Dependency-directed Tree Kernel-based Protein-Protein Interaction Extraction from Biomedical Literature

Longhua Qian **Guodong Zhou**
Natural Language Processing Lab
School of Computer Science and Technology, Soochow University
1 Shizi Street, Suzhou, China 215006
{qianlonghua,gdzhou}@suda.edu.cn

Abstract

Structured information plays a critical role in many NLP tasks, such as semantic relation extraction between named entities and semantic role labeling. This paper proposes a principled way to automatically generate constituent structure representation for tree kernel-based protein-protein interaction (PPI) extraction. The main idea behind our approach is that the critical portion in a constituent parse tree for PPI extraction can be automatically determined by the shortest dependency path between the two involved proteins, while other portion can be regarded as noise and ignored safely. Evaluation on multiple PPI corpora shows that our dependency-directed tree kernel-based method achieves promising results. This justifies the effectiveness of tree kernel-based methods for PPI extraction, in particular the advantage of dependency-directed constituent structure representation.

1 Introduction

Since determining protein interaction partners is crucial to understand both the functional role of individual proteins and the organization of the entire biological process, there is a significant interest in protein-protein interaction (PPI) extraction. However, manual collection of relevant 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, PPI extraction is much like the semantic relation extraction subtask (so called Relation Detection and Classification, RDC) defined by the ACE project (ACE, 2002-2007) 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 (Mitsumori et al., 2006; Giuliano et al., 2006; Sætre et al., 2007; Liu et al., 2010) and kernel-based methods (Bunescu et al., 2005a; Erkan et al., 2007; Airola et al., 2008; Kim et al., 2010).

Early studies on PPI extraction employ feature-based methods. However, the feature-based methods often fail to effectively capture the structured information, which is essential to identify the relationship between two proteins in a constituent or dependency-based syntactic representation.

With the wide adoption of kernel-based methods to many NLP tasks, particularly for semantic relation extraction and semantic role labeling, various kernels such as subsequence kernels (Bunescu et al., 2005a) and tree kernels (Li et al., 2008) have been applied to PPI extraction. On one hand, dependency-based kernels, such as edit distance kernels (Erkan et al. 2007), graph kernels (Airola et al., 2008) and subsequence kernels (Kim et al., 2010), show some promising results for PPI extraction. This suggests that dependency information plays a critical role in PPI extraction, much like semantic relation extraction in the newswire narratives (Culotta and Sorensen, 2004; Bunescu et al., 2005b). On the other hand, while tree kernels based on constituent parse trees achieve great success in semantic relation extraction (Zhang et al., 2006; Zhou et al., 2007a; Qian et al., 2008) and semantic role labeling (Moschitti, 2004; Zhang et al., 2008) from the newswire narratives, they haven't been fully explored for PPI extraction in the biomedical domain. Considering the similarity between the task of PPI extraction from the biomedical domain and that of relation extraction from the newswire domain, one question naturally arises: "How can kernel-based

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 10–19,
Chiang Mai, Thailand, November 8 – 13, 2011.

PPI extraction benefit from the constituent parse tree structure?"

To address this 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. This is done by taking advantage of the shortest dependency path between two involved proteins in the dependency parse tree 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 it as SDP-CPT (Shortest Dependency Path-directed Constituent Parse Tree). Experimental results on several major PPI corpora show the effectiveness of dependency-directed constituent structure representation and its preference over other state-of-the-art structure representations.

The rest of this paper is organized as follows. First, related work in PPI extraction is overviewed in Section 2. Then, Section 3 elaborates our shortest dependency path-directed constituent parse tree structure. Section 4 reports the experimental results on major PPI corpora. Finally we conclude our work in Section 5.

2 Related Work

Due to space limitation, this section only gives an overview on kernel-based methods on PPI extraction in the biomedical domain as well as semantic relation extraction in the newswire domain. For details about feature-based methods, please refer to related studies in the biomedical domain (Mitsumori et al., 2006; Giuliano et al., 2006; Liu et al., 2010) and those in the newswire domain (Zhao et al., 2005; Zhou et al. 2005, 2007b), respectively.

