation generated from an example sentence. The candidate interaction pair is marked as PROT1 and PROT2, the third protein is marked as PROT. The shortest path between the proteins is shown in bold. In the dependency based subgraph all nodes in a shortest path are specialized using a post-tag (IP). In the linear order subgraph possible tags are (B)efore, (M)iddle, and (A)fter.

tices, where L_{ij} is equal to 1 if the *i*th vertex corresponds to the *j*th label, and 0 otherwise. The edge matrix is a (sparse) $N \times N$ matrix, and represents the relation between pairs of vertices, where A_{ij} is a weight w_{ij} if the *i*th vertex is connected to the *j*th vertex, and 0 otherwise. The weight is a predefined constant whereby the edges on the shortest paths are assigned a weight of 0.9 and other edges receive a weight of 0.3. Using the Neumann Series, a graph matrix G is calculated as:

$$G = L^T \sum_{n=1}^{\infty} A^n L = L^T((I - A)^{-1} - I)L \tag{3}$$

This matrix represents the sums of all the path weights between any pair of vertices resulting in entries representing the strength of the relation between each pair of vertices. Using two input graph matrices G and G', the graph kernel $k(G, G')$ is the sum of the products of the common relations' weights, given by Eq. (4).

$$k(G, G') = \sum_{i=1}^{L} \sum_{j-1}^{L} G_{ij} G'_{ij} \tag{4}$$

The graph kernel designed in this study differs from the representation in [21] in several important aspects. First, in [21] each word in the shortest path has two labels, and the relations in the shortest path are not replaced but duplicated in the first sub-graph. Second, the shortest path is calculated by using the constituents in the PAS structure. The words in the constituents in the shortest path are distinguishably marked as being "in the shortest path" (IP). Finally, the POS information for protein names is not attached.

3.4. POS path kernel

The POS tag path between two protein names in parse trees contains rich syntactical and semantic information. To compute the similarity between two POS tag paths, we design a POS path kernel which takes into account the length of the path (the number of tags in the path) as well as the path dimension (the number of types of tags in the path). For example, for a path x equal to "NNP NP S VP NP NN", $len(x) = 6$ and $dim(x) = 5$. Furthermore, path-matching method is utilized to measure the similarity between two paths, accounting for not only the number and types of common tags, but also the order of these common tags; since the order of tags contains rich structured information and the same tags in different order may express different structures. Thus, retaining the original order of the common tags is necessary. For this purpose, we define two kinds of matching algorithms: forward and backward matching algorithm.

Firstly, we define two patterns: the standard pattern and the match pattern. For two POS tag paths aPath and bPath, the longer path is used as the standard pattern the shorter as the match pattern. The forward matching starts with the head of the standard pattern while the backward matching starts with the end of the standard pattern. These labels are matched in turn, so that if no match is found, the label will be skipped until the last label of the standard pattern is matched. We define the result of the forward matching as forePath, and the result of the backward matching as backPath. Thus for aPath = "JJ NP NP PP NP PP NP NN" and bPath = "JJ NP NP PP NP JJ," forePath = "JJ NP NP PP NP" and backPath = "JJ NP NP PP NP." The examples are shown in Figs. 5 and 6.

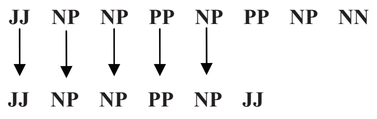


Fig. 5. An example of forward matching.

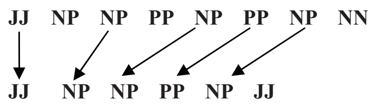


Fig. 6. An example of backward matching.

In the next step a similarity function $f_{kl}(x, y)$ is defined and used to compute the similarity between path x and y based on their lengths.

$$\begin{aligned} f_{kl}(x, y) &= \frac{len(f) + len(b)}{len(x)} \times \frac{len(f) + len(b)}{len(y)} \times \frac{\min(len(x), len(y))}{\max(len(x), len(y))} \\ &= \left(\frac{len(f) + len(b)}{\max(len(x), len(y))} \right)^2 \end{aligned} \tag{5}$$

where f and b represent the results of the forward matching and the backward matching, respectively.

Similarly, $f_{kd}(x, y)$ is defined to calculate the dimension similarity between path x and y:

$$\begin{aligned} f_{kd}(x, y) &= \frac{dim(f + b)}{dim(x)} \times \frac{dim(f + b)}{dim(y)} \times \frac{\min(dim(x), dim(y))}{\max(dim(x), dim(y))} \\ &= \left(\frac{dim(f + b)}{\max(dim(x), dim(y))} \right)^2 \end{aligned} \tag{6}$$

In the Eq. (6), $dim(f + b)$ is used to compute the total dimension of f and b (the total number of unique tag types in f and b). The length and the dimension information in our POS path kernel function can then be expressed as follows:

$$\begin{aligned} f_k(x, y) &= f_{kl}(x, y) \times f_{kd}(x, y) \\ &= \left(\frac{len(f) + len(b)}{\max(len(x), len(y))} \right)^2 \times \left(\frac{dim(f + b)}{\max(dim(x), dim(y))} \right)^2 \end{aligned} \tag{7}$$

3.5. Combination of kernels

We will now consider an example given by the sentence "***Aip1p*** interacts with ***cofilin*** to disassemble actin filaments." The extracted features for the protein pair ***Aip1*** and ***cofilin*** are given in Table 1, while Fig. 4 depicts its graph (feature) representation.

