Relation extraction methods for biomedical literature

Bui, Q.C.

Publication date
2012

Link to publication

Citation for published version (APA):
Bui, Q. C. (2012). *Relation extraction methods for biomedical literature*.

Chapter 4. A hybrid approach to extract protein-protein interactions

Abstract

Protein-protein interactions (PPIs) play an important role in understanding biological processes. Although recent research in text mining has achieved a significant progress in automatic PPI extraction from literature, performance of existing systems still needs to be improved.

In this chapter, we present a novel algorithm for extracting PPIs from literature which consists of two phases. First, we automatically categorize the data into subsets based on its syntactic properties and extract candidate PPI pairs from these subsets. Second, we apply support vector machines (SVM) to classify candidate PPI pairs using features specific for each subset. We obtain promising results on five benchmark datasets: AIMed, BioInfer, HPRD50, IEPA, and LLL with F-scores ranging from 60% to 84%, which are comparable to the state-of-the-art PPI extraction systems. Furthermore, our system achieves the best performance on cross-corpora evaluation and comparative performance in terms of speed.

[*] This chapter is based on Bui et al., "A hybrid approach to extract protein-protein interactions". *Bioinformatics*. **27**, 259-265 (2011).

4.1 Introduction

Extraction of protein-protein interactions (PPIs) from literature is an important research topic in the field of biomedical natural language processing (NLP) [50] Numerous PPIs have been manually curated and stored into databases, such as BIND, HDPR, and IntAct. However, this task has been proven time and resource-consuming. As a consequence, most data on PPIs can only be found in literature [24].

Several approaches for extracting PPIs from biomedical text have been reported. These methods range from co-occurrence to more sophisticated machine learning (ML) systems augmented by NLP techniques. Co-occurrence is the simplest approach, which results in high recall but low precision. Rule or pattern based approaches can increase precision but significantly lower recall. In addition, these rule sets are derived from training data and are therefore not always applicable to other data they are not developed for [25, 34].

Recently, many machine learning based methods have employed NLP techniques such as shallow parsing or full parsing [43, 87, 118, 119]. Since full parsing produces more elaborate syntactic information than shallow parsing, PPI extraction systems based on full parsing can potentially yield better results [26]. The output of the parser can be represented either as constituent trees or dependency trees. In this case, the PPI extraction task is treated as a binary classification problem which requires a formal protein pair representation and a suitable machine learning method. A protein pair (an instance) can be represented by a set of features, which are derived from the sentence or its syntactic structure. A machine learning method is then used to distinguish between positive and negative instances [45, 120].

Many linguistic features and machine learning methods have been proposed for the PPI extraction task. Based on feature types, these approaches can be divided into three groups. The first group focuses on lexical and word context features. Bunescu and Mooney [53] designed a subsequence kernel which uses the following information in a sentence: before the first protein, between two proteins, and after the second protein, and combined these features to obtain patterns. Giuliano et al. [87] extended this approach by using a bag-of-words and adding a local context kernel. The second group exploits syntactic features of a sentence. Sætre et al. [90] used various syntactic path features and context features related to words before, between, and after two interacting entities. Katrenko and Adriaans [88] proposed a method based on information found in the pre-defined levels of the dependency trees, such as local dependency contexts of the protein names and tree's roots. Kim et al. [51] enhanced previous work by proposing a walk kernel which explores the shortest dependency path between two proteins and a modified dependency tree with the parts-of-speech (POS) features. As an alternative to previous approaches, Airola et al. [58] introduced an all-paths graph kernel. They represented a sentence with a dependency graph and considered dependencies connecting two entities outside the shortest path as well as on the shortest path.

Along with the proposed methods, Fayruzov et al. [100], Niu et al. [95], and Van Landeghem et al. [96] also studied individual impact of a variety of feature types on the PPI extraction task. In addition, a study of Miyao et al. [26] has shown that the accuracy of syntactic parsers also contributes to the overall performance of the PPI systems. To compensate for the limitations of each individual feature set and parser errors, Miwa et al. [50, 63] proposed a method that combines all the lexical and parsing features using multiple kernels and parsers. Their system achieved the state-of-the-art performance on a number of benchmark data sets. However, the studies by Fayruzov et al. [100] and Kim et al. [98] also demonstrated that if two feature types have overlapping rather than complimentary effects, dropping one of them can result in a computationally more efficient method and potentially make a mining algorithm more robust. This argument was also confirmed in the work of Miwa et al. [63] who showed that excluding the bag-of-words feature leads to better performance on the AIMed corpus.

