<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>A hybrid approach to extract protein-protein interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Bui, Quoc-Chinh; Katrenko, Sophia; Sloot, Peter M. A.</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2010</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/18848">http://hdl.handle.net/10220/18848</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>© 2010 The Author(s). Published by Oxford University Press. All rights reserved. For Permissions, please email: <a href="mailto:journals.permissions@oup.com">journals.permissions@oup.com</a>.</td>
</tr>
</tbody>
</table>
A hybrid approach to extract protein–protein interactions

Quoc-Chinh Bui1,*, Sophia Katrenko2 and Peter M. A. Sloot1
1Computational Science and 2Theory of Computer Science, Informatics Institute, University of Amsterdam, Science Park 904, Amsterdam, The Netherlands

ABSTRACT

Motivation: 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.

Results: In this study, we propose a novel algorithm for extracting PPIs from literature which consists of two phases. First, we automatically categorize the data into subsets based on its semantic properties and extract candidate PPI pairs from these subsets. Second, we apply support vector machines (SVMs) 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 with the state-of-the-art PPI extraction systems. Furthermore, our system achieves the best performance which are comparable with the state-of-the-art PPI extraction systems. Since full parsing produces more elaborate syntactic information than shallow parsing, PPI extraction systems based on full parsing can potentially yield better results (Miyao et al., 2009). 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 ML 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 ML method is then used to distinguish between positive and negative instances (Niu et al., 2010; Sætre et al., 2010).

Many linguistic features and ML 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 (2005) 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. (2006) extended this approach by using a bag-of-words (BOWs) and adding a local context kernel. The second group exploits syntactic features of a sentence. Sætre et al. (2007) used various syntactic path features and context features related to words before, between and after two interacting entities. Katrenko and Adriaans (2007) proposed a method based on information found in the predefined levels of the dependency trees, such as local dependency contexts of the protein names and tree’s roots. Kim et al. (2008) enhanced previous work by proposing a walk kernel which exploits 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. (2008) 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.

Recently, many ML-based methods have employed NLP techniques such as shallow parsing or full parsing (Björne et al., 2010; Giles and Wren 2008; Giuliano et al., 2006; Miwa et al., 2010). Since full parsing produces more elaborate syntactic information than shallow parsing, PPI extraction systems based on full parsing can potentially yield better results (Miyao et al., 2009). 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 ML 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 ML method is then used to distinguish between positive and negative instances (Niu et al., 2010; Sætre et al., 2010).

Many linguistic features and ML 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 (2005) 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. (2006) extended this approach by using a bag-of-words (BOWs) and adding a local context kernel. The second group exploits syntactic features of a sentence. Sætre et al. (2007) used various syntactic path features and context features related to words before, between and after two interacting entities. Katrenko and Adriaans (2007) proposed a method based on information found in the predefined levels of the dependency trees, such as local dependency contexts of the protein names and tree’s roots. Kim et al. (2008) enhanced previous work by proposing a walk kernel which exploits 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. (2008) 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. (2000), Niu et al. (2010) and Van Landeghem et al. (2008) also studied individual impact of a variety of feature types on the PPI extraction task. In addition, a study of Miyao et al. (2009) 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. (2009a, b) 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 datasets. However,
the studies by Fayruzov et al. (2009) and Kim et al. (2010) 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. (2009b) who showed that excluding the BOWs 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 remain. Adding more features might sometimes improve performance, but they can introduce noise in other cases, or require more computational resources (Kim et al., 2010). In this study, we propose a novel method that consists of two phases. First, we apply semantic rules to partition the dataset 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 learning method and make it computationally effective. In general, a PPI extraction system consists of two subtasks: recognizing protein names (NER) and extracting PPI pairs. However, this study only focuses on the PPI extraction task with an assumption that relevant named entities were given.

2 METHODS

The workflow of the proposed system is as follows:

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

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 (Klein and Manning, 2003).

2.1.1 Processing protein names

In order to improve accuracy of the parser, we replace all mentioned protein names with a placeholder, 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.

2.1.2 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.