Each kernel has its own pros and cons. Firstly, all kernels ignore the words. Furthermore, the dependency path kernel ignores some deep information, and conversely, whereas the parse tree kernel does not output certain shallow relations. POS path kernel can capture syntactical and semantic information in POS tag path. The feature-based kernel is simple and efficient, but cannot capture the sentence structure. The graph kernel can treat parser's output and word features at the same time, but relies on tuning the kernel parameters and may miss some distant words and similarities of paths among more than three elements [21].

Given their different characteristics, clearly different kernels calculate the similarity between two sentences from different aspects. Thus combining the similarities can reduce the risk of missing important features and thus produces a new useful similarity measure. To realize the combination of different types of kernels based on different parse structures, we sum up the normalized output of several kernels K_m as:

$$K(x, x') = \sum_{m=1}^{M} \sigma_m K_m(x, x') \tag{8}$$

$$\sum_{m=1}^{M} \sigma_m = 1, \quad \sigma_m \geq 0, \ \forall m \tag{9}$$

Table 1
Features extracted for the protein pair ***Aip1p*** and ***cofilin*** from the sentence "***Aip1p*** interacts with ***cofilin*** to disassemble actin filaments.".

Feature names	Feature values
Words between two protein names	b_interacts, b_with b_catalytic, b_alpha/alpha', b_subunits, b_of
Left words	l_
Right words	r_to, r_disassemble, r_actin
Protein name distance	DISLessThanThree
Keyword	IncludingKeyword
SPT	(S (NP (NN PROTEIN1)) (VP (VBZ interacts) (PP (IN with) (NP (NN PROTEIN2)))))
Dependency path	(DPTREE (nsubj(interacts, PROTEIN1)) (prep(interacts, with)) (pobj(with, PROTEIN2)))
POS path	NNP NP S VP PP NP NN

Table 2
The number of features used for each type of word features on five corpora.

	AImed	BioInfer	HPRD50	IEPA	LLL
Words between two protein names	9209	9941	1610	4603	1036
Words surrounding two protein names	7337	8185	1063	1434	677

Table 3
Effectiveness of different features in the feature-based kernel and their combinations on AImed.

	P	R	F	σ_F	AUC	σ_{AUC}
Words	42.58	62.9	50.82	2.9	77.69	3.4
Words + protein names distance	43.65	62.3	51.3	5.4	78.23	3.6
Words + protein names distance + keyword	46.32	61.1	52.69	4.9	80.71	4.1

where M represents the number of kernel types, and σ_m is the experimentally determined weight of each K_m, adjusted until the overall best results are achieved. We found that each individual kernel has different performance and only when the kernels with better performances are assigned higher weights can the combination of individual kernels produce the best result. In our experiments, the weights for feature-based kernel, tree kernel, graph kernel and POS path kernel are set to 0.6, 0.15, 0.15 and 0.1, respectively in the order of performance rank (the weights of each individual kernel in combined kernel are shown in Table 5). This is a very simple combination, but the resulting kernel function contains all of the kernels' information. Comparatively, the methods in [19,21] assign the same weight to each individual kernel and their combined kernels fail to achieve the best performance.

4. Experiments

4.1. Experimental setting

We evaluated our method with five publicly available corpora that contain PPI interaction annotation: AImed [22], BioInfer [26], HPRD50 [8], IEPA [27] and LLL [20]. All the corpora are processed to a common format using transformations [28] and parsed using Stanford parser (http://nlp.stanford.edu/software/lex-parser.shtml) to generate the output of parse tree, dependency path and POS path. Further, following the approach given in [18], self-interactions are removed from the corpora and thus not considered in evaluation. In our implementation, we used the SVMLight package (http://svmlight.joachims.org/) developed by Joachims for our feature-based kernel. The polynomial kernel is chosen with parameter $d = 4$. Tree Kernel Toolkit developed by Moschitti with default parameters is chosen for our tree kernel (http://dit.unitn.it/~moschitt/Tree-Kernel.htm). For the graph kernel All-paths graph kernel proposed by Airola et al. (http://mars.cs.utu.fi/PPICorpora/GraphKernel.html) is used. Our POS path kernel is implemented with SVMLight user defined kernel interface.

In the single corpus tests we evaluated our method with 10-fold document-level cross-validation on all of the corpora. This guarantees the maximal use of the available data and allows comparison to the earlier relevant work. In particular, on the AImed corpus we applied the 10-fold split used by [14,18,21,29,30]. In cross-corpus tests we used each of the corpora in turn to train an extraction system and tested the system on the four remaining corpora.

The majority of PPI extraction system evaluations use the balanced F-score measure for quantifying the performance of the systems. This metric is defined as $F\text{-score} = (2PR)/(P + R)$ where P denotes precision and R recall. However, according to [18], there is a critical weakness in the F-score metric in comparisons involving different corpora. As an alternative to F-score, we also evaluated the performance by AUC measure [31], as due to its important property that it is invariant to the class distribution of the used dataset, it has been recommended for performance evaluation [32].