Although many approaches have been proposed in the past, the problem of finding the most suitable features for extracting PPIs remains. Adding more features might sometimes improve performance, but they can introduce noise in other cases, or require more computational resources [98]. In this chapter, we propose a novel method that consists of two phases. First, we apply semantic rules to partition the data set into subsets according to its semantic properties and extract candidate PPI pairs from these subsets. Second, we introduce enhanced feature sets for use with a machine learning classifier to classify these extracted PPI pairs.

To the best of our knowledge, this is the first method that categorizes data into subsets and selects the most appropriate features for each subset. As a result, we increase the robustness of the learning method and make it computationally effective. Our study only focuses on the PPI extraction task with an assumption that relevant named entities were manually annotated.

4.2 System and methods

The workflow of the proposed system is as follows:

1. Text preprocessing
2. Extracting candidate PPI pairs
3. Classifying extracted PPI pairs

4.2.1 Text preprocessing

Text preprocessing includes converting protein names using a predefined rule set, filtering out input sentences with only one protein, splitting input sentences contain multiple clauses, and parsing sentences using the Stanford lexical parser[1].

Processing protein names

In order to improve accuracy of the parser, we replace all mentioned protein names with a place holder, i.e. PRO1, PRO2 (we refer to them as PRO*). We define a rule set to resolve the problem with embedded protein names (e.g., AIMed corpus), protein names that share prefix or suffix (e.g., AIMed and BioInfer corpora), and protein names including multiple positions. After this process, the number of proteins in the sentence is not changed. The list of original protein names for each sentence is maintained for reference purposes.

Replacing parenthetical remarks and splitting a sentence

If no word inside parentheses is a protein name, then all words and the parentheses are removed. In case the sentence consists of multiple clauses, the system splits it into clauses. The resulting sentence is referred to as a *simplified sentence*.

In the last step of text preprocessing, a simplified sentence that contains at least two protein names is analyzed with the Stanford lexical parser to produce a syntactic tree. All parse trees are stored in a local database for later use. An example of a sentence and its output after text preprocessing step is given below:

AIMed.d101.s852: The bZIP domains of Fos and Jun mediate a physical association with the TATA box-binding protein.

Its simplified sentence: The bZIP domains of PRO0 and PRO1 mediate a physical association with the PRO2.

4.2.2 Extracting candidate PPI pairs

Previous studies [48, 55] and our study in chapter 3 have shown that, in biomedical text, relation between two entities (protein-protein, protein-gene, drug-mutation, and others) can be expressed in the following abstract forms:

Form 1: PRO_i *word** REL (verb) *word** PRO_j

Example: *PRO1 interacts with PRO2.*

Form 2: PRO_i *word** REL (noun) *word** PRO_j

Example: PRO1 *has a weak interaction with PRO2.*

Form 3: REL (noun) *word** PRO_i *word** PRO_j

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

Example: *interaction between PRO1 and PRO2*.

Here, REL is a cue word (interaction, inhibit, etc.) and can be a noun or a verb. *word** are tokens between PRO* and REL. PRO_i and PRO_j can be any protein pair and $j>=i+1$ (e.g., <PRO1, PRO2>). In addition, a closer look at the annotated corpora (e.g., AIMed and BioInfer) reveals that we can define two more forms:

> **Form 4** (*compound form*): PRO_i/PRO_{i+1} or PRO_i PRO_{i+1} or PRO_i-PRO_{i+1}, if two entities appear in the sentence with the patterns above, they seem to have interaction.
>
> Example: *PRO1/PRO2 binding; PRO1-PRO2 compound*.
>
> **Form 5** (*complex form*): PRO_i *word** PRO_j *word** REL
>
> Example: *PRO1, PRO2 interact*; *in PRO1, PRO2 complex*.