2.2 Extracting candidate PPI pairs

Previous studies (Bui et al., 2010; Fundel et al., 2007; Rinaldi et al., 2010) 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, word REL (verb) word PRO.
  - Example: PRO1 interacts with PRO2.

- Form 2: PRO, word REL (noun/verb) word PRO.
  - Example: PRO1 has a weak interaction with PRO2.

- Form 3: REL (noun) word PRO, word PRO.
  - 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, and PRO can be any protein pair with ∑ ≥ 0 (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 (complex form): PRO*PROi+1 or PRO*PROi+1 or PRO*PROi+1.
  - Example: PRO1/PRO2 binding; PRO1-PRO2 compound.

- Form 5 (complex form): PRO, word REL; PRO, 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* and REL, PRO and PRO can be any protein pair with ∑ ≥ 0 (e.g. <PRO1, PRO2>). 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*)) S+ (VP (PRO*)).
- (b) Partial clause: VP (NP (PRO*)) S+ (S (VP (PRO*)).
- (c) Subclause: S (NP (PRO*)) S+ (VP (S (SBAR (PRO*)).
- (d) NP pattern: NP <(NP (REF) S+ (PP (PRO*))>.

Here S (clause), NP (noun phrase), VP (verb phrase), PP (prepositional phrase) and SBAR (subclause) are constituents of the parse tree. PRO* is a placeholder for a protein name, and REL is a cue word of the input sentence. 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, word REL; PRO, word REL.
  - Example: PRO1, PRO2 interact; in PRO1, PRO2 complex.

2.2.1 Relation list

We created a relation list by combining the relation lists used in the previous work by Chowdhary et al. (2009), Fundel et al. (2007) and Niu et al. (2010).

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

- Procedure for Form 1: this procedure requires a parse tree which contains full clause, partial clause or subclause 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.
A hybrid approach to extract PPIs

Procedure for Form 5: form candidate PPI pairs from a simplified sentence if they satisfy the following form: \( \text{PRO}_i \text{ word } \text{REL}_i \text{ word } \text{PRO}_j \text{ and if the distance from the PRO}_i \) to the \text{REL} is shorter than five tokens.

For these procedures, the extracted candidate PPI pairs obtained from the same procedure are grouped together, i.e. outputs from Form 1 are grouped into group 1. Among these groups, group 2 overlaps group 1 when \text{REL} is a verb, with the purpose to recover PPI pairs that the group 1 failed to extract due to the parser errors (Miyao et al., 2009).

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 1 below).

**Algorithm 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** form 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 for

2.3 Features

In this section, we propose feature sets for ML classification. Our feature sets are combinations of some features that were previously proposed by Chowdhary et al. (2009), Giuliano et al. (2006), Miwa et al. (2009a, b) and Niu et al. (2010). For each candidate PPI pair extracted from an input sentence, the system outputs a triple, e.g. \( \text{PRO}_i, \text{REL}, \text{PRO}_j \) then the following features are generated:

- **Keyword:** is a relation word (\text{REL}) from the extracted triple. In addition, we also count the number of protein names (\text{C1}) and relation words (\text{C2}) in each simplified sentence.
- **Distance:** we use two features: \text{D1}, \text{D2} (Fig. 1a) to measure the distance (number of word tokens) between \text{PRO}_i, \text{REL} and \text{REL}-\text{PRO}_j \text{ or between REL}-\text{PRO}_j \text{ and PRO}_j \text{ for group 3}.
- **Height:** we use two features: \text{H1}, \text{H2} (Fig. 1a) to measure the distance from the joint node connecting \text{PRO}_i \text{ and PRO}_j \text{ 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 \text{PRO}_i \text{ and } \text{PRO}_j \text{.}
- **POS:** we use two lists: Pos1, Pos2 (Fig. 1a) to store POS and syntactic features from the joint node connecting \text{PRO}_i \text{ and PRO}_j \text{ in the parse tree, respectively.}
- **Lexical:** this feature is a modification of a BOWs. 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 (\text{PRO}_i), then its value is replaced with ‘KEY’. We use four lists of tokens:
  - L1: a list of tokens between \text{PRO}_i \text{ and } \text{REL}, or between \text{REL} \text{ and } \text{PRO}_j \text{ for group 3.}
  - L2: a list of tokens between \text{REL} \text{ and } \text{PRO}_j \text{, or between } \text{PRO}_i \text{ and } \text{PRO}_j \text{ for group 3.}