4.2. Experimental results and discussion

In this section, we firstly discuss the effectiveness of different features used in the feature-based kernel, SPT and dependency tree and their extensions, and different kernels on AImed corpus. Here AImed is used since it is sufficiently large for training and reliably

Table 4
Effectiveness of SPT, dependency tree and their extensions on AImed.

	P	R	F	σ_F	AUC	σ_{AUC}
SPT	40.09	66.74	50.13	3.2	76.32	2.7
SPT Extension	42.37	65.8	51.56	3.3	78.05	2.2
Dependency	18.76	58.33	29.17	3.2	54.37	2.3
Dependency Extension	20.49	56.18	30.03	3.6	56.11	2.1
SPT + Dependency	42.29	65.65	51.52	5.1	77.24	2.8
SPT extension + dependency Extension	43.71	64.65	52.24	4.8	79.19	2.6

Table 5
Effectiveness of different kernels and performance comparison with those of [21] on AImed. The weights of each individual kernel in combined kernels are in the parentheses after the kernel name.

	P	R	F	σ_F	AUC	σ_{AUC}
Feature-based kernel	46.32	61.1	52.69	3.6	80.71	2.7
BOW [21]			52.8		82.1	
Tree kernel	43.71	64.65	52.24	3.1	79.19	2.6
Tree kernel [21]			58.2		82.5	
Graph kernel	52.66	64.56	57.20	5.6	83.27	2.8
Graph kernel [21]			59.5		85.9	
Tree kernel (0.5) + feature-based kernel (0.5)	50.44	68.49	58.05	3.3	84.19	2.3
Tree kernel + BOW [21]			60.5		85.9	
Graph kernel (0.7) + feature-based kernel (0.3)	51.33	69.58	59.02	4.1	84.68	3.1
Graph kernel + BOW [21]			57.8		85.2	
Graph kernel (0.7) + tree kernel (0.3)	53.43	68.57	59.66	5.8	85.51	3.4
Tree kernel + graph kernel [21]			61.9		87.6	
Feature-based kernel (0.2) + Tree kernel (0.2) + graph kernel (0.6)	57.4	70.75	63.9	4.5	87. 83	2.9
Tree kernel + graph kernel + BOW [21]			60.8		86.8	
POS path kernel	31.89	62.77	42.39	4.3	69.59	3.2
Feature-based kernel(0.15) + tree kernel(0.15) + Graph kernel (0.6) + POS path kernel (0.1)	57.72	71.07	64. 41	4.3	88.46	2.8

testing of machine learning methods. It has recently been applied in numerous evaluations [15] and can be seen as an emerging de facto standard for PPI extraction method evaluation. This is followed by comprehensive evaluation of our method across five PPI corpora and the comparison of our results with earlier work.

4.2.1. *Effectiveness of different features in the feature-based kernel*

The number of features used for each type of word features is listed in Table 2. In our method, no feature selection is performed. We tried stemming, but found little decline in performance. The classification performances of different features in the feature-based kernel tested on AImed are shown in Table 3 (the precision, recall, and *F*-score values are achieved with the optimal threshold settings obtained from the 10-fold cross-validations).

For the feature-based kernel, an *F*-score of 50.82% and an AUC of 77.69% are achieved using only word features. With the introduction of protein names distance and keyword feature, the *F*-score and AUC improved to 52.69% and 80.71%, respectively. Compared with protein names distance feature, the keyword feature contributes more to the performance improvement (1.39 and 2.48 percentage units increase in *F*-score and AUC, respectively), as the latter employs domain knowledge, improving the performance. Exploiting domain knowledge may be a promising method to further improve PPI extraction performance. Similar works have been reported recently. Danger et al. defined a PPI ontology, PPIO, and showed some preliminary results guided by their novel approach [33]. Furthermore, He et al. proposed a novel framework of incorporating PPI ontology knowledge into the extraction from biomedical literature in order to address the emerging challenges of deep natural language understanding [34].

4.2.2. *Effectiveness of SPT parse tree, dependency tree and their extensions*

The performances of SPT parse tree, dependency path tree and their extensions tested on AImed are shown in Table 4. Using only SPT achieves an *F*-score of 50.13% and an AUC of 76.32%, while, after the introduction of SPT extension, dependency tree and its extension, the *F*-score and AUC are improved to 52.24% and 79.19%, respectively (2.11 percentage units increase in *F*-score and 2.87 in AUC). Though the performance of dependency tree kernel itself is poor (an *F*-score of 30.03% and an AUC of 56.11% after extension), when combined with SPT parse tree kernel, it can help improve the total performance, as evident in 0.68 percentage units increase in *F*-score (52.24–51.56%) and 1.14 percentage units' increase in AUC (79.19–78.05%).

In addition, as discussed in Section 3.2.1 and 3.2.2, SPT and dependency path tree extensions can improve the performance by including richer context information outside SPT and dependency path. Thus, they jointly contribute to the improvement of performance by almost 0.7 percentage units in *F*-score (52.24–51.52%) and 2 percentage units in AUC (79.19–77.24%).