PPI extraction in biomedical domain

Representative kernel-based methods on PPI extraction take advantage of lexical or dependency information.

Bunescu et al. (2005a) adopt a generalized substring kernel over a mixture of words and word classes to extract protein interactions from biomedical corpora and semantic relations from newswire corpora. Particularly, they achieve the F1-score of 54.2 in extracting protein interactions from the AIMed corpus. Erkan et al. (2007) first 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. Sætre et al. (2007) use a tree kernel over dependency structures from two parsers and achieve the F1-score of 52.0 for PPI extraction from the AIMed corpus. Airola et al. (2008) introduce an all-dependency-paths graph kernel to capture complex dependency relationships between words and attain a significant performance boost at the expense of computational complexity. They achieve the F1-score of 56.4 in PPI extraction from the AIMed corpus. Kim et al. (2010) adopt a walk-weighted subsequence kernel based on shortest dependency paths to explore various substructures such as e-walks, partial match, and non-contiguous paths. They achieve the F1-score of 56.7 on the AIMed corpus.

Semantic relation extraction in newswire domain

In the literature, various kernels-based methods (Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu et al., 2005b; Zhang et al., 2006; Zhou et al., 2007a; Qian et al., 2008) have been widely used in semantic relation extraction in the newswire domain. In particular, Zhang et al. (2006), Zhou et al. (2007a) and Qian et al. (2008) adopt convolution tree kernels (Collins and Duffy, 2001) over constituent parse trees and show great success with comparable or even better performance than feature-based ones, motivated by the pioneer work of Moschitti et al. (2004; 2008) on semantic role labeling.

While convolution kernels (Haussler et al., 1999) can effectively capture structured information in discrete objects, the key problem for tree kernel-based methods lies largely in how to appropriately represent structured syntactic information inherent in relation instances. Zhang et al. (2006) discover that the Shortest Path-enclosed Tree (SPT) achieves the best performance among five tree setups. Zhou et al. (2007a) further extend it to Context-Sensitive Shortest Path-enclosed Tree (CS-SPT), which includes necessary predicate-linked path information. Qian et al. (2008) propose to automatically determine the appropriate part of a constituent parse tree by considering constituent dependencies on each node along the shortest path between two entity mentions and discarding irrelevant nodes. However, their adopted constituent dependency rules are manually constructed and thus difficult to adapt to other domains and languages.

