Journal of Biomedical Informatics 45 (2012) 535–543

Contents lists available at SciVerse ScienceDirect

Journal of Biomedical Informatics

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

Tree kernel-based protein–protein interaction extraction from biomedical literature

Longhua Qian, Guodong Zhou *

NLP Lab, School of Computer Science and Technology, Soochow University, 1 Shizi Street, Suzhou 215006, China

ARTICLE INFO

Article history:
Received 4 August 2011
Accepted 15 February 2012
Available online 25 February 2012

Keywords:
Protein–protein interaction
Convolution tree kernel
Constituent parse tree
Shortest dependency path

ABSTRACT

There is a surge of research interest in protein–protein interaction (PPI) extraction from biomedical literature. While most of the state-of-the-art PPI extraction systems focus on dependency-based structured information, the rich structured information inherent in constituent parse trees has not been extensively explored for PPI extraction. In this paper, we propose a novel approach to tree kernel-based PPI extraction, where the tree representation generated from a constituent syntactic parser is further refined using the shortest dependency path between two proteins derived from a dependency parser. Specifically, all the constituent tree nodes associated with the nodes on the shortest dependency path are kept intact, while other nodes are removed safely to make the constituent tree concise and precise for PPI extraction. Compared with previously used constituent tree setups, our dependency-motivated constituent tree setup achieves the best results across five commonly used PPI corpora. Moreover, our tree kernel-based method outperforms other single kernel-based ones and performs comparably with some multiple kernel ones on the most commonly tested AIMed corpus.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

Since determining protein interaction partners is crucial for understanding both the functional role of individual proteins and the organization of the entire biological process, there is a surge of research interest in protein–protein interaction (PPI) extraction. However, manual collection of PPI information from thousands of biomedical research papers published every day (e.g. MEDLINE) is so time-consuming and labor-demanding that automatic extraction approaches with the help of NLP techniques become necessary.

In principle, the task of PPI extraction is much like the semantic relation extraction subtask (also called Relation Detection and Classification, RDC) defined by the ACE project[1] in the newswire domain. Therefore, various kinds of machine learning methods have been borrowed from the newswire domain to the biomedical domain: feature-based methods [1–5] and kernel-based methods [6–10].

Early studies on PPI extraction employ feature-based methods, which extract various lexical, syntactic and semantic features from the sentence containing two involved proteins and make use of a machine learner to extract PPI instances. However, it is often difficult for feature-based methods to effectively represent the structured information in a constituent- or dependency-based syntactic representation without the burden of feature engineering, though this kind of structured information is essential to identify the semantic relationship between two proteins. As a remedy, Bui et al. [5] propose a two-stage hybrid approach, which first categorizes the data into subsets according to its semantic and syntactic structured properties, and then extracts various common features as well as those ones related to specific subsets for PPI detection.

With the wide adoption of kernel-based methods to many NLP tasks, such as for semantic relation extraction and semantic role labeling, various kernels, most of which are related to dependency information derived from sentences, have been applied to PPI extraction due to their capability to search implicitly high dimensional feature spaces. Bunescu and Mooney [6] adopt a generalized substring kernel over a mixture of words and word classes to extract protein interactions from biomedical corpora as well as semantic relations from newswire corpora. Erkan et al. [7] define two similarity functions based on cosine similarity and edit distance among dependency paths between two entities, and then incorporate them in semi-supervised learning for PPI extraction using SVM and KNN classifiers. Airola et al. [8] introduce an all-dependency-paths graph kernel to capture complex dependency relationships between words and obtain a significant performance boost albeit at the expense of computational complexity. Kim et al. [9] propose a walk-weighted subsequence kernel based on shortest dependency paths to explore various substructures such as e-walks, partial matching, and non-contiguous paths. Chowdhury et al. [10] investigate the effect of mildly extended dependency trees on PPI extraction using an unlexicalized partial tree kernel.

* Corresponding author. Address: Soochow University, 1 Shizi Street, Mailbox 158#, Suzhou 215006, China.
E-mail address: gdzhou@suda.edu.cn (G. Zhou).

[1] http://www.ldc.upenn.edu/Projects/ACE/.

1532-0464/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2012.02.004

On one hand, these dependency-based kernels suggest that dependency information plays a critical role in PPI extraction, much like it does in semantic relation extraction in newswire narratives [11,12]. On the other hand, while tree kernels based on constituent parse trees achieve great success in semantic relation extraction [13–15] and semantic role labeling [16,17] from the newswire narratives, they have not been fully explored for PPI extraction in the biomedical domain. Particularly, Zhang et al. [13] discover that the Shortest Path-enclosed Tree (SPT) achieves the best performance among five tree setups. Zhou et al. [14] further extend it to Context-Sensitive SPT (CS-SPT), which includes necessary predicate-linked path information. Qian et al. [15] propose to determine the appropriate part of a constituent parse tree by hand-crafted heuristics. However, the initial attempt to adopt a single kernel over constituent parse trees [18] for PPI extraction does not show promising results, and its combination with a bag-of-words kernel achieves mediocre performance [21]. Considering the similarity between the task of PPI extraction in the biomedical domain and that of relation extraction in the newswire domain, one question naturally arises: "How can kernel-based PPI extraction benefit from the constituent parse tree structure?"

To address the question, this paper presents a principled way to automatically generate a precise and concise constituent parse tree representation for kernel-based methods, motivated by the success of employing dependency information in PPI extraction and the rich structural representation power of constituent parse trees in semantic relation extraction. This is done by taking advantage of the shortest dependency path between two involved proteins in the dependency graph structure of a sentence. Specifically, only the words appearing on the shortest dependency path and their associated constituents in the constituent parse tree are considered as necessary and thus kept as the essential part of the constituent parse tree. In this paper, we refer to this constituent parse tree as shortest dependency path-directed constituent parse tree (SDP-CPT) since its construction is guided by the shortest dependency path. Experimental results on five major PPI corpora show the effectiveness of dependency-directed constituent structure representation and its advantage over other state-of-the-art structure representations.

2. Methods

This section first illustrates the limitations of previously used constituent parse tree setups, then focuses on the importance and various encoding schemes of shortest dependency paths, and finally presents the shortest dependency path-directed constituent parse tree (SDP-CPT).