4.2.3. *Effectiveness of different kernels*

The performance of different kernels tested on AImed is shown in Table 5, showing that graph kernel has superior performance compared to other three. As discussed in Section 3.4, the reason is that the graph kernel can treat the parser's output and word features at the same time. The performance of the feature-based kernel ranks second since it uses protein names distance and keyword feature in addition to words features; otherwise, with only words features, its performance (an *F*-score of 50.82% and an AUC of 77.69%) deteriorates below that of the tree kernel. The performance of the tree kernel is comparable to that of the feature-based kernel. Finally, POS path kernel performance is ranked the lowest, since it only utilizes the POS tag information.

The experimental results show that, when two or more individual kernels are combined, better performances are achieved. When the graph kernel is combined with the feature-based kernel, the performance is improved by 1.82 percentage units in *F*-score and 1.41 percentage units in AUC. When further combined with the tree kernel, the performance is improved by 4.88 percentage units in *F*-score and 3.15 percentage units in AUC. Finally, the addition of the POS path kernel improves the performance by 0.51 and 0.63 percentage units in *F*-score and AUC, respectively. As discussed in Section 3.5, since different kernels calculate the similarity between two sentences from different aspects, the combination of kernels covers more knowledge and is effective for PPI extraction.

The performance comparison between our kernels and those in [21] is also made in Table 5. Our feature-based kernel, tree kernel, and graph kernel correspond to the BOW, tree, and graph kernel in [21], respectively. Overall, each individual kernel in [21] achieves better performance than the corresponding one in our implementation. The performance of the BOW kernel in [21] is almost the same as our feature-based kernel in *F*-score (52.69% vs. 52.8%). The performance of the tree kernel in [21] is better than our tree kernel (58.2% vs. 52.4% in *F*-score and 82.5% vs. 79.19% in AUC); the reason is, as discussed in Section 3.2.2, that it uses the predicate type information to represent the dependency types. The performance of the graph kernel in [21] is also better than our graph kernel (59.5% vs. 57.2% in *F*-score and 85.9% vs. 83.27% in AUC), further discussed in Section 3.3.

However, unlike our results, the combination of kernels in [21] does not always contribute to performance improvement. For example, the graph kernel in [21] performs best, and combined with the tree kernel, achieves performance improvement of 2.4 percentage units in *F*-score and 1.7 percentage units in AUC. However, when further combined with the BOW kernel, the performance decreases by 1.1 percentage units in *F*-score and 0.8 in AUC. In fact, the performance deteriorates even when the graph kernel alone is combined with the BOW kernel. That shows that the introduction of the BOW kernel into the graph kernel leads to the deterioration of the performance. Similarly, the performance of the hybrid kernel in [19] is inferior to that of the individual walk kernel. The reason for this lack of improvement when individual kernels are combined may be that in [19] and [21] each kernel is assigned the same weight when combined. As discussed in Section 3.5, we found that only when the kernels with better performances are assigned higher weights can the combined kernel produce the best result. In our experiments, the weights for feature-based, tree, graph and POS path kernel are set to 0.6, 0.15, 0.15 and 0.1, respectively in the descending order of performance.

4.2.4. Performance on the individual corpora

In Table 6 we provide the performance comparison with the graph kernel approach presented in [18] and the combined kernel approach presented in [21] on the five corpora. For reference, the performance of the co-occurrence (or *alltrue*, which simply assigns each candidate into the interaction class) baseline is listed.

Since the graph kernel has been included into our combined kernel, our method achieves much better performances than those of graph kernel approach on all corpora apart from the IEPA corpus, on which the performances similar (75.73% vs. 75.1% in *F*-score and 84.72% vs. 85.1% in AUC).

Compared to the approach implemented in [21], our method achieves better performances with exception of the BioInfer corpus. On AImed, which can be seen as an emerging de facto standard for PPI extraction method evaluation, our method achieves an *F*-score of 64.41% and an AUC of 88.46%, 3.61 and 1.66 percentage units higher than those of the method in [21]. As discussed in more detail in Section 4.2.6, this level of performance is comparable to the state-of-the-art in machine learning-based PPI extraction.

The Precision, Recall and *F*-score performances of RelEx and AkanePPI system [35] are also listed in Table 6. Based on natural language preprocessing to produce dependency parse trees, RelEx system [8] applies a small number of simple rules to these trees to extract relations. AkanePPI [16] combines the version of the deep syntactic parser Enju that has been retrained on the GENIA corpus [36] with a shallow dependency parser [37]. A support vector machine with tree kernels [24] is used to extract rules for identifying pairs of interacting genes/proteins from the BioInfer corpus (referred to as AkanePPI(B) in [35]). Overall, the performances of RelEx and AkanePPI(B) are far below those of [18,21] and our method; however, AkanePPI(B) achieves the best performance on the BioInfer corpus, since it is trained on the corpus. In addition, on smaller corpora (HPRD50, IEPA and LLL), the performances of RelEx are comparable to those of [18]. The performance of the walk kernel presented in [19] is also listed in the last row of Table 6. The walk kernel consists of patterns of two vertices and their intermediate edge (vertex-walk or v-walk), as well as sequences of two edges and their common vertex (edge-walk or e-walk), extracted from the shortest path between two proteins in the graph. The walk kernel was only tested on the LLL corpus and obtained a 77.5 *F*-score.