Form 4 is expressed as a pattern and requires an exact match. For other forms, there might be one or more tokens between <PRO_i, REL>, <REL, PRO_j> or <PRO_i, PRO_j>. Based on these basic forms, we now map the semantic relations of these forms into parse trees. For convenience, we define the following patterns:

a) *Full clause*: S < ((NP << PRO*) $++ (VP << PRO*))

b) *Partial clause*: VP < ((NP << PRO*) $++ (S < (VP << PRO*)))

c) *Sub-clause*: S < ((NP << PRO*) $++ (VP < (S|SBAR << PRO*)))

d) *NP pattern*: NP < ((NP <<# REL) $++ (PP << PRO*))

Here S (clause), NP (noun phrase), VP (verb phrase), PP (prepositional phrase), and SBAR (sub-clause) are constituents of the parse tree, PRO* is a place holder for a protein name, and REL is a cue word of the input sentence. All patterns above are written using the Tregex syntax[2], which is developed within the Stanford parser package. Figure 4.1 illustrates some parse trees with the patterns mentioned above, e.g., the parse tree in Figure 4.1a shows a *full clause*, Figure 4.1b shows a *NP pattern* and Figure 4.1c shows a *sub-clause*.

Relation list

We created a relation list by combining the relation lists used in the previous work by Chowdhara *et al.* [111], Fundel *et al.* [55], and Niu *et al.*[95].

In the following section, we describe the procedures and algorithm to extract candidate PPI pairs from a parse tree or a simplified sentence:

[2] http://nlp.stanford.edu/software/tregex.shtml

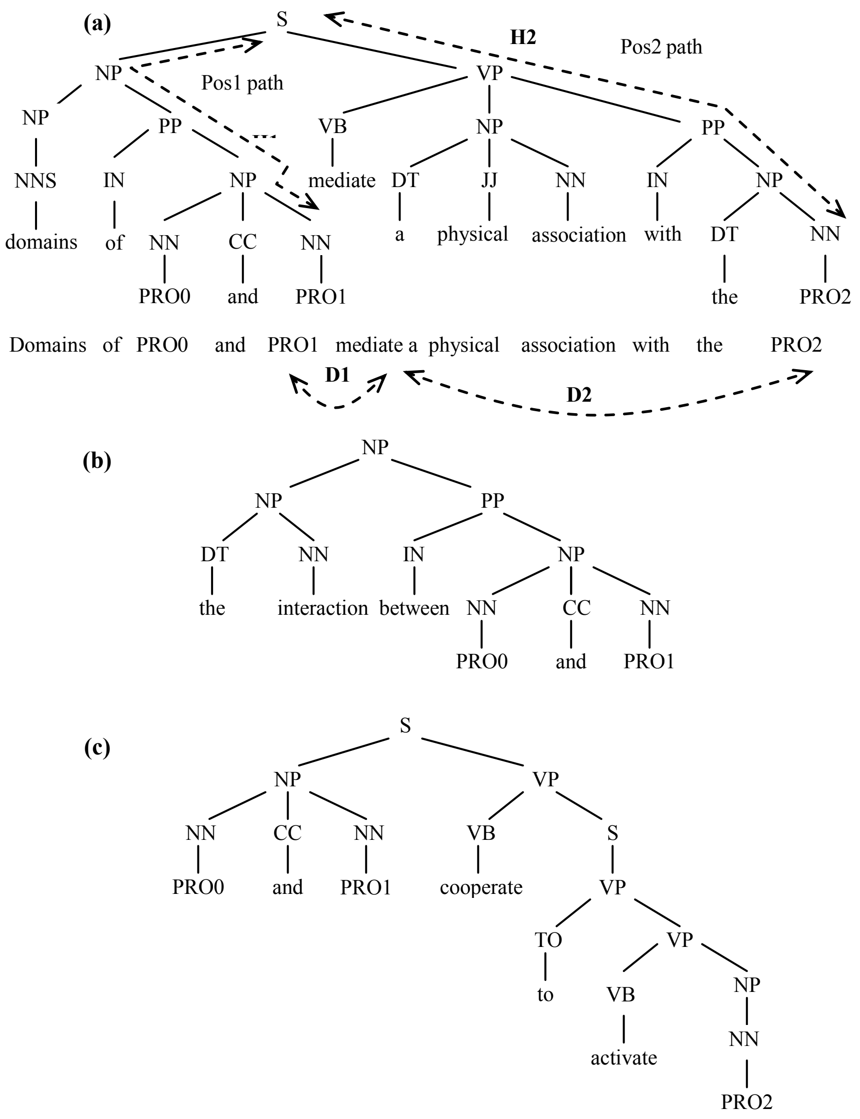
(a)
H2
Pos2 path
Pos1 path
Domains of PRO0 and PRO1 mediate a physical association with the PRO2
D1
D2
(b)
(c)

Figure 4.1. Examples of parse trees output from the Stanford lexical parser. Figure 4.1(a) shows some features (D1, D2, H1, H2) and paths from the join node connecting a given protein pair (PRO1, PRO2).