2.1. Limitations of current constituent parse tree setups

It is widely acknowledged that the key problem for the success of tree kernel-based semantic relation extraction is how to represent the constituent parse tree in a precise and concise manner. Zhang et al. [13] explore five kinds of tree setups and find that the Shortest Path-enclosed Tree (SPT) achieves the best performance. However, unlike the locality of semantic relations in the newswire domain [19], most of PPI instances in the biomedical domain span a relatively long distance, leading to the complexity and diversity of PPI instances [20]. Therefore, it is not surprising that previous tree kernels over constituent parse trees have not yet achieved promising results for PPI extraction just as they do in the news domain. Tikk et al. [18] extensively compare different kernel-based methods on PPI extraction and show that the tree kernel over the constituent parse tree only achieves F1 of 34.6 on the AIMed corpus. Miyao et al. [21] conduct a comprehensive comparison of different syntactic representations for PPI extraction and find that the phrase structure tree in the form of the constituent parse tree (in the PTB style in their paper) combined with bag-of-words performs significantly worse than other representations. Actually, our preliminary experiment on PPI extraction via the convolution tree kernel over SPT only achieves F1 of about 47 on the AIMed corpus (c.f. Table 4 in Section 3.3). Such low performance can be demonstrated to a certain degree via a typical example as illustrated in Fig. 1, where the interaction between PROT1 and PROT2 (their actual names as well as other protein names have been blinded in the constituent parse tree) can only be determined by the overall constituent structure of the sentence. Obviously, SPT fails to identify this interaction instance since SPT ignores the constituents outside the shortest path connecting two proteins (Fig. 1: T_2: SPT).

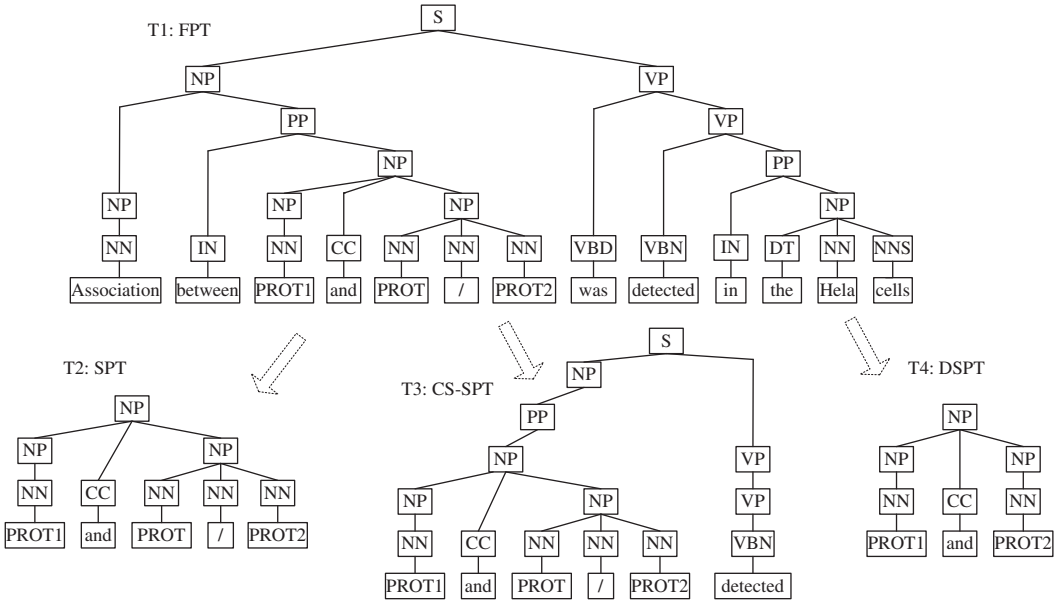

Fig. 1. Different tree setups for a PPI instance between cdc25A and cdc2 from the sentence "Association between cdc25A and cyclin B1/cdc2 was detected in the HeLa cells." (AIMed.d68.s581).

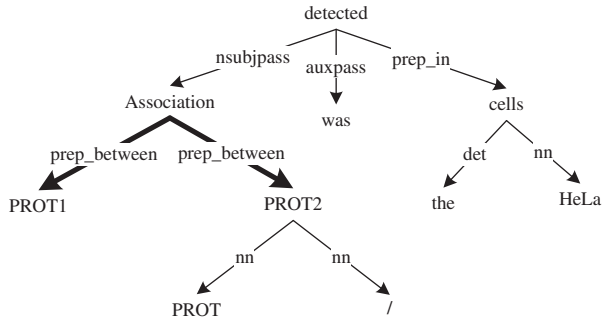

Fig. 2. Dependency parse tree corresponding to the sentence in Fig. 1.

For the Context-Sensitive SPT (CS-SPT) proposed by Zhou et al. [14], which extends necessary predicate-linked path information outside SPT, some critical information is still missing while there is some noisy information. This can be shown in Fig. 1 (T_3: CS-SPT). Although the word "detected" and its associated constituents are added, the more important portion of "association between" and their associated constituents are still missing while the noisy words "PROT /" still remaining.

In order to overcome the shortcomings in SPT and CS-SPT, Qian et al. [15] propose a dynamic syntactic parse tree (DSPT) by exploiting constituent dependencies to refine the constituent parse tree. Specifically, they manually devise five categories of constituent dependencies, motivated by various kinds of lexical dependency. When refining each node along the shortest path in the constituent parse tree, these constituent dependencies are used to determine how to remove or reduce irrelevant or redundant constituents, eventually leading to a more precise and concise parse tree structure. Nevertheless, this tree structure still suffers from the following three shortcomings:

- It fails to include the constituents beyond the lowest common ancestor to the tree root, similar to CS-SPT as proposed in Zhou et al. [14]. This may not be a serious problem for semantic relation extraction due to the locality of most semantic relations defined in the ACE RDC corpus, but it certainly is for PPI extraction.
- The rules adopted to utilize constituent dependencies are manually crafted and thus are not easily adapted to other domains and languages. For example, while the constituent dependencies related to noun phrases are effective in the newswire domain (e.g. the ACE RDC corpus), this may not be true for PPI extraction in the biomedical literature.
- The constituent dependencies have been divided into only five categories. Such division may be too coarse to reflect the substantial difference between various kinds of dependencies (considering there are 55 kinds of minor-typed dependencies in the Stanford Dependency representation).

In this paper, we attempt to address these problems by employing the shortest dependency path in the dependency parse tree for reshaping the constituent parse tree in a principled way in the context of PPI extraction from the biomedical literature.