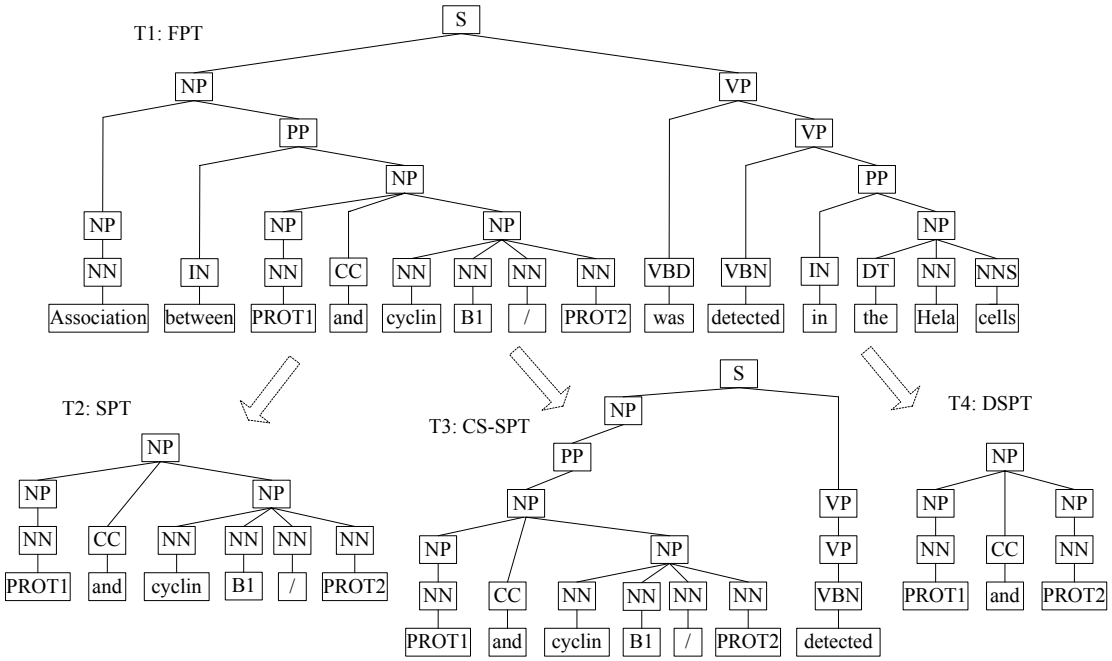

Figure 1. Different tree setups for a PPI instance between PROT1 and PROT2 from sentence "Association between PROT1 and cyclin B1 / PROT2 was detected in the HeLa cells." in the AIMed corpus

In this paper, we make use of the shortest dependency path in the dependency tree to refine the constituent parse tree. Specifically, all the words which appear on the shortest dependency path, together with their associated constituents, are kept in the constituent parse tree, while other constituents are removed, forming a Shortest Dependency Path-directed Constituent Parse Tree (SDP-CPT).

3 Constituent Structure Representation

This section first illustrates the limitations of commonly-used constituent parse tree setups, then emphasizes the importance of shortest dependency path in representing the constituent parse tree, and finally presents the shortest dependency path-directed constituent parse tree (SDP-CPT).

3.1 Limitations of Current 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. (2006) 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 (Zhou et al., 2005), most of PPI instances in the biomedical domain spans a relatively long distance, leading to more complexity and diversity (Bunescu et al., 2005c; Airola et al., 2008). 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. Miyao et al. (2008) 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 (called PTB in their paper) performs significantly worse than the other representations. Tikk et al. (2010) extensively compares different kernel-based methods on PPI extraction and show that the tree kernel over the constituent parse tree only achieves the F1-score of 34.6 on the AIMed corpus. Actually, our preliminary experiment on PPI extraction via the convolution tree kernel over SPT only achieves the F1-score of about 47 on the AIMed corpus. Such poor performance can be justified to a certain extent via a typical instance as illustrated in Figure 1, where the interaction between PROT1 and PROT2 (their actual names have been replaced) can be only determined by the overall constituent structure of the sentence. Obviously, SPT will fail to identify this interaction instance since SPT ignores the constituents outside the shortest path (Figure 1: T_2: SPT).

For the Context-Sensitive SPT (CS-SPT), as proposed in Zhou et al. (2007a), which extends necessary predicate-linked path information outside SPT, some critical information is still missing while there exists some noisy information. For the instance as shown in Figure 1 (T_3: CS-SPT), although the word "detected" and its asso-

ciated constituents are added, the more important portion of "association between" and their associated constituents are still missing while the noisy words "cyclin B1 / " still remaining.

In order to overcome the shortcomings in SPT and CS-SPT, Qian et al. (2008) 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 dependencies. 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 futile constituents, eventually leading to a more precise and concise parse tree structure. However, this tree structure still suffers from the following three shortcomings:

1) It disregards the constituents beyond the lowest common ancestor to the tree root, similar to CS-SPT as proposed in Zhou et al. (2007a). This may be largely due to the locality of semantic relations as defined in the ACE RDC corpus, which Zhou et al (2007a) and Qian et al. (2008) tackle.
2) The rules adopted to tackle constituent dependencies are manually constructed and thus may not be 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.
3) 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 for the Stanford Dependency representation).

In this paper, we attempt to address these problems by considering 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.