Procedure for Form 1: This procedure requires a parse tree which contains *full clause*, *partial clause*, or *sub-clause* patterns. The procedure is as follows:

a. Check a head word of VP that corresponds to satisfied pattern. If this verb belongs to the relation list, use it as REL.

b. Create a *right_keys* list: use all keys in VP as *right_keys*.

c. Create a *left_keys* list: use all keys in NP if a satisfied pattern is *full clause* or *sub-clause*. If a pattern is *partial clause,* then find a sister NP immediately preceding VP.

d. Form candidate PPI pairs: enumerate the *left_keys* and *right_keys* to compose a triple <PRO1-REL-PRO2>.

Procedure for Form 2: Form candidate PPI pairs from a simplified sentence if they satisfy the following form: PRO* – REL – PRO*.

Procedure for Form 3: This procedure requires a parse tree which contains *NP pattern*. The procedure is as follows:

a. Check a head word of NP that corresponds to satisfied pattern. If this noun belongs to the relation list, use it as REL.

b. Find a proposition as a splitter in pattern's PP phrase in order to create *left_keys* and *right_keys* lists.

c. Form candidate PPI pairs: enumerate the *left_keys* and *right_keys* to compose a triple <PRO1-REL-PRO2>.

Procedure for Form 4: Form candidate PPI pairs from an input sentence if they satisfy any of the following forms: PRO*/PRO*; PRO* PRO*; PRO*-PRO*.

Procedure for Form 5: Form candidate PPI pairs from the simplified sentence if they satisfy the following form: PRO*- PRO* - REL*,* and if the distance from the first PRO* in the pattern to the REL is shorter than 5 tokens.

For these procedures, the extracted candidate PPI pairs obtained from the same procedure are grouped together i.e. output from Form1 are grouped into group 1. Among these groups, group 2 is a special case of group 1, with the purpose to recover PPI pairs that the group 1 failed to extract due to the parser errors [26]. As a consequence, in some cases, a PPI pair may belong to more than one group. Therefore, the order in which patterns are applied is important (as described in the Algorithm 4.1 below).

Algorithm 4.1. // Algorithm to extract candidate PPI pairs.

Input: simplified sentence, parse tree, and the relation list

```
Output: list of candidate PPI pairs of corresponding groups
Init: used_list = null // store extracted pairs to avoid overlap since one pair can
satisfy more than one pattern.
For a list of sub-sentence/parse tree
    For form from 1 to 5
        Extract candidate pairs from parse tree or simplified sentence
        If candidate pairs found
            For each candidate pair
             Check whether this pair in the used_list
              If not found
                    Store this pair to the corresponding group
                    Store this pair to the used_list
                End if
            End for
        End if
    End for
End for
```

Features

In this section we propose feature sets for machine learning classification. Our feature sets are the modification and combination some of the features that were previously proposed by Chowdhary *et al.* [111], Giuliano *et al.* [87], Miwa *et al.* [45, 50], and Niu *et al.* [95]. For each candidate PPI pair extracted from an input sentence, the system outputs a triple e.g., <PRO_i, REL, PRO_j> then the following features are generated:

Keyword: is a relation word (REL) from the extracted triple. In addition, we also count the number of protein names (C1) and relation words (C2) in each simplified sentence.

Distance: We use two features: D1, D2 (see Figure 4.1a) to measure the distance (number of words/tokens) between PRO1-REL and REL-PRO2 (or between REL-PRO1 and PRO1-PRO2 for group 3).

Height: We use two features: H1, H2 (see Figure 4.1a) to measure the distance from the joint node connecting PRO1 and PRO2 in the parse tree. These features are similar to D1 and D2 except that they measure the number of nodes on the paths from a local root to PRO1 and PRO2.

POS: We use two lists: Pos1, Pos2 (see Figure 4.1a) to store POS and syntactic features from the joint node connecting PRO1 and PRO2 in the parse tree, respectively.