2.2. Shortest dependency path

Lexical dependencies can capture both local- and long-range relationships among words occurring in the same sentence. Such dependency relationships offer a condensed representation of the structured information necessary to assess the relationship between two proteins. In order to capture the necessary information inherent in the dependency parse tree for extracting PPI instances, various kernels based on dependency paths, such as edit distance kernel [7], all-dependency-path graph kernel [8], and walk-weighted subsequence kernels [9] have been proposed. Likewise for semantic relation extraction in the newswire domain, the kernels on dependency trees [11] and the shortest dependency path [12] have been proposed. One common characteristic of these kernels is that they all contain the shortest dependency path between two proteins and sometimes assign more weights to this path than to other ones [8]. This implies the importance of the shortest dependency path over other paths in the dependency tree or the dependency graph. Fig. 2 depicts the dependency tree in the Stanford Dependency scheme corresponding to the sentence in Fig. 1 with the shortest path highlighted with bold lines.[2] The shortest dependency path between "PROT1" and "PROT2" ("PROT1 → prep_between → association ← prep_between ← PROT2") demonstrates that this kind of information is very helpful to PPI extraction.

Currently, there are two widely used dependency representation schemes, viz. CoNLL scheme (adopted by CoNLL'2007 and CoNLL'2008 shared tasks) [22,23] and Stanford Dependency (SD) scheme (adopted by the Stanford parser) [24]. These two schemes differ significantly in the representation of passive construction, position of auxiliary and modal verb, or coordination. It is generally acknowledged that the Stanford Dependency scheme is closer to the targeted semantic representation from the perspective of information extraction [25]. Particularly, among the four styles of Stanford representations (basic, collapsed, CCprocessed and collapsedTree), CCprocessed ("collapsed dependencies with propagation of conjunct dependencies") can much simplify patterns in relation extraction since dependencies involving prepositions as well as referent of relative clauses are effectively collapsed to reflect direct dependencies between content words, and dependencies involving conjuncts are propagated in order to get additional dependencies.

In general, dependency relationships can be generated in two different ways. One is to directly apply native dependency parsers such as GDep [26] or MST [27] to produce CoNLL or SD style outputs. The other is first to generate PTB-like constituent parse trees using phrase-based parsers such as Stanford parser [28] or Berkeley parser [29], and then convert them into CoNLL or SD style representations using some Treebank conversion script. Miyao et al. [21] empirically show that the performance scores of PPI extraction using bag-of-words features plus dependency parse trees or constituent parse trees derived from different parsers are significantly different. Miwa et al. [30] find that types and structures of different dependency representations have specific advantages and disadvantages for the event extraction task. Buyko and Hahn [25] also demonstrate that the dependency graph representation has a crucial impact on the achievement level of IE task, particularly for event extraction in the biomedical domain. Therefore, when using dependency information for refining the constituent parse trees, various combinations of these parsers and dependency representations need to be compared in order to find the most appropriate one for PPI extraction in the biomedical domain (c.f. Section 3.3).

2.3. SDP-CPT: shortest dependency path-directed constituent parse tree

Considering the importance of the shortest dependency path in PPI extraction and the effectiveness of employing dependency

[2] To avoid cycles in the tree, the "conj_and" typed dependency between PROT1 and PROT2 has been removed since it cannot provide any useful clue to the interaction.

Algorithm GeneSDP-CPT

Input: a sentence *S* and two proteins *PROT1* and *PROT2* in it

Output: an *SDP-CPT*

Steps:

1) Given *S*, generate the constituent parse tree *CPT* using a constituent parser, and the dependency graph *DG* using a dependency parser
2) Extract the shortest constituent path (SCP) between *PROT1* and *PROT2* from *CPT*, denoted as $SCP=\{scp_i\}$, $i=1,\ldots,L$ with its root scp_r
3) Construct the shortest dependency path (SDP) between *PROT1* and *PROT2* from *DG*. The path and its sequence of dependency types are denoted as follows:
   $SDP=\{sdp_i\}$, $i=1,\ldots,N$, where $sdp_1=PROT1$, $sdp_N=PROT2$
   $SDT=\{sdt_i\}$, $i=2,\ldots,N$
4) Set *SDP-CPT* = *SCP*
5) For each sdp_i in $SDP \setminus \{PROT1, PROT2\}$, Repeat
   (a) Find in *CPT* the leaf node n_w corresponding to sdp_i
   (b) Add the path connecting n_w and scp_r into *SDP-CPT*
   (c) If the dependency type sdt_i for sdp_i is "prep_*xx*", such as "prep_*of*", then
      i. Extract the word *xx* from sdt_i
      ii. Find in *CPT* the node n_{xx} corresponding to *xx*
      iii. Add the path connecting n_w and scp_r into *SDP-CPT*
   End if
6) Merge two consecutive NP/VP nodes along the *SDP-CPT* paths into a single one, where the parent node has only one child node.
7) Return *SDP-CPT*

Fig. 3. Algorithm for generating the SDP-CPT.

information to refine the constituent parse tree for tree kernel-based semantic relation extraction in the newswire domain, it is a natural idea to automatically generate the proper constituent parse tree with the help of the shortest dependency path. Specifically, one can reshape the constituent parse tree by making use of the shortest dependency path between two proteins. Fig. 3 describes the algorithm (GeneSDP-CPT) to generate the shortest dependency path-directed constituent parse tree (SDP-CPT).

Note that in Step 3 the first and last elements of the shortest dependency path *SDP* are *PROT1* and *PROT2* respectively while every element sdt_i in *SDT* denotes the dependency type between sdp_i and sdp_{i-1}. In Step 5 the repetition should be done for all the nodes on *SDP* except *PROT1* and *PROT2* since they have already been included in *SDP-CPT*. Step 5(b) and Step 5(c)-iii connect a single word to *SDP-CPT*, that is, add the word node and its associated constituents into *SDP-CPT* to retain the integrality of *SDP-CPT*. This often occurs in two scenarios, one is when the word node is enclosed by *SCP*, the associated constituents are its ancestors up to *SCP*; the other is when the word node is outside *SCP*, then the associated constituents form a path connecting the word node to the root of *SCP*. In addition, Step 5(c) is particularly necessary, since a dependency relation of the type "prep_*xx*(*governor*, *dependent*)" in the case of the SD collapsed/CCprocessed style implies two relations, one between *xx* and *dependent* and the other between *governor* and *xx*, which are both important to PPI extraction and therefore should be recovered consequently. However, in other uncollapsed cases this step is skipped. Finally, merging in Step 6 is performed on two consecutive NP/VP nodes along the SDP-CPT path when the upper node has only one child node. This step is partly motivated by recursive NPs which contain another NP as their child and will be simplified to non-terminal nodes with the single in and out arcs when they are trimmed. Merging of these nodes will make the tree structures for inter-protein interactions more general, though slightly affect the precision. The similar reason may also apply to the merging of two VP nodes.