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{fig1.png}
\caption{Examples of parse trees output from the Stanford lexical parser (a) parse tree satisfies full clause pattern. (b) parse tree satisfies NP pattern and (c) parse tree satisfies subclause pattern. Some features (D1, D2, H1, H2) and paths from the join node connecting a given protein pair (PRO1, PRO2) are also shown in (a).}
\end{figure}
3 RESULTS AND DISCUSSION

3.1 Datasets

We use five corpora (http://mars.cs.utu.fi/PPICorpora/GraphKernel.html) which have been converted into a unified format and are provided by Pyysalo et al. (2008): AIMed, BioInfer, HPRD50, IEPA and LLL. Table 2 shows the quantitative information of all five corpora.

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 (Miwa et al., 2009b). We split the corpora as recommended by Airola et al. (2008) in order to directly compare our results against the performance of other systems. In the cross-corpora test, we conduct two experiments. First, we use one corpus for training and test the system on the four remaining corpora. Second, we use four corpora (ALL) for training and test on the remaining corpus.

Precision, recall and F-score are used as evaluation metrics.

\[
F\text{-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},
\]

where Precision = TP / (TP + FP) and Recall = TP / (TP + FN).

Table 1. List of features and their usage for each group

<table>
<thead>
<tr>
<th>Feature</th>
<th>Keyword</th>
<th>Distance</th>
<th>Height</th>
<th>POS</th>
<th>Lexical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>Rel, C1, C2</td>
<td>D1, D2</td>
<td>H1, H2</td>
<td>Pos1,2</td>
<td>L1, L2</td>
</tr>
<tr>
<td>Group 2</td>
<td>Rel</td>
<td>D1, D2</td>
<td>H1, H2</td>
<td>L1, L2</td>
<td></td>
</tr>
<tr>
<td>Group 3</td>
<td>Rel, C1, C2</td>
<td>D1, D2</td>
<td>H1, H2</td>
<td>Pos1,2</td>
<td>L1, L2</td>
</tr>
<tr>
<td>Group 4</td>
<td>C1, C2</td>
<td>L3, L4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Group 5</td>
<td>Rel, C1, C2</td>
<td>D1, D2</td>
<td>L1, L4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

L3: a list of tokens before PROi.
L4: a list of tokens after PROi.

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 BOW 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 1 shows the list of features corresponding to each group. Furthermore, we also use the Ranker (attribute selection) method from the WEKA ML package (Witten and Frank, 2005) to determine the length of the POS and lexical lists, which are limited to maximum six attributes.

ML method: Support vector machines (SVMs) have been widely used in the PPI extraction task, and has shown competitive performance over other learning methods (Kim et al., 2010; Sartes et al., 2010). 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 candidate PPI pair.

For more details on text preprocessing, PPI extraction and feature generation steps, see Supplementary source code provided in this article.

Table 2. Statistics on five corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>AIMed</th>
<th>BioInfer</th>
<th>HPRD50</th>
<th>IEPA</th>
<th>LLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive pairs</td>
<td>1000</td>
<td>2534</td>
<td>163</td>
<td>335</td>
<td>164</td>
</tr>
<tr>
<td>All pairs</td>
<td>5834</td>
<td>9666</td>
<td>433</td>
<td>817</td>
<td>330</td>
</tr>
</tbody>
</table>