Lexical: This feature is a modification of a bag-of-words (BOW). Instead of using all tokens, we only consider tokens that belong to the *relation list* and a list of prepositions: *and, or, by, through, in, of, to,* and *between*. If a token is a protein (PRO*), then its value is replaced with “KEY”. We use four lists of tokens:

L1: a list of tokens between PRO1 and REL, or between REL and PRO1 for group 3.

L2: a list of tokens between REL and PRO2, or between PRO1 and PRO2 for group 3.

L3: a list of tokens before PRO1.

L4: a list of tokens after PRO2.

Selection of features for each individual group

The important benefit of partitioning data into subsets is that we can select the most appropriate features for each subset. Let us consider the following two cases: a PPI pair in form 4 (compound form) and a PPI pair in form 1 (full clause). For a PPI pair in a compound form, e.g. PRO1-PRO2, the shortest path features proposed in the previous work become useless because no feature can be extracted from this path. In this case, the bag-of-word features seem to be the most appropriate ones. In contrast, for the PPI pairs in form 1, e.g. PRO1 interacts with PRO2, the shortest path, and the tokens between PRO1 and PRO2, play an important role. Based on the properties of each group of extracted PPI pairs, we manually select features that are potentially suitable for that group. Table 4.1 shows the list of features corresponding to each group. Furthermore, we also use the attribute selection function from the WEKA machine learning package [49] to determine the length of the POS and lexical lists, which are limited to maximum 6 attributes.

Machine learning method

Support vector machines (SVM) have been widely used in the PPI extraction task, and has shown competitive performance over other learning methods ([63, 98]). In this work, we use SVM classifier with a default RBF kernel and tune the complexity parameter C by using *CVParameterSelection* function from the WEKA. All individual features mentioned above are combined into a single feature vector for the classification task. Depending on each group of PPI candidates, the number of features ranges from 12 to 18. Table 4.1 summarizes features used for each group of the PPI candidates. For example, for a candidate PPI pair <PRO1, PRO2> extracted from group 1 (as shown in Figure 4.1a), the following features are generated:

REL: mediate; D1: 1, D2: 6; C1: 3, C2: 2; H1: 5, H2:4; L2: association, with, null; Pos2: VB, PP, NP, NN, null, null, true.

Table 4.1. List of features and their usage for each group

Feature	Keyword	Distance	Height	POS	Lexical
Group 1	REL, C1, C2	D1, D2	H1, H2	Pos1,2	L2, L2
Group 2	REL	D1, D2	H1, H2		L1, L2
Group 3	REL, C1, C2	D1, D2	H1, H2	Pos1,2	L2
Group 4	C1, C2				L3, L4
Group 5	REL C1, C2	D1, D2			L3, L4

4.3 Results and Discussion

4.3.1 Data sets

We use five corpora[3] which have been converted into a unified format and are provided by Pyysalo et al. [29]: AIMed, BioInfer, HPDR50, IEPA, and LLL. Table 4.2 shows the quantitative information of all 5 corpora.

Table 4.2. Statistics on 5 corpora

Corpus	AIMed	BioInfer	HPRD50	IEPA	LLL
Positive pairs	1000	2534	163	335	164
All pairs	5834	9666	433	817	330

4.3.2 Evaluation methods

We perform two types of evaluation, a single corpus test and a cross-corpora test. In the single corpus test, we evaluate the performance of the proposed method by 10-fold abstract-wise cross-validation (CV), and use the one-answer-per-occurrence criterion [63]. We split the corpora as recommended by Airola *et al.* [58] in order to directly compare our results against performance of other systems. In the cross-corpora test, we conduct two experiments. First, we use one corpus for training and

[3] http://mars.cs.utu.fi/PPICorpora/GraphKernel.html

test the system on the four remaining corpora. Second, we use four corpora (ALL) for training and test on the remaining corpus.

4.3.3 Performance of PPI extraction algorithm

Table 4.3 shows the results of the PPI extraction algorithm on 5 corpora. In order to calculate recall for each corpus, we use the number of original positive pairs from Table 4.1 (e.g., for AIMed, TP+FN=1000, for BioInfer, TP+FN=2534). It is worth noting that, in some systems [58, 98], self-interaction pairs are removed prior to the evaluation therefore the F-score can be higher if we adopt these settings.