In order to visually demonstrate the process of generating an SDP-CPT, we take the sentence and the two proteins shown in Fig. 1 as an example. Fig. 4 illustrates the detailed generation process of this instance.

First, the shortest dependency path (SDP) and the shortest constituent path (SCP) derived from the Stanford parser are generated as shown in Fig. 4a and b respectively. Then, each word in the SDP together with its associated constituents is added into the SCP. Here, since there is only one word "Association" in SDP besides PROT1 and PROT2, the word "Association" needs to be connected to the SCP, thus leading to a path "Association → NN → NP → PP → NP → NP" being added as indicated by the dashed lines. Hereafter, since the dependency type between "PROT1" and

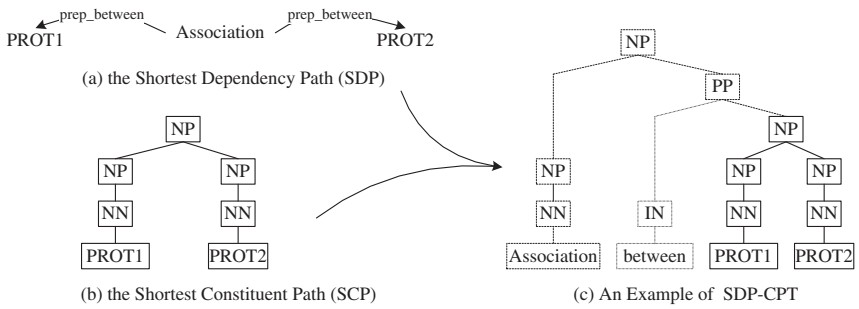

Fig. 4. An example of SDP-CPT.

"Association" is prep_*between*, the preposition word "*between*" and its associated constituents are also added into the SCP as indicated by the dotted lines. No further post-processing is necessary in this example, so SDP-CPT is eventually formed. Compared to other tree setups in Fig. 1, namely SPT, CS-SPT and DSPT, obviously SDP-CPT is much more concise and precise for this PPI instance.

3. Results and discussion

This section systematically evaluates the performance of our shortest dependency path-directed constituent parse tree (SDP-SPT) on PPI extraction across several major PPI corpora.

3.1. Data sets and preprocessing

In order to fairly compare our work with other PPI extraction systems, we use five PPI corpora, namely, AIMed [20], BioInfer [31], HPRD50 [32], IEPA [33] and LLL [34], as our benchmark data sets. These corpora are converted to a unified format from the original versions by Pyysalo et al. [35] and made publicly available for academic research.[3] Most of the evaluation is done on the widely used AIMed corpus, which contains 177 Medline abstracts with PPI instances, and 48 abstracts without any PPI instances. In total, there are 4084 protein references and around 1000 annotated protein–protein interactions in this data set. The exact numbers of positive instances and negative instances in these corpora are shown in the first row in Table 4.

In this paper, a potential PPI instance is generated for any pair of two proteins in a sentence. That is, if a sentence contains n proteins, $\binom{n}{2}$ protein pairs are generated. In particular, for the AIMed corpus all the self-interactions (59 instances) are removed and all the PPI instances with nested protein names are retained (154 instances), as adopted in most literature. Eventually, 1000 positive instances and 4834 negative instances are generated. For other four corpora, the processing procedure is similar except that no self-interactions are found. Besides, for a potential PPI instance, the two involved proteins are replaced by PROT1 and PROT2 respectively while other proteins are replaced by PROT in order to blind the learner for fair comparison with other work. Finally, all the sentences in these corpora are parsed using the Stanford parser or the Berkeley parser to generate the constituent parse trees, and various native dependency parsers or a conversion script are used to generate the dependency graphs.

3.2. Classifier and evaluation metrics

In our experimentation, we select Support Vector Machines (SVMs) as the classifier since SVM represents the state-of-the-art in the machine learning research community. In particular, we use the SVM^{light} [36] with the convolution tree kernel SVM^{light}–TK [37][4] to induce the model from the training dataset and then classify the test dataset. The convolution tree kernel [38] computes the similarity between two constituent parse trees in terms of the number of their common subtree structures. Particularly, we adopt the SST (Subset Trees) kernel instead of the ST (SubTrees) kernel and the PT (Partial Trees) kernel, similar to other studies [13–15,39] on relation extraction, since the first one yields better performance than the others.

Evaluation is done using 10-fold document-level cross-validation, each of which contains 90% of documents as the training data and the remaining 10% as the test data. Particularly, for the AIMed corpus we apply the exactly same 10-fold split as widely used in relevant studies. Following conventions, the parameters C for SVM is set to the ratio of negative instances to positive ones in respective corpora, and λ for the convolution tree kernel is set to default 0.4. Furthermore, the One Answer per Occurrence in the Document (OAOD) strategy is adopted, which means that the correct interaction must be extracted for each occurrence. This guarantees the maximal use of the available data, and more importantly, allows fair comparison with relevant work. All the experiments are evaluated using commonly used Precision (P), Recall (R) and harmonic F-score (F1). As an alternative to F1-score, the AUC (*area under the receiver operating characteristics curve*) score which is invariant to the class distribution of the test dataset, is also provided for comparison.

Finally, in order to determine whether an improvement of performance is statistically significant or not, we perform approximate randomization tests similar to [40] using a Perl script adapted from Randomized Parsing Evaluation Comparator.[5] The null hypothesis is that the two different settings under consideration have the same performance of F1. Conventionally, the null hypothesis is rejected for values of p less than or equal 0.05.

Table 1
Performance comparison on the AIMed corpus with different lengths of dependency paths using all kinds of dependency types (the SD CCprocessed style).