Table 3. Results of PPI extraction algorithm on five corpora

<table>
<thead>
<tr>
<th>Corpus</th>
<th>AIMed</th>
<th>BioInfer</th>
<th>HPRD50</th>
<th>IEPA</th>
<th>LLL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>FP</td>
<td>TP</td>
<td>FP</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>Group 1</td>
<td>500</td>
<td>701</td>
<td>849</td>
<td>648</td>
<td>101</td>
</tr>
<tr>
<td>Group 2</td>
<td>143</td>
<td>112</td>
<td>215</td>
<td>154</td>
<td>14</td>
</tr>
<tr>
<td>Group 3</td>
<td>113</td>
<td>322</td>
<td>330</td>
<td>529</td>
<td>17</td>
</tr>
<tr>
<td>Group 4</td>
<td>22</td>
<td>55</td>
<td>170</td>
<td>73</td>
<td>0</td>
</tr>
<tr>
<td>Group 5</td>
<td>39</td>
<td>192</td>
<td>134</td>
<td>182</td>
<td>6</td>
</tr>
<tr>
<td>Total</td>
<td>817</td>
<td>1382</td>
<td>1698</td>
<td>1586</td>
<td>138</td>
</tr>
<tr>
<td>Recall (%)</td>
<td>81.7</td>
<td>67</td>
<td>84.7</td>
<td>88.1</td>
<td>85.4</td>
</tr>
<tr>
<td>Precision (%)</td>
<td>37.2</td>
<td>51.7</td>
<td>42.2</td>
<td>62.2</td>
<td>62.9</td>
</tr>
<tr>
<td>F-score (%)</td>
<td>51.1</td>
<td>58.4</td>
<td>71.7</td>
<td>73.4</td>
<td>83.6</td>
</tr>
</tbody>
</table>

3.3 Performance of PPI extraction algorithm

Table 3 shows the results of the PPI extraction algorithm on five corpora. In order to calculate recall for each corpus, we use the number of original positive pairs from Table 1 (e.g. for AIMed, TP + FN = 1000, for BioInfer, TP + FN = 2534). It is worth noting that, in some systems (Airola et al., 2008; Kim et al., 2010), self-interaction pairs are removed prior to the evaluation; therefore, the F-score can be higher if we adopt these settings.

Table 3 also presents the properties of each extracted group in each corpus and the differences between corpora. Clearly, group 1 is the most common among all corpora and accounts for >50% of TP pairs of all corpora except for BioInfer. The first three groups are also common for all five corpora and account for >80% of the extracted pairs. However, groups 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 five corpora studied previously by Pyysalo et al. (2008).
The results in Table 3 demonstrate that the proposed algorithm is capable of extracting of >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 manner, and this is inhibited by coexpression of the PRO4 (PRO5) PRO6.

For this input sentence, our system extracts only four TP pairs, but the number of positive PPI pairs provided by the BioInfer corpus is 11 (see sentence BioInfer.d436.d0 in BioInfer corpus for more detail). Therefore, to increase recall on BioInfer, additional study of this corpus is needed.

Table 3 also shows that the performance of algorithm 1 is comparable with other ML-based systems. Moreover, it outperforms the best rule-based system on all five corpora [see Kabiljo et al. (2009) for more details].

### 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 dataset from Table 1, the number of positive and negative instances of the AIMed and BioInfer corpora are quite balanced (Table 3). To calculate the F-score, we accumulated all TP and 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 1. For the IEPA, HPRD50 and LLL corpora, however, we only use the first three original positive pairs of each corpus in Table 1. For TP+FN values, similarly to Section 3.3, we use the confusion matrix. For TP+FN values, similarly to Section 3.3, we use the confusion matrix.

Table 4 shows the performance of the classifier on five corpora. The results indicate that the classifier 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 shows that our system achieves the best performance on three corpora: IEPA, HPRD50 and LLL using 10-fold CV when compared with other results reported so far.

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 the same features. The performance is measured by the F-score (%).

Table 6. Performance comparison with other systems on AIMed

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
<th>F-score (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satree (2010)</td>
<td>Feature-based, two parsers</td>
<td>64.2</td>
</tr>
<tr>
<td>Miwa (2009a)</td>
<td>Multiple kernels, two parsers</td>
<td>65.7</td>
</tr>
<tr>
<td>Kim (2010)</td>
<td>Walk-weighted subsequence kernels, one parser</td>
<td>58.6</td>
</tr>
<tr>
<td>Airola (2008)</td>
<td>All-paths graph kernel, one parser</td>
<td>56.4</td>
</tr>
<tr>
<td>Niu (2010)</td>
<td>Linear kernel, one parser</td>
<td>53.5</td>
</tr>
<tr>
<td>BKS</td>
<td>RBF kernel, one parser</td>
<td>51.2</td>
</tr>
</tbody>
</table>

To further improve on three largest corpora (AIMed, BioInfer and IEPA) when we partition data and select feature specific for each sub-corpus. This clearly shows the benefit of our hybrid approach that combines rule with partition data.