Similar to [18], we also come to the conclusion that the *F*-score performance varies strikingly between the different corpora, with result on LLL approximately 20 percentage units higher than that on AImed (83.01–64.41%), whereas, for the distribution-invariant AUC measure, the performance for all of the corpora falls in the range of 84.74–88.46%. The results provide an argument in favor of applying the AUC metric instead of, or in addition to, *F*-score.

4.2.5. Cross-corpus performance

The types of named entities annotated, the definition of what exactly constitutes an interaction and the relative positive/negative distributions of pairs can vary significantly over different corpora. The cross-corpus evaluation aims to shed light on whether the learned models generalize beyond the specific characteristics of the data they were trained on [18]. We also explore this issue through a cross-corpus evaluation. Five extraction systems are trained, one on each corpus, and subsequently tested on the four remaining corpora.

Table 7 presents the cross-corpus performance comparison with [18,21] measured with AUC, while in Table 8 the optimal *F*-score results (selecting the positions from the precision/recall curves that would lead to highest *F*-scores) are given. The overall performance of our method is superior to that of [18] (in most cases, the AUCs and *F*-scores are more than 3 percentage units higher). In particular, the model trained on one of two small corpora HPRD50 and LLL tends to achieve better performance on another. For instance, the model trained on HPRD50 achieves an AUC of 74.9% and an *F*-score of 74.23% (10.9 and 6.33 percentage units higher in AUC and *F*-score, respectively than those of [18]) when tested on LLL, while the model trained on LLL achieves an AUC of 79.6% and an *F*-score of 67.77% (10.2 and 7.97 percentage units higher in AUC and *F*-score, respectively than those of [18]) when tested on HPRD50.

Our method achieves comparable performances to that of [21]. More specifically, when tested on smaller corpora (HPRD50, IEPA and LLL), our method can achieve better results than [21]: the models trained on all corpora – AImed, BioInfer, HPRD50, IEPA, and LLL – produce average performance improvements of 5.07, 1.83, 2.7, 1.5, and 4.35 percentage units in AUC and 4.71, 1.6, 1.87, 0.22, and 2.96 percentage units in *F*-score, respectively compared to those trained on the combination of HPRD50, IEPA and LLL.

In addition, our method achieves many similar results to those presented in [18]. Firstly, the performance varies significantly depending on the training and test corpus and, unlike in the single corpus cross-validation evaluations, the results are scattered, ranging from 64.8% to 87.2% AUC. On the large corpora, the performance of trained models is markedly inferior in all the cases compared to the single corpus evaluation performance. However, on the two small corpora HPRD50 and LLL, this is not the case. Performance on HPRD50 of the model trained on Aimed, and on LLL for the model trained on IEPA is comparable as the single corpus evaluation results. These results suggest that a larger amount of training data can compensate for the differences in corpus annotation strategies to a large extent. Secondly, through ranking the corpora separately according to the results on each of the other corpora, we also found that the performance is directly related to the size of the training data available. The largest corpus, BioInfer, performs best, followed by AImed, IEPA, HPRD50 and LLL, in deceasing order by data quantity.

4.2.6. Performance compared to other methods

The comparison with relevant results reported in related research is summarized in Table 9. The best performing system combines multiple layers of syntactic information by using a combination of multiple kernels based on several different parsers and achieves an *F*-score of 63.5% and an AUC of 87.9% [38]. Our method uses only the Stanford parser output to produce parse tree, dependency structure (path and graph) and POS path information, yielding a comparable performance. This is due to the following three key reasons: (1) With feature-based kernel, besides the commonly used word feature, protein names distance and keyword feature are introduced to improve the performance. In

Table 6
Performance comparison on the individual corpora with the graph kernel method [18] and the combined kernel in [21], RelEx [8], AkanePPI(B) [16] and the walk kernel [19] (only on LLL corpus). Counts of positive and negative examples in the corpora and precision, recall, *F*-score and AUC with standard deviations provided for *F*-score and AUC. The figures of RelEx and AkanePPI system are from [35]. GK: (G)raph (K)ernel method [18] and CK: the (C)ombined (K)ernel [21].

Corpus	Method	Statistics								co-occ.	
		#POS.	#NEG.	*P*	*R*	*F*	σ_F	AUC	σ_{AUC}	*P*	*F*
AImed	ours	1000	4655	57.72	71.07	64. 41	4.8	88.46	2.6		
	CK	1000	4834	55.0	68.8	60.8	6.6	86.8	3.3		
	GK	1000	4834	52.9	61.8	56.4	5.0	84.8	2.3	17.8	30.1
	RelEx	1000	4834	40	50	44					
	AkanePPI(B)	1000	4834	29.1	52.9	37.5					
BioInfer	ours	2512	6910	57.02	77.31	65.84	4.3	84.96	3.3		
	CK	2534	7119	65.7	71.1	68.1	3.2	85.9	4.4		
	GK	2534	7132	56.7	67.2	61.3	5.2	81.9	4.9	26.6	41.7
	RelEx	2534	7132	39	45	41					
	AkanePPI(B)	2534	7132	56.8	85.4	68.2					
HPRD50	ours	163	270	67.98	84.11	74.38	4.3	84.74	8.2		
	CK	163	270	68.5	76.1	70.9	10.3	84.6	6.3		
	GK	163	270	64.3	65.8	63.4	11.4	79.7	6.3	38.9	55.4
	RelEx	163	270	76	64	69					
	AkanePPI(B)	163	270	52	55.8	53.8					
IEPA	ours	335	482	70.57	82.06	75.73	4.5	84.72	4.0		
	CK	335	482	67.5	78.6	71.7	7.8	84.4	4.2		
	GK	335	482	69.6	82.7	75.1	7.0	85.1	5.1	40.8	57.6
	RelEx	335	482	74	61	67					
	AkanePPI(B)	335	482	66.2	51.3	57.8					
LLL	ours	164	166	79.10	87.61	83.01	7.8	87.0	3.5		
	CK	164	166	77.6	86.0	80.1	14.1	86.3	10.8		
	GK	164	166	72.5	87.2	76.8	17.8	83.4	12.2	55.9	70.3
	RelEx	164	166	82	72	77					
	AkanePPI(B)	164	166	76.7	40.2	52.8					
	Walk Kernel	164	166	72.5	83.3	77.5					