Length	P (%)	R (%)	F1	AUC
SPT (baseline)	57.0	40.7	47.1	79.9
SCP + L0 (SCP)	45.0	19.5	26.5	67.9
SCP + L1	59.7	45.8	51.4	80.2
SCP + L2	59.2	51.7	55.0	82.3
SCP + L3	58.0	51.9	54.6	82.2
SCP + L4	59.3	54.0	56.2	82.6
SDP-CPT	58.2	55.8	**56.9**	**82.7**

3.3. Experimental results

3.3.1. Comparison of different lengths of dependency paths on the AIMed corpus

Table 1 reports the performance of PPI extraction on the AIMed corpus with different lengths of the dependency paths used for generating structured representation, where constituent parse trees are generated using the Stanford parser and dependency paths in the SD CCprocessed style are further derived from these constituent parse trees, since this combination seems most suitable for PPI extraction (c.f. Section 2.2). (We will compare the impact of different dependency styles in the next experiment.) Here, two partial dependency paths on SDP, starting from each of the two proteins respectively, are utilized to direct the generation of the constituent parse tree representation. The length of these two partial paths is shown in the 1st column. The words corresponding to the nodes on these two dependency paths together with these words' associated constituents in the constituent parse tree are added to SCP. For example, the length of 0 (L0) means that no word and constituent will be added to SCP, while the length of 1 (L1) means that the words corresponding to the governors of two proteins on SDP and these words' associated constituents in the constituent parse tree are added to SCP. Particularly, the performance of the SPT setup is also listed as the baseline for comparison. The best performance scores of F1 and AUC among all setups are displayed in bold (here SDP-CPT). Approximate randomization tests are conducted between F1 of SDP-CPT and other settings. The settings that are statistically different from the best F1 (i.e., $p \leqslant 0.05$) are underlined.

[3] http://mars.cs.utu.fi/PPICorpora/GraphKernel.html.
[4] http://ai-nlp.info.uniroma2.it/moschitti/.
[5] http://www.cis.uppen.edu/~dbikel/software/html#comparator.

Table 2
Performance comparison on the AIMed corpus using SDP-CPT with different kinds of dependency encoding schemes and dependency paths derived from different parsers.

Parser	SD Basic		SD collapsed		SD CCprocessed		CoNLL	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC
Stanford	52.9	80.8	52.1	80.6	**56.9**	**82.7**	53.6	82.0
Berkeley	53.1	82.2	52.5	81.7	**55.4**	**83.6**	53.5	82.1
GDep	–	–	–	–	–	–	52.0	81.3
MST	–	–	–	–	–	–	53.1	81.7

This table shows that the constituent parse tree directed by the shortest dependency path (SDP-CPT) achieves the best performance of 58.2/55.8/56.9/82.7 in P/R/F1/AUC, statistically outperforming SPT by 9.8 units in F1 and 2.8 units in AUC largely due to the substantial increase in recall. This suggests that SDP-CPT can recover much useful information to SPT while removing some noise from SPT. It also shows

- The performance of the SCP corresponding to the length of 0 is lowest, since they contain no information derived from the shortest dependency path.
- With the increase of the length of dependency paths, more and more useful information derived from SDP is included in the constituent parse tree and the performance reaches the highest for SDP-CPT (all the words corresponding to all the nodes on the SDP and their associated constituents are added).

In summary, the above results suggest that, using the constituent parse tree and the corresponding dependency path derived from the Stanford parser, SDP-CPT can achieve the best performance. Therefore, all the subsequent experiments adopt the SDP-CPT setup unless otherwise specified.

3.3.2. Comparison of different kinds of dependency encoding schemes derived from various parsers on the AIMed corpus

We compare in Table 2 the impact of four different kinds of dependency encoding schemes with two commonly used constituent parsers (Stanford and Berkeley) and two native dependency parsers (GDep and MST) generating the short dependency path while retaining constituent parse trees derived from the Stanford parser. Specifically, the SD style representations of both Stanford and Berkeley are derived using the Stanford parser while their CoNLL style representations are converted using PennConverter.[6] GDep and MST parsers directly generate CoNLL-style dependency representations. Since the conversion from the native dependency-derived CoNLL style to the SD style is usually inaccurate and lossy, leading to poor performance, we do not include these results here. This table shows that the SD CCprocessed style representation derived from the Stanford parser performs best with F1/AUC of 56.9/82.7 (in bold fonts), while the same style representation derived from the Berkeley parser perform comparably with F1/AUC of 55.4/83.6 (in bold fonts). We also conduct experiments using these dependency settings on the Berkeley-derived constituent parse trees and find that they perform much lower than those on the Stanford-derived constituent parse trees (e.g., the F1/AUC scores for the Berkeley parse tree with Stanford- and Berkeley-derived dependency paths are 53.8/81.0 and 51.6/80.8 respectively[7]). Additionally, the table shows that the CoNLL style representations derived from both native dependency parsers and constituent-based parsers perform worse than the SD CCprocessed style, most probably due to the latter's richness in representing semantic relations between words. These experimental results further justify the effectiveness of the shortest dependency path in the SD CCprocessed style for refining constituent parse trees.

3.3.3. Contribution of different types of Stanford dependencies on the AIMed corpus

Table 3 compares the contribution of various kinds of dependencies used for generating SDP-CPT on the AIMed corpus. All the typed dependency relations are grouped into 4 major classes, namely *Modifier*, *Argument*, *Conjunction* and *Others*. For every major type, minor dependency types, if any exists, are further grouped in the order of their potential importance. The percentage of each minor type with respect to all dependency types being employed when generating the SDP-CPT is listed in Column 2. Particularly, the tree setup without using any dependency type, which corresponds to that with the length of 0 (SCP + L0) in Table 1, is shown at the first row. Furthermore, the dependency types are added in two different ways:

- *Individual*: the dependency types are used individually with their performance scores shown inside the parentheses.

- *Accumulative*: the dependency types are incrementally used one by one with their performance scores shown outside the parentheses. The "+" sign before the type means that its use can boost the performance in F1 or AUC score and thus will be passed down to the next iteration.