Table 4.3 also presents the properties of each extracted group in each corpus and the difference between corpora. Clearly, group 1 is the most common among all corpora and accounts for more than 50% of true positive pairs of all corpora except for BioInfer. The first three groups are also common for all 5 corpora and account for more than 80% of the extracted pairs. However, group 4 and 5 are more corpus-specific and can be found mostly in AIMed and BioInfer corpora. This information provides more insight for understanding the properties of 5 corpora studied previously by Pyysalo *et al.* [29].

Table 4.3. Results of PPI extraction algorithm on 5 corpora

Corpus	AIMed		BioInfer		HPRD50		IEPA		LLL	
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP
Group 1	500	701	849	648	101	38	170	78	106	13
Group 2	143	112	215	154	14	10	26	26	2	4
Group 3	113	322	330	529	17	23	92	61	32	11
Group 4	22	55	170	73	0	8	4	3	0	2
Group 5	39	192	134	182	6	5	3	6	0	1
Total	**817**	**1382**	**1698**	**1586**	**138**	**84**	**295**	**174**	**140**	**31**
Recall	81.7%		67%		84.7%		88.1%		85.4%	
Precision	37.2%		51.7%		62.2%		62.9%		81.9%	
F-score	**51.1%**		**58.4%**		**71.7%**		**73.4%**		**83.6%**	

The results in Table 4.3 demonstrate that the proposed algorithm is capable of extracting of more than 80% of positive pairs (recall >80%) on AIMed, HPRD50, IEPA, and LLL corpora. This means that these four corpora share some common patterns or the same annotation policy. However, our method can only extract 67% positive pairs from BioInfer. To find out why our algorithm has lower recall on the

BioInfer corpus, we examine sample sentences from BioInfer which the system failed to extract. By analyzing these sentences, we discovered that in some cases, BioInfer has a special annotation policy. Consider, for example, the following sentence:

BioInfer.d436.s0: *Moreover, ectopic expression of PRO0 and PRO1 can stimulate the PRO2 promoter in an PRO3-dependent manner, and this is inhibited by coexpression of the PRO4 (PRO5) PRO6.*

For this input sentence, our system extracts only 4 true positive pairs, but the number of positive PPIs pairs provided by the BioInfer corpus is 11 (see sentence *BioInfer.d436.s0* in BioInfer corpus for more detail). Therefore, to increase recall on BioInfer, additional study of this corpus is needed.

Table 4.3 also shows that the performance of algorithm 1 is comparable with other machine learning based systems. Moreover, it outperforms the best rule-based system on all 5 corpora (see [34]) for more details).

4.3.4 Single corpus evaluation

With the extracted PPI pairs obtained in the previous step, we use a standard SVM classifier to classify them. Comparing with the original data from Table 4.1, the number of positive instances and negative instances of the AIMed and BioInfer corpora is quite balance (see Table 4.3). To calculate the F-score, we accumulated all true positives (TP) and false positive (FP) pairs from the 10-fold CV test (reported by WEKA via the confusion matrix). For TP+FN values, similarly to section 3.3, we use the original positive pairs of each corpus in Table 4.1. For the IEPA, HPRD50, and LLL corpora, however, we only use the first three groups of extracted pairs because the number of candidate pairs in the group 4 and 5 is too small for the 10-fold cross-validation.

Table 4.4. Results of the 10-fold abstract-wise cross-validation on 5 corpora. ΔP and ΔF show the increase in precision and F-score and ΔR shows the decrease in recall compared to their corresponding values in Table 4.3. The values are shown in percentage (%).

Corpus	Precision	ΔP	Recall	ΔR	F-Score	ΔF
AIMed	55.3	18	68.5	(13.2)	61.2	10
BioInfer	61.7	10	57.5	(9.5)	60.0	2
HPRD50	70.2	8	77.9	(6.8)	73.8	2
IEPA	67.4	5	83.9	(4.2)	74.7	1
LLL	84.1	2	84.1	(1.3)	84.1	1

Table 4.4 shows the performance of the classifier on five corpora. The results indicate that by using our feature vectors, the classifier can further boost the performance of the overall system. When comparing precision on each corpus before and after applying classification, we can see a significant increase in precision (ΔP values). Among all corpora, the AIMed corpus gains the most increase in precision with ΔP of 14%. Even on a small corpus like LLL, with already very high precision, the final precision still gains 2%. For each corpus, recall decreases after applying classification but the final F-scores increase for all corpora and range from 1% to 10%. Furthermore, Table 4.4 shows that our system achieves the best performance on three corpora: IEPA, HPRD50, and LLL using 10-fold cross-validation when compared to other results reported so far.