Table 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 with the state-of-the-art systems proposed by Miwa et al. (2009a) and Sætre et al. (2010). In addition, our approach significantly outperforms other methods based on one parser’s output.

### 3.5 Cross-corpora evaluation

In addition to the standard 10-fold CV, recent studies also suggested to test existing approaches to PPI extraction using cross-corpus. This type of evaluation can show whether a system trained on one corpus can perform equally well on other corpora, with an...
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. (2008) has shown that in order to reduce the number of computational resources used by ML, one has to consider trade-offs between performance (in this case measured by the F-score) and the computational time required by ML. Despite this 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 five groups. Furthermore, we conduct the experiment proposed by Fayruzov et al. (2009) and Kabiljo et al. (2009), in which four corpora are used for training and the remaining corpus for testing.

Table 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 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 to group 5 due to their 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. (2009a) and Airola et al. (2008). In their work, better performance is also obtained with BioInfer being used for training.

Interestingly, Table 7 shows that even though the F-score on each corpus in the cross-corpora test is lower than F-score on the single corpus test, the precision is significantly higher when compared against the initial values in Table 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 8 shows the performance of our system (BKS) in the cross-corpora setting compared with other approaches. The results in Table 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. However, Miwa et al. (2009a) and Airola et al. (2009b) have obtained higher performance compared with ours by applying corpus weighting.

### 3.6 Computational performance

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. (2008) has shown that in order to reduce the number of computational resources used by ML, one has to consider a trade-off between performance (in this case measured by the F-score) and the computational 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. (2009b) demonstrate that by using suitable features as well as a learning method, they can improve the performance of their previously proposed system (Miwa et al., 2009a). In addition, Fayruzov et al. (2009) have pointed out that the more kernels the system uses the more computational resources are needed.

Table 9 shows the running time of our system on the full dataset compared against partitioned dataset of the AIMed corpus. The system runs much faster on the partitioned dataset with both RBF kernels from WEKA and LibSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/). In addition, since our method partitions data into five groups, we also test it in a node with eight CPUs (Xeon 3.0 GHz processor and 4 GB of RAM). In this test, we use the RBF kernel from LibSVM and the classifiers are run in parallel. The maximum time used by the system is 12 s. 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.

### 4 CONCLUSIONS

In this article, we propose a novel method for extracting PPI pairs from literature. Our approach combines the strength of both semantic rules and ML 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.

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...
A hybrid approach to extract PPI

4-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 computational resources. 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 semantic rules; therefore, it is generic and can be easily applied to new datasets. Furthermore, it is easy to set-up because it only uses publicly available NLP tools and a standard ML package. In addition, the proposed method can also be extended to extract new relation types in biomedical text, e.g. complex event extraction (Björne et al., 2010; Miwa et al., 2010, Rinaldi et al., 2010).

For future work, we plan to integrate a NER tagger into our system in order to study the effect of its performance when a gold-standard NER is not given. In addition, the performance of the overall PPI extraction task largely depends on the output of the extraction phase. Increasing recall in this phase would further boost the performance, especially in case of the BioInfer corpus.

ACKNOWLEDGEMENTS

The authors sincerely thank Dr Makoto Miwa and Rick Quax for their useful comments and suggestions.

Funding: This research was supported by the European Union through the DynaNets project http://www.dynanets.org EU grant agreement no: 233847; the Vietnamese Overseas Training Program, and the VL-e grant (BSIK grant).

Conflict of Interest: none declared.

REFERENCES

Niu,Y. et al. (2009) Evaluation of linguistic features useful in extraction of interactions from PubMed. Application to annotating known, high-throughput and predicted interactions in DDI. Bioinformatics, 26, 111–119.

[13:34 16/12/2010 Bioinformatics-btq620.tex]