Table 7
Cross-corpus performance comparison with the graph kernel approach [18] and the combined kernel [21] measured with AUC. Rows correspond to training corpora and columns to test corpora. Δ denotes the average performance improvement of each model over that of the graph kernel approach. GK: (G)raph (K)ernel method [18] and CK: the (C)ombined (K)ernel [21].

	AImed			BioInfer			HPRD50			IEPA			LLL			Δ	Avg. rank
	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours		
AImed				67.7	74.8	74.1	82.4	81.5	84.2	76.1	77.4	82.7	77.8	77.3	84.1	5.27	2
BioInfer	77.8	80.2	79.6				75.2	81.1	82.7	79.3	81.8	84	83.3	83.5	85.2	3.97	1.5
HPRD50	72.5	75.5	73.8	61.8	68.5	70.4				74.9	78	78.9	64.0	75.4	74.9	6.2	3.25
IEPA	70.2	72.7	73.2	72.2	75.6	75.1	80.0	75.9	80.9				82.5	89.2	87.2	2.87	2
LLL	61.8	67.3	64.8	61.0	69.3	68.9	69.4	73.9	79.6	74.8	76.2	79.2				6.37	3.75

Table 8
Cross-corpus performance comparison with the graph kernel approach [18] and the combined kernel [21] measured with *F*-score and optimal thresholds. *F*-score results for cross-corpus testing with the optimal thresholds. Rows correspond to training corpora and columns to test corpora. Δ denotes the average performance improvement of each model over that of the graph kernel approach. GK: (G)raph (K)ernel method [18] and CK: the (C)ombined (K)ernel [21].

	AImed			BioInfer			HPRD50			IEPA			LLL			Δ	Avg. rank
	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours	GK	CK	Ours		
AImed				47.1	53.1	54.89	69.0	68.3	72.05	67.4	68.1	72.56	74.5	73.5	78.41	4.97	1.75
BioInfer	47.2	49.6	49.04				63.9	68.3	69.57	68.0	71.4	72. 27	78.0	76.9	79.56	3.33	1.75
HPRD50	42.2	43.9	43.32	42.5	48.6	47.88				65.1	67.8	69.51	67.9	72.2	74.23	4.31	3.25
IEPA	39.1	40.4	41.89	51.7	55.8	55.64	67.5	66.5	69. 05				77.6	83.2	80.87	2.86	2
LLL	33.3	38.6	34.99	42.5	48.9	49.41	59.8	64	67.77	64.9	65.6	67.75				5.35	3.75

Table 9
Comparison on AImed. Precision, recall, *F*-score and AUC results for methods evaluated on AImed with the correct cross-validation methodology.

Method	*P*	*R*	*F*	AUC
Our: combined kernel	57.72	71.07	64. 41	88.46
Miwa et al. [38]			63.5	87.9
Miwa et al. [21]	58.7	66.1	61.9	87.6
Miyao et al. [12]	54.9	65.5	59.5	
Giuliano et al. [29]	60.9	57.2	59.0	
Airola et al. [18]	52.9	61.8	56.4	84.8

particular, the introduction of keyword feature is a way of employing domain knowledge and proves to improve the performance effectively. With the appropriate features, feature-based kernel outperforms all other individual kernels. (2) Tree kernel can capture the structured syntactic connection information between the two entities. Our tree kernel combines the information of parse tree and dependency path tree and introduces their extensions to capture richer context information outside SPT and dependency path when necessary. In addition, POS path kernel is also introduced to capture syntactical and semantic information in POS tag path. (3) Different kernels calculate the similarity between two sentences from different aspects. Thus our combined kernel can reduce the risk of missing important features, yielding new useful similarity measures. More specifically, the weighted linear combination of individual kernel used instead of assigning the same weight to each individual kernel is experimentally proven to contribute to the performance improvement.