Table 4.5. Performance comparison between full dataset, filtered dataset and partitioned dataset using 10-fold CV on five corpora. Full dataset and filtered dataset use the same feature sets. The performance is measured by the F-score (%).

Corpus	AIMed		BioInfer		HPRD50		IEPA		LLL	
	P	F	P	F	P	F	P	F	P	F
Full dataset	51.8	52.5	56.5	58.7	65.7	68.6	66.0	72	78.9	83.6
Filtered dataset	55.8	55.2	63.2	59.6	76	76	67.8	72.9	85.2	84.4
Split dataset	55.3	61.2	61.7	60	70.2	73.8	67.4	74.7	84.1	84.1

To study the benefit of filtering data (obtained after applying rules on the full dataset) and partitioning data, we conduct the experiment in which the full dataset and the filtered dataset use all features in Table 4.1. However, in this experiment the REL feature is not available, D1 and D1 features are replaced by D1+D2, and L1 and L2 features are merged. Table 4.5 summarizes the performance of the system on the full dataset (all PPI pairs), filtered dataset, and partitioned dataset. The results from Table 4.5 show that when evaluating on filtering dataset, the system has better performance on all five corpora compared with full dataset. In addition, the performance further improve on three largest corpora (AIMed, BioInfer, IEPA) when we partition data and select feature specific for each sub-dataset. This clearly shows the benefit of our hybrid approach that combines rule with partition data.

Table 4.6 illustrates a comparison of our system (BKS) against recent work on the AIMed corpus, which is considered the *de-facto* benchmark for the PPI extraction task. However, the comparison may not be straightforward because these PPI systems can use different text preprocessing and learning methods. The results show that our system with its F-score of 61.2% is comparable to the state-of-the-art systems proposed by Miwa *et al*. [63] and Sætre *et al.* [45]. In addition, our approach significantly outperforms other methods based on one parser's output.

Table 4.6. Performance comparison with other systems on AIMed

System	Description	F-score (%)
Sætre [45]	Feature-based, two parsers	64.2
Miwa [50]	Multiple kernels, two parsers	60.8
Kim [98]	Walk-weighted subsequence kernels, one parser	56.6
Airola [58]	All-paths graph kernel, one parser	56.4
Niu [95]	Linear kernel, one parser	53.5
BKS	RBF kernel, one parser	61.2

4.3.5 Cross-corpora evaluation

In addition to the standard 10-fold cross-evaluation, recent studies also suggested to test existing approaches to PPI extraction using cross-corpora. This type of evaluation can show whether a system trained on one corpus can perform equally well on other corpora, with an assumption that these corpora addressed the same PPI task. In this setting, one corpus is used for training and the remaining corpora are used for testing. For the evaluation, we only use the two largest corpora, i.e. AIMed and BioInfer, for training since the other corpora are too small after partitioning onto 5 groups. Furthermore, we conduct the experiment proposed by Fayruzov *et al.* [100] and Kabiljo *et al.* [34], in which four corpora are used for training and the remaining corpus for testing.

Table 4.7 shows the results of the cross-corpora evaluation on five corpora. Here, the columns correspond to the training set and the rows correspond to test sets. Table 4.7 illustrates that our system achieves the best performance when BioInfer is used as the training set. Note that we do not use data from group 4 and group 5 due to theirs corpus-specific properties, therefore the performance might be higher if these groups are used. This finding is also consistent with the evaluation results by Miwa *et al.* [50] and Airola *et al.* [58]. In their work, better performance is also obtained with BioInfer being used for training.

Interestingly, Table 4.7 shows that even though the F-score on each corpus in the cross-corpora test is lower than F-score in the single corpus test, the precision is significantly higher when compared against the initial values in Table 4.3. This means that performance on all corpora can be boosted when precision is given a priority. This also implies that the classifier is able to learn from cross-corpora training data. In other words, the proposed feature vectors are compatible across the corpora.

Table 4.7. Results of the cross-corpora test on five corpora. Columns correspond to training and rows correspond to testing corpora. ALL refers to a situation where four corpora are used for training and the remaining corpus for testing. Precision (P) and F-score (F) are shown in percentage (%).