Table 3 shows that with the use of all *Argument* types and all *Modifier* types, the SDP-CPT attains the best performance of 59.1/57.6/58.1/83.3 in P/R/F1/AUC as shown in bold fonts, outperforming the SDP-CPT with all dependency types used (58.2/55.8/56.9/82.7 of P/R/F1/AUC in the last row of Table 1). Approximate randomization tests of F1 are conducted between two consecutive settings in the accumulative mode, and the types statistically distinguished from the latter one are underlined, which means that using the latter type statistically improves the performance. This table also shows

- The dependency types of *subj*, *obj*, *nn* and *prep* yield statistically substantial performance improvement both in the accumulative mode and in the individual mode.
- The dependency types of *Conjunction* and *Others* harm the performance in the accumulative mode, although *Conjunction* improves the performance in the individual mode.
- It is interesting to note that while the dependency types of *arg-others* and *mod-others* harm the performance in the individual mode, they slightly, though not statistically significantly, improve the performance in the accumulative mode.

Since the governors of *subj* and *obj* types are verbs, those of the *prep* type are nouns or prepositions, and those of the *nn* type are nouns, the above results are consistent with our observation that some verbs like "bind" or "interact", some prepositions like "with" or "of", and some nouns like "interaction" or "expression", which are on the shortest dependency path between two proteins, are particularly important for PPI extraction. Therefore, in the following experiments all the *Argument* and *Modifier* types are included while the *Conjunction* and *Others* types are excluded. Here, we notice that the best way to choose the effective dependency types is via a development set other than the cross-validation strategy we currently adopt, that is, a disjoint development set should be set aside to determine the effectiveness of each dependency type. However, as the corpus size is relatively small, the allocation of a

[6] http://nlp.cs.lth.se/software/treebank_converter/.

[7] Probably because the English Factorized PCFG model we use for the Stanford parser already involves the GENIA corpus, the Stanford parser outperforms the Berkeley parser in the biomedical domain, though the latter outperforms the former in intrinsic evaluations [41].

Table 3
Contribution of different typed dependencies on the AIMed corpus with the SDP-CPT setup in the accumulative mode (outside parentheses) and in the individual mode (inside parentheses).

Typed dependency	%	P (%)	R (%)	F1	AUC
SCP	–	45.0	19.5	26.5	67.9
Argument					
+subj	10	52.5 (52.5)	33.2 (33.2)	40.4 (40.4)	72.7 (72.7)
+obj	31	54.7 (53.8)	42.4 (42.4)	47.4 (47.0)	76.6 (76.6)
+arg-others	2	54.5 (48.8)	44.9 (14.6)	48.7 (21.3)	76.5 (68.6)
Modifier					
+nn	10	56.2 (54.9)	48.2 (38.5)	51.7 (44.6)	81.4 (77.5)
+prep_*xx*	20	58.2 (53.4)	55.2 (39.2)	56.6 (44.8)	83.1 (76.2)
+mod-others	5	**59.1** (46.8)	**57.6** (15.7)	**58.1** (22.3)	**83.3** (67.3)
Conjunction	12	58.9 (48.9)	55.0 (23.5)	56.7 (30.5)	82.8(69.8)
Others	10	58.4 (47.5)	53.8 (14.8)	55.8 (20.4)	83.0 (69.5)

Table 4
Comparison of F1 (outside parentheses) and AUC (inside parentheses) between SDP-CPT and different tree setups across major PPI corpora.

Tree setups	AIMed	BioInfer	IEPA	HPRD50	LLL
Ratio of POS/NEG	1000/4834	2534/7119	335/482	163/270	164/166
MCT	41.0 (78.0)	50.8 (76.7)	54.4 (78.6)	48.0 (73.4)	77.1 (73.4)
SPT	47.1 (79.9)	53.4 (73.7)	64.6 (82.2)	64.9 (81.6)	79.4 (86.1)
CS-SPT	46.5 (80.2)	54.5 (74.5)	65.0 (81.0)	63.6 (79.9)	80.1 (86.0)
DSPT	50.0 (77.8)	58.3 (78.5)	65.6 (80.9)	66.1 (80.3)	77.3 (79.3)
SDP-CPT	**58.1** (**83.3**)	**62.4** (**83.6**)	**69.8** (**82.0**)	**68.8** (**83.4**)	**84.6** (**89.2**)

reasonable development set seems impractical, so we settle for the cross-validation strategy and acknowledge that there may be an overfitting tendency.

3.3.4. Comparison of different constituent parse tree structures across major PPI corpora

Table 4 compares the performance of F1 (outside parentheses) and AUC (inside parentheses) between SDP-CPT and the previously used tree setups across major PPI corpora. For direct comparison purpose we re-implement two other effective tree setups for semantic relation extraction in the newswire domain, i.e., CS-SPT [14] and DSPT [15]. Additionally, the numbers of positive and negative instances in each corpus are reported in the 1st row and the performance scores of MCT (Minimum Complete Tree, the complete sub-tree rooted by the lowest common ancestor of the two proteins under consideration) are also listed in the 2nd row for reference. Approximate randomization tests are conducted between the F1 of the best one (in bold, i.e. SDP-CPT) and each of other tree structures, and the tree structures statistically different from SDP-CPT are underlined. The table shows

- Among all tree setups, SDP-CPT (in bold fonts) performs the best, and consistently and statistically significantly outperforms all other tree setups on three of the five PPI corpora except HPRD50 and LLL, whose sizes (the total number of positive and negative instances) are too small to pass significance tests.
- CS-SPT slightly outperforms SPT on most corpora while DSPT performs divergently on different corpora (for brevity, we omit their results of significance tests). The reason that DSPT performs excellently in the newswire domain [15] but not so much for PPI extraction may be that the heuristic rules they use to trim the constituent trees are more suitable for the newswire domain, thus limiting their capability of domain adaptation. This is the very issue we want to address in this article.

In summary, the above results suggest the superiority and generality of our SDP-CPT on various kinds of PPI corpora from the biomedical literature.

3.3.5. Comparison of kernel-based PPI extraction systems on the AIMed corpus

Table 5 compares our kernel-based system with other state-of-the-art PPI extraction systems on the AIMed corpus using the same 10-fold data splitting (we are not sure for [1,3,5] since they do not state this explicitly). The best scores of P/R/F1 for three categories (i.e., single kernels, feature-based ones, composite kernels) are shown in bold fonts respectively. It shows that our individual kernel-based system performs better than all the other single kernel-based systems on the AIMed corpus. Particularly, our SDP-CPT kernel outperforms the partial tree kernel over constituent parse trees [18]. It even outperforms the composite kernel combining bag-of-words (BOW) and constituent parse trees [21]. Although our individual kernel still performs worse than the composite kernels [43–45], the strength of our method lies in the potential

Table 5
Performance comparison of kernel-based PPI extraction systems on the AIMed corpus.