5. Conclusions

In this paper we present a multiple kernel learning-based approach to extracting PPI from biomedical literature. The approach combines four individual kernels – feature-based, tree, graph and POS path kernel – weighed according to their individual performance and achieves much better performance than each individual kernel. This indicates that the features in individual kernels are complementary and can be successfully combined in a hybrid kernel due to: (1) the flat entity information captured by the feature-based kernel; and (2) the structured syntactic connection information between the two entities captured by the tree kernel, graph kernel and POS path kernel. Experimental results show that our method can achieve state-of-the-art performance with respect to comparable evaluations, with 64.41% *F*-score and 88.46% AUC on the AImed corpus.

Similar to [18,21], a cross-corpus evaluation is performed to provide insight into the challenges involved in applying a system beyond the data it was trained on. Experimental evaluations show that the performances of our method are much better than those of graph kernel approach. Compared to [21], our method achieves comparable results and, when tested on smaller corpora, it shows superior performance. In addition, we come to some conclusions similar to those of [18]: (1) the performance is generally inferior than when training and testing on data from the same corpus, with exception of small corpora; and (2) a larger training data set can compensate for the differences in corpus annotation strategies to a large extent.

As discussed earlier, when combining individual kernels, assigning higher weights to the kernels with better performances can improve output quality. However, manual selection of appropriate weight for each kernel is a time-consuming and imperfect trial and error process. Therefore, finding an automatic method to combine the outputs of individual kernels to achieve the best final PPI classification result will be the focus of our future research. In addition, our experimental results show that, as a way of employing domain knowledge, the introduction of keyword feature proves to be effective in improving the performance. Therefore, introduction of more appropriate domain knowledge into PPI extraction is another problem to be studied.

Acknowledgments

This work is supported by grant from the Natural Science Foundation of China (Nos. 60673039 and 61070098), the National High Tech Research and Development Plan of China (No. 2006AA01Z151), the Fundamental Research Funds for the Central Universities (No. DUT10JS09) and Liaoning Province Doctor Startup Fund (No. 20091015).

References

[1] Zanzoni A, Montecchi-Palazzi L, Quondam M, Ausiello G, Helmer-Citterich M, Cesareni G. Mint: a molecular interaction database. FEBS Letters 2002;513(1):135–40.
[2] Bader G, Betel D, Hogue C. BIND – the biomolecular interaction network database. Nucleic Acids Research 2001;31(1):248–50.
[3] Xenarios I, Fernandez E, Salwinski L, Duan XJ, Thompson MJ, Marcotte EM, et al. DIP: the database of interacting proteins. Nucleic Acids Research 2000;28(1):289–91.
[4] Blaschke C, Valencia A. The frame-based module of the Suiseki information extraction system. IEEE Intelligent Systems 2002;17(2):14–20.
[5] Ono T, Hishigaki H, Tanigam A, Takagi T. Automated extraction of information on protein–protein interactions from the biological literature. Bioinformatics 2001;17(2):155–61.
[6] Corney DP, Buxton BF, Langdon WB, Jones DT. BioRAT: extracting biological information from full-length papers. Bioinformatics 2004;20(17):3206–13.
[7] Sekimizu T, Park HS, Tsujii J. Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. Genome Informatics Series: Workshop on Genome Informatics 1998;9:62–71.
[8] Fundel K, Küffner R, Zimmer R. RelEx – Relation extraction using dependency parse trees. Bioinformatics 2007;23(3):365–71.
[9] Xiao J, Su J, Zhou GD, Tan CL. Protein–protein interaction extraction: a supervised learning approach. In: Hahn U, Valencia A, editors. Proceedings of the first international symposium on semantic mining in biomedicine. Aachen Germany: CEUR; 2005. p. 10–3.
[10] Zhou D, He Y, Kwoh CK. Extracting protein–protein interactions from the literature using the hidden vector state model. In: Alexandrov V, Albada D, Sloot P, Dongarra J, editors. Proceedings of the international workshop on bioinformatics research and applications. Berlin, Germany: Springer; 2006. p. 718–25.
[11] Yang ZH, Lin HF, Li YP. BioPPISVMExtractor. a protein–protein interaction extractor for biomedical literature using SVM and rich feature sets. Journal of Biomedical Informatics 2010;43(1):88–96.
[12] Miyao Y, Sætre R, Sagae K, Matsuzaki T, Tsujii J. Evaluating contributions of natural language parsers to protein–protein interaction extraction. Bioinformatics 2009;25(3):394–400.
[13] Cristianini N, Taylor JS. An introduction to support vector machines and other kernel-based learning methods. New York, USA: Cambridge University Press; 2000.
[14] Bunescu RC, Mooney RJ. Subsequence kernels for relation extraction. In: Weiss Y, Schölkopf B, Platt J, editors. Advances in neural information processing systems 18. Cambridge, MA: MIT Press; 2006. p. 171–8.
[15] Moschitti A. Making tree kernels practical for natural language processing. In: Agirre E, Ravera SB, Pianta M, editors. Proceedings of the 11th conference of the European chapter of the Association for Computational Linguistics. Morristown, NJ, USA: Association for Computational Linguistics; 2006. p. 113–20.
[16] Sætre R, Sagae K, Tsujii J. Syntactic features for protein–protein interaction extraction. In: Baker C, Su J, editors. Proceedings of the 2nd international symposium on languages in biology and medicine. Aachen, Germany: CEUR; 2007. p. 6.1–6.14.
[17] Bunescu RC, Mooney RJ. A shortest path dependency kernel for relation extraction. In: Mooney RJ, editor. Proceedings of human language technology and empirical methods in natural language processing. Morristown, NJ, USA: Association for Computational Linguistics; 2005. p. 724–31.
[18] Airola A, Pyysalo S, Björne J, Pahikkala T, Ginter F, Salakoski T. All-paths graph kernel for protein–protein interaction extraction with evaluation of cross-corpus learning. BMC Bioinformatics 2008;9(Suppl. 11):S2.
[19] Kim S, Yoon J, Yang J. Kernel approaches for genic interaction extraction. Bioinformatics 2008;24(1):118–26.
[20] Nédellec C. Learning language in logic-genic interaction extraction challenge. In: Cussens J, Nédellec C, editors. Proceedings of the 4th Learning Language in Logic Workshop. 2005. p. 31–7.
[21] Miwa M, Soetre R, Miyao Y, Tsujii J. Protein–protein interaction extraction by leveraging multiple kernels and parsers. Journal of Medical Informatics 2009;78(12):e39–46.
[22] Bunescu RC, Ge R, Kate RJ, Marcotte EM, Mooney RJ, Ramani AK, et al. Comparative experiments on learning information extractors for proteins and their interactions. Artificial Intelligence in Medicine 2005;33(2):139–55.
[23] Collins M, Duffy N. Convolution kernels for natural language. In: Dietterich TG, Becker S, Ghahramani Z, editors. Advances in neural information processing systems 14. Cambridge, MA: MIT Press; 2002. p. 625–32.
[24] Moschitti A. A study on convolution kernels for shallow semantic parsing. In: Proceedings of the 42nd annual meeting on association for computational linguistics. Morristown, NJ, USA: Association for Computational Linguistics; 2004. p. 335–42.
[25] Zhang M, Zhang J, Su J, Zhou GD. A composite kernel to extract relations between entities with both flat and structured feature. In: Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the ACL. Morristown, NJ, USA: Association for Computational Linguistics; 2006. p. 825–32.
[26] Pyysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Järvinen J, et al. A corpus for information extraction in the biomedical domain. BMC Bioinformatics 2007;8(50):50.
[27] Ding J, Berleant D, Nettleton D, Wurtele E. Mining MEDLINE: abstracts, sentences, or phrases? In: Altman RB, Dunker AK, Hunter L, Lauderdale K, Klein TE, editors. Proceedings of pacific symposium on biocomputing. NJ, USA: World Scientific Press; 2002. p. 326–37.
[28] Pyysalo S, Airola A, Heimonen J, Björne J, Ginter F, Salakoski T. Comparative analysis of five protein–protein interaction corpora. BMC Bioinformatics 2008;9(Suppl. 3):S6.
[29] Giuliano C, Lavelli A, Romano L. Exploiting shallow linguistic information for relation extraction from biomedical literature. In: Agirre E, Ravera SB, Pianta M, editors. Proceedings of the 11th conference of the European chapter of the association for computational linguistics. Morristown, NJ, USA: Association for Computational Linguistics; 2006. p. 401–8.
[30] Landeghem SV, Saeys Y, de Peer YV, De Baets B. Extracting protein–protein interactions from text using rich feature vectors and feature selection. In: Salakoski T, Schuhmann DR, Pyysalo S, editors. Proceedings of the third inter-