Corpus	AIMed		BioInfer		ALL	
	P	F	P	F	P	F
AIMed	-	-	44.4	55.2	44.1	55.1
BioInfer	57.1	54.4	-	-	64.3	50.5
HPRD50	67.0	72.3	69.9	74.0	68.8	70.8
IEPA	67.3	73.6	68.6	73.8	70.1	72.9
LLL	85.7	83.0	85.0	82.4	86.0	80.1

Table 4.8 shows the performance of our system (BKS) in the cross-corpora setting compared with other approaches. The results in Table 4.8 demonstrate that BKS outperforms others when AIMed and BioInfer corpora are used as training sets. In addition, for all evaluation tests, we use the same settings for training corpora except for the complexity term (C parameter of the RBF kernel). This is one step closer to the real world situation.

Table 4.8. Performance comparison with other systems using cross-corpora evaluation. Columns correspond to training and rows correspond to test sets. The performance is measured by the F-score (%).

	AIMed				BioInfer			
	BKS	kBSP [33]	Miwa [50]	Airola [58]	BKS	kBSP [33]	Miwa [50]	Airola [58]
AIMed	-	-	-	-	**55.2**	40.3	49.6	47.2
BioInfer	**54.4**	24.8	53.1	47.1	-	-	-	-
HPRD50	**72.3**	51.0	68.3	69.0	**74.0**	69.8	68.3	63.9
IEPA	**73.6**	36.7	68.1	67.4	**73.8**	72.4	71.4	68.0
LLL	**83.0**	44.4	73.5	74.5	**82.4**	80.6	76.9	78.0

4.3.6 Performance time

Another important aspect in PPI extraction is the computational time taken to train and test the system. A previous study by Van Landeghem *et al.* [91] has shown that in order to reduce the number of computational resources used by machine learning, one has to consider a trade-off between performance (in this case measured by the F-score) and the performance time required by ML. Despite this fact, only few systems report on how many computational resources their systems use for training and testing. Miwa *et al*. [63] demonstrate that by using suitable features as well as a learning method, they can improve the performance of their previously proposed system [50]. In addition, Fayruzov *et al.* [100] have pointed out that the more kernels the system uses, the more computational resources are needed.

Table 4.9 shows the running time of our system on the full dataset compared against partitioned dataset of the AIMed corpus. On the partitioned dataset, the system runs faster on both implementations of RBF kernel. In addition, since our method partitions data into 5 groups, we also test it in a node with 8 CPUs (Xeon 3.0 GHz). In this test, we use the RBF kernel from LibSVM and the classifier is run in parallel. The maximum time used by the system is 12 seconds. This means that our system not only runs fast on a single PC, but can also be used in parallel, which is particularly suited for large-scale experiments.

Table 4.9. Performance time for 10-fold CV of full and partitioned dataset of AIMed corpus. These values exclude time for parsing and text preprocessing. Two implementations of RBF kernel from WEKA and LibSVM[4] are used. The experiment is run on a PC with an Intel 3.2GHz processor and 4GB of RAM

Dataset	WEKA - RBF	LibSVM - RBF
Full dataset	10812 seconds	194 seconds
Partitioned dataset	77 seconds	39 seconds

4.4 Conclusions

In this paper, we propose a novel method for extracting PPI pairs from literature. Our approach combines the strength of both syntactic rules and machine learning classification. The evaluation on five benchmark corpora has shown that our system achieves results comparable with the best PPI extraction methods on a single corpus. It outperforms other systems on cross-corpora test and has fast running time.

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

The proposed method consists of two phases: partitioning data into subsets then extract candidate PPI pairs from these subsets, and classifying extracted PPI pairs. The advantages of this method are four-fold. First, it filters out a significant amount of negative (non-interaction) PPI pairs, thus balancing the number of positive and negative pairs in training data. Second, by partitioning data into subsets, we can select the most appropriate features for each subset, which potentially improves the final performance. Third, our system only uses a small set of features and therefore performs the best in terms of performance time. Finally, the classifier can be run in parallel on each subset, which is desirable for the large- scale experiments.

In addition, our method uses five syntactic rules therefore it is generic and can be easily applied to new data sets. Furthermore, it is easy to set-up because it only uses publicly available NLP tools and a standard machine learning package. In addition, the proposed method can also be extended to extract new relation types in biomedical text, e.g. complex event extraction [118, 121].