3.2 Shortest Dependency Path

Lexical dependencies can indicate both local and long-range relationships among words occurring in the same sentence. Such dependency relationships offer a condensed representation of the information necessary to assess the relationship between two proteins or entities. In order to capture the necessary information inherent in the depedency parse tree for extracting PPI instances, various kernels based on dependency paths, such as edit distance kernel (Erkan et al., 2007), all-dependency-path graph kernel (Airola et al., 2008), and walk-weighted subsequence kernels (Kim et al., 2010) have been proposed. Likewise for semantic relation extraction in the newswire domain, the kernels on dependency trees (Culotta and Sorensen, 2004) and the shortest dependency path (Bunescu et al., 2005b) have been proposed. One common characteristic to these kernels is that they all contain the shortest dependency path and usually assign more weights to them than to other ones, similar to the graph kernel proposed by Airola et al. (2008). This indicates the importance of the shortest dependency path over other paths in the dependency path tree or the dependency graph.

Currently, there are two established dependency representations available, viz. CoNLL scheme (adopted by CoNLL'2007 and CoNLL'2008 Shared tasks) (Nivre et al., 2007; Surdeanu et al., 2008) and Standford scheme (adopted by Stanford parser) (de Marneffe et al., 2006). 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 scheme is closer to the targeted semantic representation from the perspective of relation extraction (Buyko and Hahn, 2010). Particularly, among the four styles of Stanford representations, "collapsed dependency" can much simplify patterns in relation extraction since dependencies involving preposition, conjunct as well as referent of relative clause are effectively collapsed to reflect direct dependencies between content words. Therefore, the collapsed variant of Stanford scheme is adopted in this paper to refine the constituent parse tree as described in the next subsection.

3.3 SDP-CPT: Shortest Dependency Path-directed Constituent Parse Tree

Considering the importance of dependency path in PPI extraction and the effectiveness of employing dependency 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, we can reshape the constituent parse tree by making use of the shortest dependency path between two proteins. Figure 2 describes the procedure to

generate the Shortest Dependency Path-directed Constituent Parse Tree (SDP-CPT).

Note that Step 3(a) in Figure 2 is necessary since a dependency tuple of the type "prep_*xx*(*governor*, *dependent*)" implies a relationship between the preposition *xx* and the *dependent*, which is important to PPI. For Step 3(b), when a word on which the two proteins are directly or indirectly dependent is discovered, it is natural to add this path for maintaining the integrality of SDP-CPT.

Input: a sentence and two proteins in it
Output: an SDP-CPT
Steps:

1) Given the input sentence, generate the constituent parse tree using a constituent parser, and various dependency tuples using a dependency parser.
2) Given the two proteins, extract the shortest constituent path (SCP, i.e. the shortest path-enclosed tree) from the constituent parse tree and construct the shortest dependency path (SDP) from the dependency tuples.
3) For each word along the SDP, add the corresponding leaf word node and its upper constituents to the SCP. Particularly,
   a) when the dependency type is "prep_*xx*", such as "prep_of", the preposition *xx* and its associated constituent are also added;
   b) when the word to be added is outside the SCP, a new path from the current lowest common ancestor to one of the added words' ancestors is also added.
4) Merge any two consecutive NP/VP nodes along the paths into a single one.

Figure 2. Procedure for generating SDP-CPT

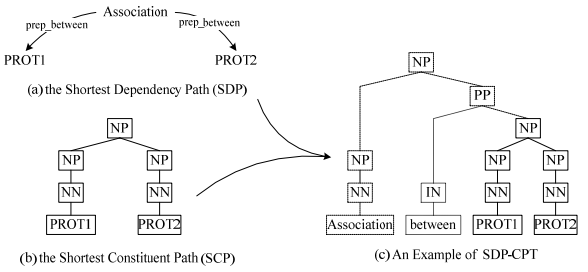

Figure 3. Generation of an example of SDP-CPT

In order to demonstrate the process of generating a SDP-CPT, we take the sentence and the two proteins shown in Figure 1 as an example. Figure 3 illustrates the detailed generation process. First, the shortest dependency path (SDP) and the shortest constituent path (SCP) are generated as depicted in Figure 3(a) and 3(b) respectively. Then, every word in the SDP is added into the SCP together with its associated constituents. In this case, since the two protein names in the SDP share a common ancestor "Association", the word "Association" together with its constituent ancestors are added into the SCP and a new path "NP→PP→NP" is created as rendered by the dashed lines. Finally, since the dependency type between "PROT1" and "Association" is *prep_between*, the preposition word "between" and its constituent ancestors are added into the SCP as rendered by the dotted lines. Since no further post-processing is necessary in this example, SDP-CPT is eventually formed. Compared to other tree setups in Figure 1, namely SPT, CS-SPT and DSPT, obviously SDP-CPT is much more concise and precise for this PPI instance.