national symposium on semantic mining in biomedicine. Turku, Finland: Turku Centre for Computer Sciences; 2008. p. 77–84.

[31] Hanley JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 1982;143(1):29–36.

[32] Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 1997;30(7):1145–59.

[33] Danger R, Rosso P, Pla F, Molina A. PPIEs: protein–protein interaction information extraction system. Procesamiento del lenguaje Natural 2008;40:137–43.

[34] He YL, Nakata K, Zhou D. Ontology-based protein–protein interactions extraction from literature using the hidden vector state model. In: Bonchi F, Berendt B, Giannotti F, Gunopulos D, Turini F, Zaniolo C, Ramakrishnan N, Wu X, editors. Proceedings of 2008 IEEE international conference on data mining workshops. NJ, USA: IEEE Computer Society; 2008. p. 736–43.

[35] Kabiljo R, Clegg AB, Shepherd AJ. A realistic assessment of methods for extracting gene/protein interactions from free text. BMC Bioinformatics 2009;10:233.

[36] Hara T, Miyao Y, Tsujii J. Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In: Bunt H, Merlo P, Nivre J, editors. Trends in parsing technology, text, speech and language technology, vol. 43. Berlin, Germany: Springer; 2011. p. 257–75.

[37] Sagae K, Tsujii J. Dependency parsing and domain adaptation with LR models and parser ensembles. In: Eisner J, editor. Proceedings of the CoNLL shared task session of EMNLP-CoNLL 2007. Morristown, NJ, USA: Association for Computational Linguistics; 2007. p. 1044–50.

[38] Miwa M, Sætre R, Miyao Y, Ohta T, Tsujii J. Combining multiple layers of syntactic information for protein–protein interaction extraction. In: Salakoski T, Schuhmann DR, Pyysalo S, editors. Proceedings of the third international symposium on semantic mining in biomedicine. Turku, Finland: Turku Centre for Computer Sciences; 2008. p. 101–8.