PPI extraction systems	P (%)	R (%)	F1
Single kernel			
Our SDP-CPT kernel	59.1	57.6	**58.1**
Dependency path [9]	61.4	53.3	56.6
Dependency graph [8]	52.9	**61.8**	56.4
Word subsequence [6]	**65.0**	46.4	54.2
Dependency tree [10]	56.9	39.0	46.3
Constituent parse tree [18]	39.2	31.9	34.6
Feature-based			
Mitsumori et al. [1]	54.2	42.6	47.7
Niu et al. [3]	43.2	**70.2**	53.5
Liu et al. [4]	**63.4**	48.8	54.7
Giuliano et al. [2]	60.9	57.2	59.0
Bui et al. [5]	55.3	68.5	**61.2**
Composite kernel			
BOW + dependency path [42]	**64.3**	44.1	52.0
BOW + constituent parse tree [21]	46.5	63.9	53.7
Dependency + predicate argument structure (PAS) [21]	54.9	65.5	59.5
BOW + dependency graph + PAS [44]	55.0	**68.8**	60.8
BOW + shortest path + dependency graph [43,45][a]	–	–	**64.2**

[a] Since the part of PPI extraction in [45] is an improved version of [43], their results are the same. This is also the case for Table 6.

Table 6
Performance comparison of kernel-based PPI extraction systems across multiple PPI corpora.

Corpus	Our SDP-CPT		Bun. [6]	Airola et al. [8]		Kim [9]	Tikk et al. [18]		Miwa et al. [43] Sætre et al. [45]		Miwa et al. [44]		Bui et al. [5]
	F1	AUC	F1	F1	AUC	F1	F1	AUC	F1	AUC	F1	AUC	F1
AIMed	58.1	83.3	54.2	56.4	84.8	56.6	43.8	77.6	**64.2**	**89.1**	60.8	86.8	61.2
BioInfer	62.4	83.6	–	61.3	81.9	57.6	47.6	73.3	67.6	**86.1**	**68.1**	85.9	60.0
HPRD50	68.8	83.7	–	63.4	79.7	67.8	69.7	**84.0**	69.7	82.8	70.9	82.2	**73.8**
IEPA	69.8	82.8	–	**75.1**	85.1	72.9	70.7	81.0	74.4	**85.6**	71.7	84.4	74.7
LLL	**84.6**	**89.9**	–	76.8	83.4	82.4	79.1	86.8	80.5	86.0	80.1	86.3	84.1

of combining SDP-CPT with other kernels to further improve the PPI extraction performance.

3.3.6. Comparison of kernel-based PPI extraction systems on five major PPI corpora

Finally we compare our kernel-based system with other systems using the same 10-fold data splitting strategy on major PPI corpora in Table 6. The best F1 and AUC scores for each corpus are highlighted. The table indicates that our method performs best on the LLL corpus among all systems. Compared with single kernel systems [6,8,9], our kernel-based system performs best in F1 score among four of five corpora. Although our kernel-based system performs worse than [5,43,45], Miwa et al. [43] and Sætre et al. [45] leverage multiple kernels and parsers while Bui et al. [5] adopt a two-stage approach, first using semantic and structured properties and then resorting to feature-based methods.

4. Conclusion

This paper presents a principled way to automatically generate the constituent parse tree for PPI extraction by making use of the shortest dependency path between two proteins. Although previous research indicates the difficulty of employing constituent parse trees for PPI extraction due to the relatively long distance between two proteins, our detailed analysis and evaluation indicate that the kernel-based method over the constituent parse tree structure can achieve promising results for PPI extraction from biomedical literature, provided that constituent parse trees are properly refined using the shortest dependency path to keep their critical parts while safely removing noisy information. This justifies the effectiveness of constituent parse trees for PPI extraction in the biomedical domain as well as for semantic relation extraction in the newswire domain.

In addition, we demonstrate that among different dependency encoding schemes for guiding the generation of constituent parse trees, the SD CCprocessed style can achieve the best performance. This is consistent with the argument that "collapsing" is often useful in simplifying patterns in relation extraction applications [24], though divergent with the finding that the SD basic variant is more suitable for the biomedical event extraction task [25,30]. We also find that the *Argument* dependency types (such as *subj* and *obj*) and the *Modifier* dependency types (such as *nn* and *prep*) are most useful for guiding the generation of constituent tree structures for PPI extraction. Significance tests, particularly approximate randomization tests, which have not yet been conducted in related works for PPI extraction, justify the reliability of our method, and show that our SDP-CPT tree setup statistically outperforms the previously used tree setups on three of the five PPI corpora except HPRD50 and LLL, whose sizes are actually too small to be used as reliable evaluation benchmark corpora.

Acknowledgments

Funding: China 863 Project 2012AA011102, China NSF Grants 60873150, 60970056 and 90920004, China Jiangsu NSF Grants BK2010219 and 11KJA520003.