4 Experimentation

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.

4.1 Data Sets and Preprocessing

In order to fairly compare our work with other PPI extraction systems, we use five PPI corpora, i.e., AIMed (Bunescu et al., 2005a), BioInfer (Pyysalo et al., 2007), HPRD50 (Fundel et al., 2007), IEPA (Ding et al., 2002) and LLL (Nédellec, 2005). Particularly, 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. Totally, there are 4,084 protein references and around 1,000 annotated protein-protein interactions in this data set.

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, 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. Besides, for a potential PPI instance, the two involved proteins are replaced by PROT1 and PROT2 respectively 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[1] to generate both the con-

[1] http://nlp.stanford.edu/software/lex-parser.shtml

stituent parse trees and their corresponding dependency tuples.

4.2 Classifier and Evaluation Metrics

In our experimentation, we select Support Vector Machines (SVM) as the classifier since SVM represents the state-of-the-art in the machine learning research community. In particular, we use the SVM^{light} (Joachims, 1998) with the convolution tree kernel function SVM^{light}–TK (Moschitti, 2004)[2] to compute the similarity between two constituent parse trees.

Evaluation is done using 10-fold document-level cross-validation, each of which contains 90% of documents as the training data and 10% as the test data. Particularly, for the AIMed corpus we apply the exactly same 10-fold split as widely used in a series of relevant studies (e.g., Bunescu et al., 2005a; Giuliano et al., 2006). 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 OAOD (One Answer per Occurrence in the Document) 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 F1-score (F1). As an alternative to F1-score, the AUC (*area under the receiver operating characteristics curve*) score is proved to be invariant to the class distribution of the test dataset. Therefore, we also provide the AUC score of our system for referrence as by Airola et al. (2008).

4.3 Experimental Results

Comparison of different lengths of dependency paths on the AIMed corpus

Table 1 reports the performance of PPI extraction on the AIMed corpus corresponding to different lengths of the dependency paths using all kinds of dependency types. Here, two partial dependency paths on SDP, starting from each of the two proteins respectively, are utilized to generate the tree representation. The length of these two paths is shown in the 1st column. The words corresponding to the nodes on these two paths together with these words' associated constituents in the parse tree are added to SCP. For example, the length of 0 (L0) means that not a single word or constituent will be added to SCP, while the length of 1 (L1) means that the words corresponding to the parents of two proteins on SDP and these words' associated constituents in the parse tree are added to SCP. Meanwhile, the performance of the SPT setup is also listed as the baseline for comparison.

Length	P (%)	R (%)	F1	AUC
SPT	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	59.6	54.3	56.7	82.7

Table 1. Performance comparison of PPI extraction on the AIMed corpus with different lengths of dependency paths using all kinds of dependency types

This table shows that the constituent parse tree directed by the shortest dependency path (SDP-CPT) achieves the best performance of 59.6/54.3/56.7/82.7 in P/R/F1/AUC, significantly outperforming SPT by 9.6 units in F1 and 2.8 units in AUC largely due to the substantial increase in recall. This indicates that SDP-CPT can remove much noise in SPT while adding some useful information. 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 SDP-CPT can achieve the best performance. Therefore, all the subsequent experiments adopt the SDP-CPT setup unless specified.

Contribution of different kinds of dependencies on the AIMed corpus

Table 2 compares the contribution of various kinds of dependencies in 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

[2] http://ai-nlp.info.uniroma2.it/moschitti/

any exists, are further ordered by their potential importance. The percentage of occurring frequency with which each minor type is employed when generating the SDP-CPT with respect to the total number of dependency tuples is listed in Column 2. Particularly, the tree setup without using any dependency type, which corresponds to that with the length of 0 (SCP) in Table 1, is displayed at the top row. Furthermore, the dependency types are added in two different ways:

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

Typed Dependency	%	P(%)	R(%)	F1	AUC
SCP+L0	-	45.0	19.5	26.5	67.9
Argument					
+subj	10	52.5 (52.6)	33.2 (33.2)	40.4 (40.4)	72.7 (72.7)
+obj	31	56.2 (53.8)	46.2 (42.4)	50.4 (47.0)	76.6 (76.6)
+arg-others	2	56.1 (48.8)	47.0 (14.6)	50.9 (21.3)	76.5 (68.6)
Modifier					
+nn	10	58.1 (54.9)	53.4 (38.5)	55.1 (44.6)	81.4 (77.5)
+prep	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 2. 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)

Table 2 shows that with the addition 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 added (59.6/54.3/56.7/82.7 in P/R/F1/AUC). Particularly, it shows

- The dependency types of *subj*, *obj, prep* and *nn* yield 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, though *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 improve the performance in the accumulative mode.

Since the governors of *subj* and *obj* types are verbs, those of the *prep* type are nouns and 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", on which two proteins are directly or indirectly dependent, are particularly important for PPI extraction. Henceforth, in the following experiments all the *Argument* and *Modifier* types are included while the *Conjunction* and *Others* types are excluded.

Tree setups	AIMed	BioInfer	HPRD 50	IEPA	LLL
Ratio of POS/NEG	1000/ 4834	2534/ 7119	163/ 270	335/ 482	164/ 166
MCT	31.8 (78.0)	53.8 (76.7)	48.0 (73.4)	62.3 (78.6)	77.1 (73.4)
SPT (baseline)	47.1 (79.9)	54.2 (73.7)	61.3 (81.6)	66.6 (82.2)	79.4 (86.1)
CS-SPT	46.5 (80.2)	54.5 (74.5)	63.6 (79.9)	66.8 (81.0)	80.1 (86.0)
DSPT	50.0 (77.8)	58.3 (78.5)	66.0 (80.3)	68.6 (80.9)	77.3 (79.3)
SDP-CPT	**58.1** (83.3)	**62.4** (83.6)	**68.8** (83.4)	**69.8** (82.0)	**84.6** (89.2)

Table 3. Comparison of F1-score(outside parentheses) and AUC(inside parentheses) between SDP-CPT and different tree setups across major PPI corpora

Comparison of different constituent parse tree structures across major PPI corpora

Table 3 compares the performance of F1-score (outside parenthesis) and AUC (inside parentheses) between SDP-CPT and the previously-used tree setups across major PPI corpora. Particularly, SPT is used as a baseline and for comparison we re-implement two other effective tree setups for semantic relation extraction in the newswire domain, i.e. CS-SPT (Zhou et al., 2007a) and DSPT (Qian et al., 2008). Significance tests are conducted between each of them and the baseline.

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. The table shows

- Among all tree setups SDP-CPT performs best and significantly outperforms SPT consistently on most PPI corpora.
- CS-SPT slightly outperforms SPT on most corpora while DSPT performs divergently on different corpora. The reason that DSPT performs excellently in the newswire domain (Qian et al., 2008) but not so much for PPI extraction may be that the heuristic rules they use to prune the constituent trees are more suitable for the newswire domain, thus limiting their capability of domain adaptation.

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

PPI extraction systems	P(%)	R(%)	F1
Our SDP-CPT kernel	59.1	57.6	**58.1**
Dependency path: Kim et al. (2010)	61.4	53.3	56.7
Dependency graph: Airola et al. (2008)	52.9	**61.8**	56.4
Word subsequence: Bunescu et al. (2005a)	**65.0**	46.4	54.2
Constituent parse tree: Tikk et al. (2010)	39.2	31.9	34.6
BOW+Dependency path: Sætre et al. (2007)	**64.3**	44.1	52.0
BOW+Constituent parse tree: Miyao et al. (2008)	50.9	56.1	53.0
Global+Local context: Giuliano et al. (2006)	60.9	57.2	59.0
Dependency+Predicate Argument Structure: Miyao et al. (2008)	54.9	**65.5**	59.5
BOW+Shortest Path+Depencency graph: Miwa et al. (2009)	-	-	**64.2**