References

[1] Mitsumori T, Murata M, Fukuda Y, Doi K, Doi H. Extracting protein–protein interaction information from biomedical text with SVM. IEICE Transactions on Information and Systems 2006;E89-D(8):2464–6.
[2] Giuliano C, Lavelli A, Romano L. Exploiting shallow linguistic information for relation extraction from biomedical literature. In: Proceedings of EACL'2006. p. 401–8.
[3] Niu Y, Otasek D, Jurisica I. Evaluation of linguistic features useful in extraction of interactions from PubMed; application to annotating known, high-throughput and predicted interactions in I^2D. Bioinformatics 2010;26(1):111–9.
[4] Liu B, Qian LH, Wang HL, Zhou GD. Dependency-driven feature-based learning for extracting protein–protein interactions from biomedical Text. In: Proceedings of COLING'2010 (Poster). p. 757–65.
[5] Bui Q-C, Katrenko S, Sloot PMA. A hybrid approach to extract protein–protein interactions. Bioinformatics 2011;27(2):259–65.
[6] Bunescu R, Mooney R. Subsequence kernels for relation extraction. In: Proceedings of NIPS'2005. p. 171–8.
[7] Erkan G, Özgür A, Radev DR. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In: Proceedings of EMNLP-CoNLL'2007. p. 228–37.
[8] 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(S1).
[9] Kim S, Yoon J, Yang J, Park S. Walk-weighted subsequence kernels for protein–protein interaction extraction. BMC Bioinformatics 2010;11:107.
[10] Chowdhury FM, Lavelli A, Moschitti A. A study on dependency tree kernels for automatic extraction of protein–protein interaction. In: Proceedings of BioNLP'2011. p. 124–33.
[11] Culotta A, Sorensen J. Dependency tree kernels for relation extraction. In: Proceedings of ACL'2004.
[12] Bunescu R, Mooney R. A shortest path dependency kernel for relation extraction. In: Proceedings of EMNLP'2005. p. 724–31.
[13] Zhang M, Zhang J, Su J, Zhou GD. A composite kernel to extract relations between entities with both flat and structured features. In: Proceedings of ACL-COLING'2006. p. 825–32.
[14] Zhou GD, Zhang M, Ji DH, Zhu QM. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In: Proceedings of EMNLP/CoNLL'2007. p. 728–36.
[15] Qian LH, Zhou GD, Zhu QM, Qian PD. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In: Proceedings of COLING'2008. p. 697–704.
[16] Moschitti A, Pighin D, Basili R. Tree kernels for semantic role labeling, special issue on semantic role labeling. Comput Linguist 2008;34(2):193–224.
[17] Zhang M, Che WX, Zhou GD, Aw AT, Tan CL, Liu T, et al. Semantic role labeling using a grammar-driven convolution tree kernel. IEEE Trans Audio, Speech Lang Process 2008;16(7):1315–29.
[18] Tikk D, Thomas P, Palaga P, Hakenberg J, Leser U. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. PLoS Computational Biology 2010;6(7).
[19] Zhou GD, Su J, Zhang J, Zhang M. Exploring various knowledge in relation extraction. In: Proceedings of ACL'2005. p. 427–34.
[20] Bunescu R, Ge R, Kate R, Marcotte E, Mooney R, Ramani A, et al. Comparative experiments on learning information extractors for proteins and their interactions. J Artif Intell Med 2005;33(2):139–55.
[21] Miyao Y, Sagae K, Sætre R, Matsuzaki T, Tsujii J. Evaluating contributions of natural language parsers to protein–protein interaction extraction. Bioinformatics 2009;25(3):394–400.
[22] Nivre J, Hall J, Kubler S, McDonald R, Nilsson J, Riedel S, et al. The CoNLL 2007 shared task on dependency parsing. In: Proceedings of EMNLP-CoNLL'2007. p. 915–32.
[23] Surdeanu M, Johansson R, Meyers A, Màrquez L, Nivre J. The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In: Proceedings of CoNLL'2008. p. 159–77.
[24] de Marneffe M-C, MacCartney B, Manning CD. Generating typed dependency parses from phrase structure parses. In: Proceedings of LREC'2006. p. 449–54.
[25] Buyko E, Hahn U. Evaluating the impact of alternative dependency graph encodings on solving event extraction tasks. In: Proceedings of EMNLP'2010. p. 982–92.
[26] Sagae K, Tsujii J. Dependency parsing and domain adaptation with LR models and parser ensembles. In: Proceedings of EMNLP-CONLL'2007. p. 1044–50.

[27] McDonald R, Pereira F, Ribarov K, Hajic J. Non-projective dependency parsing using spanning tree algorithms. In: Proceedings of HLT/EMNLP'2005. p. 523–30.
[28] Klein D, Manning CD. Accurate unlexicalized parsing. In: Proceedings of ACL'2003.
[29] Petrov S, Klein D. Improved inference for unlexicalized parsing. In: Proceedings of HLT-NAACL'2007. p. 449–54.
[30] Miwa M, Pyysalo S, Hara T, Tsujii T. Evaluating dependency representation for event extraction. In: Proceedings of COLING'2010. p. 779–87.
[31] Pyysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Jarvinen J, et al. BioInfer: a corpus for information extraction in the biomedical domain. BMC Bioinformatics 2007;8:50.
[32] Fundel K, Küffer R, Zimmer R. RelEx—relation extraction using dependency parse trees. Bioinformatics 2007;23(3):365–71.
[33] Ding J, Berleant D, Nettleton D, Wurtele E. Mining medline: abstracts, sentences, or phrases? In: Pacific Symposium on Biocomputing; 2002. p. 326–37.
[34] Nédellec C. Learning language in logic-genic interaction extraction challenge. In: Proceedings of the LLL'05 Workshop; 2005. p. 97–9.
[35] Pyysalo S, Airola A, Heimonen J, Björne J, Ginter F, Salakoski T. Comparative analysis of five protein–protein interaction corpora. BMC Informatics 2008;9(Suppl 3):S6.
[36] Joachims T. Text categorization with support vector machine: learning with many relevant features. In: Proceedings of ECML'1998. p. 137–42.
[37] Moschitti A. A study on convolution kernels for shallow semantic parsing. In: Proceedings of ACL'2004.
[38] Collins M, Duffy N. Convolution kernels for natural, language. NIPS'2001. p. 625–32.
[39] Nguyen TT, Moschitti A, Riccardi G. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In: Proceedings of EMNLP'2009; 2009. p. 1378–87.
[40] Chinchor N. The statistical significance of the MUC-4 results. In: Proceedings of the 4th conference on message understanding; 1992. p. 30–50.
[41] Petrov S, Barrett L, Thibaux R, Klein D. Learning accurate, compact, and interpretable tree annotation. In: Proceedings of COLING-ACL'2006. p. 433–40.
[42] Sætre R, Sagae K, Tsujii J. Syntactic features for protein–protein interaction extraction. In: Proceedings of LBM'07 vol. 319. p. 6.1–6.14.
[43] Miwa M, Sætre R, Miyao Y, Tsujii J. A rich feature vector for protein–protein interaction extraction from multiple corpora. In: Proceedings of EMNLP'2009. p. 121–30.
[44] Miwa M, Sætre R, Miyao Y, Tsujii J. Protein–protein interaction extraction by leveraging multiple kernels and parsers. Int J Med Inform 2009;78: e39–46.
[45] Sætre R, Yoshida K, Miwa M, Matsuzaki T, Kano Y, Tsujii J. Extracting protein interactions from text with the unified AkaneRE event extraction system. IEEE/ACM Trans Comput Biol Bioinformatics 2010;7(3):442–53.