Table 4. Performance comparison of kernel-based PPI extraction systems on the AIMed corpus

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

Table 4 compares our kernel-based system with other state-of-the-art kernel-based ones on the AIMed corpus using the exactly same 10-fold data splitting. It shows that our individual kernel-based system performs better than all the other individual kernel-based systems on the AIMed corpus. Particularly, our SDP-CPT kernel significantly outperforms the Partial Tree kernel over constituent parse trees (Tikk et al., 2010). It even significantly outperforms the composite kernel combining BOW and constituent parse trees (Miyao et al., 2008). Although our individual kernel performs worse than the composite kernels as adopted by Miyao et al. (2008) and Miwa et al. (2009), the strength of our kernel-based system lies in the simplicity of our shortest dependency path-directed constituent parse tree.

5 Conclusion and Future Work

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 tree information for PPI extraction due to the relatively long distance between two proteins, our detailed analysis and evaluation indicate that the constituent parse tree can achieve promising results for PPI extraction. Moreover, our dependency-directed constituent parse tree structure provides a general way to automatically determine the constituent parse tree for a wide class of related learning tasks, such as semantic relation extraction, semantic role labeling and even coreference resolution.

For future work, we would like to apply our approach to other NLP tasks. Meanwhile, we will investigate the effect of constituent parse information on dependency-based relational learning in better exploring the synergy between dependency and constituent-based syntactic information.

Acknowledgement

This research is supported by Projects 60873150, 60970056, and 90920004 under the National Natural Science Foundation of China, and the Project BK2010219 under the Provincial Natural Science Foundation of Jiangsu, China.

References

ACE. 2002-2007. The Automatic Content Extraction (ACE) Projects. 2007. http//www.ldc.upenn.edu/ Projects/ACE/.

Airola A., Pyysalo S., Björne J., Pahikkala T., Ginter F. and Salakoski T. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross corpus learning. *BMC Bioinformatics*.

Bunescu R. and Mooney R. 2005a. Subsequence kernels for relation extraction. In *Proceedings of NIPS'05*: 171–178. December 2005.

Bunescu R. and Mooney R.. 2005b. A shortest path dependency kernel for relation extraction. In *Proceedings of EMNIP'05*: 724–731. Vancover, B.C.

Bunescu R., Ge R., Kate R., Marcotte E., Mooney R., Ramani A. and Wong Y. 2005c. Comparative Experiments on learning information extractors for Proteins and their interactions. *Journal of Artificial Intelligence in Medicine*, 33(2): 139–155.

Buyko E. and Hahn U. 2010. Evaluating the impact of alternative dependency graph encodings on solving event extraction tasks. In *Proceedings of EMNLP'2010*: 982-992.

Collins M. and Duffy N. 2001. Convolution Kernels for Natural Language. *NIPS 2001*: 625-632.

Culotta A. and Sorensen J. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL'04*. July 2004. Barcelona, Spain.

de Marneffe M.-C., MacCartney B. and Manning C. D. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Ding J., Berleant D., Nettleton D. and Wurtele E. 2002. Mining medline: abstracts, sentences, or phrases? *Pacific Symposium on Biocomputing*, pages 326–337.

Erkan G., Ozgur A. and Radev D.R. 2007. Semi-Supervised Classification for Extracting Protein Interaction Sentences using Dependency Parsing, In *Proceedings of EMNLP-CoNLL'07*: 228–237. June, 2007, Prague, Czech.

Fundel K., Kuffner R. and Zimmer R. 2007. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Giuliano C., Lavelli A. and Romano L. 2006. Exploiting Shallow Linguistic Information for Relation Extraction from Biomedical Literature. In *Proceedings of EACL'06*: 401–408. April, 2006, Trento, Italy.

Haussler D. 1999. Convolution Kernels on Discrete Structures. *Technical Report UCS-CRL-99-10*, University of California, Santa Cruz.

Joachims T. 1998. Text Categorization with Support Vector Machine: learning with many relevant features. *In Proceedings of European Conference on Machine Learning(ECML'1998)*: 137-142.

Kim S., Yoon J., Yang J. and Park S. 2010. Walk-weighted subsequence kernels for protein-protein interaction extraction. *Journal of BMC Bioinformatics*, 2010, 11:107.

Li J., Zhang Z., Li X. and Chen H. 2008. Kernel-Based Learning for Biomedical Relation extraction. *Journal of the American Society for Information Science and Technology*, 59(5):756–769.

Liu B., Qian L.H., Wang H.L. and Zhou G.D. 2010. Dependency-Driven Feature-based Learning for Extracting Protein-Protein Interactions from Biomedical Text. In *Proceedings of COLING'2010*: 757-765 (Poster).

Mitsumori T., Murata M., Fukuda Y., Doi K. and Doi H. 2006. Extracting protein-protein interaction information from biomedical text with SVM. *IEICE Transactions on Information and Systems*, E89-D (8): 2464–2466.

Miwa M., Sætre R., Miyao Y. and Tsujii J. 2009. A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora. In *Proceedings of EMNLP'09*: 121–130. August 2-7, 2009, Singapore.

Miyao Y., Sætre R., Sagae K., Matsuzaki T. and Tsujii J. 2008. Task-oriented evaluation of syntactic parsers and their representations. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08: HLT)*: 46-54.

Moschitti A. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In *Proceedings of ACL'2004*. July 21-26, 2004. Barcelona, Spain.

Moschitti A., Pighin D. and Basili R. 2008. Tree Kernels for Semantic Role Labeling, Special Issue on Semantic Role Labeling, *Computational Linguistics*. 34(2): 194-224.

Nédellec C. 2005. Learning language in logic-genic interaction extraction challenge. In *Proceedings of the LLL'05 Workshop*.

Nivre J., Hall J., Kubler S., McDonald R., Nilsson J., Riedel S. and Yuret D. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. *EMNLP-CoNLL 2007*: 915–932.

Pyysalo S., Ginter F., Heimonen J., Björne J., Boberg J., Jarvinen J. and Salakoski T. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Qian L.H., Zhou G.D., Zhu Q.M. and Qian P.D. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. *COLING-2008*: 697-704. Manchester, UK.

Surdeanu M., Johansson R., Meyers A., Màrquez L., and Nivre J. 2008. The CoNLL-2008 Shared Task

on joint parsing of syntactic and semantic dependencies. *CoNLL'2008*.

Sætre R., Sagae K. and Tsujii J. 2007. Syntactic features for protein-protein interaction extraction. In *Proceedings of LBM'07*, (319): 6.1–6.14.

Tikk D., Thomas P., Palaga P., Hakenberg J. and Leser U. 2010. A Comprehensive Benchmark of Kernel Methods to Extract Protein–Protein Interactions from Literature. *PLoS Computational Biology*, 6(7).

Zelenko D., Aone C. and Richardella A.. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, (2): 1083-1106.

Zhang M., Zhang J., Su J. and Zhou G.D. 2006. A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. *ACL/COLING-2006*: 825-832. Sydney, Australia.

Zhang M., Che W.X., Zhou G.D., Aw A.T., Tan C.L., Liu T. and Li S. 2008. Semantic Role Labeling using a Grammar-driven Convolution Tree Kernel. *IEEE Transaction on Audio, Speech and Language Processing*. 2008, 16(7): 1315-1329.

Zhao S.B. and Grishman R. 2005. Extracting relations with integrated information using kernel methods. *ACL-2005*: 419-426.

Zhou G.D., Su J., Zhang J. and Zhang M. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL'2005*: 427-434. 25-30 June 2005, Ann Arbor, Michigan, USA

Zhou G.D., Zhang M., Ji D.H. and Zhu Q.M. 2007a. Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. *EMNLP/CoNLL-2007*: 728-736.

Zhou G.D. and Zhang M. 2007b. Extracting relation information from text documents by exploring various types of knowledge. *Information Processing and Management*. 43(2007): 969-982.