25

Extract Protein-Protein Interactions From the Literature Using Support Vector Machines with Feature Selection

Yifei Chen, Feng Liu and Bernard Manderick
Vrije Universiteit Brussel
Belgium

1. Introduction

The objective of text mining is to automatically identify, extract, manage, integrate and exploit the information in texts (Ananiadou & McNaught, 2006). In order to understand biological texts, it is not enough to know what the extracted proteins are. Also the interactions between them should be extracted. Therefore, extracting relations between proteins is an important and more advanced text mining task in biological domain.

The study of the protein-protein interactions (PPI) is one of the most pressing problems. Characterizing protein interaction pairs is crucial to understand not only the functional role of individual proteins but also the organization of entire biological processes (Krallinger et al., 2007). Several approaches have been applied to PPI pair extraction including purely statistical co-occurrence approaches (De Bruijn & Martin, 2002; Craven, 1999), pattern-matching approaches (Baumgartner Jr. et al., 2007; Ray & Craven, 2001; Hakenberg et al., 2008) and machine learning approaches such as maximum entropy (Grover et al., 2007) and support vector machines (SVMs) (Airola et al., 2008; Bunescu et al., 2005; Zelenko et al., 2003).

In this chapter, we propose a *Protein-Protein Interaction Pair Extractor* (*PPIEor*) to extract PPI pairs from the biological literature. *PPIEor* is essentially a SVM for binary classification, which uses a linear kernel and a rich and informative set of features based on linguistic analysis, contextual words, interaction words, interaction patterns, specific domain information and so forth.

2. Methods

2.1 System description

Fig. 1 shows the overall architecture of *PPIEor* consisting of a number of components including a preprocessor, a feature extractor and feature selector, a SVM-based classifier and a post-processor.

PPIEor's input data are the biological articles annotated with protein names. Before extracting the features, the inputs have to go first through the *preprocessor* which includes a *clause parsing* module, a *coreference resolution* module and a *pair extraction* module. Then these processed articles and the extracted candidate PPI pairs together with external interaction databases

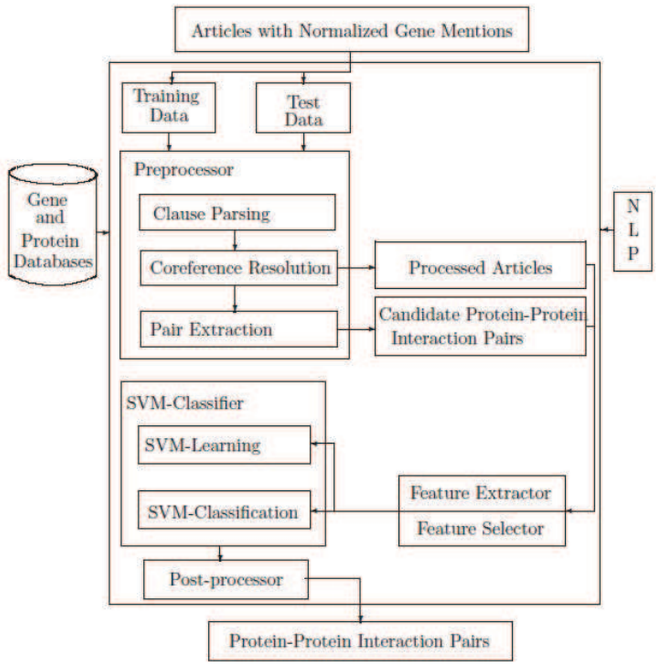

Fig. 1. The overall architecture of the Protein-Protein Interaction Pair Extractor (*PPIEor*)

like MINT[1] and IntAct[2] are used by the *feature extractor* and the *feature selector* to extract a rich and informative set of features. Next, the *binary SVM classifier*, the core component of *PPIEor*, is used to predict whether the candidate PPI pairs are correct or not. Finally, the output from the SVM classifier is combined with the self-interaction protein pairs generated by the *post-processor* to produce the final PPI pairs.

2.2 Data set

The data set used to train, tune and evaluate *PPIEor* consists of articles that have Structured Digital Abstracts (SDAs) (Ceol et al., 2008) from the journal, *FEBS Letters*[3]. More specifically, the total of 61 articles taken from DOI:10.1016/j.febslet.2008.01.064 to DOI:10.1016/j.febslet. 2008.11.009 are used as the training set. And the total of 39 articles with SDAs in FEBS Letters (from DOI:10.1016/j.febslet.2008.11.022 to DOI:10.1016/j.febslet .2009.03.013) are used as the test set. The SDA is an extension of the regular journal article abstract containing the PPI relations between the protein pairs mentioned in the article. Making use of these SDAs, we can obtain the PPI pairs of the articles in the data set. We denote this data set as $Data_{FEBS}$ and it contains 228 unique PPI pairs in the training set and 123 unique PPI pairs in the test set.

2.3 Protein name annotation

As it can be seen in Fig. 1, *PPIEor*'s input data are the articles annotated with *protein names*. Because the interactions only exist between proteins, protein names have to be recognized before we can extract the PPI pairs. Therefore, after building $Data_{FEBS}$, the next step is to annotate $Data_{FEBS}$ with protein names.

[1]http://mint.bio.uniroma2.it/mint/ [accessed on 20/03/2010]
[2]http://www.ebi.ac.uk/intact/ [accessed on 20/03/2010]
[3]http://www.febsletters.org/ [accessed on 20/03/2010]

In FEBS Letters, protein names are represented by database identifiers (database ID) of UniProt[4]. Therefore, we manually annotated articles with *golden standard* protein names which were obtained from the Structured Digital Abstracts (SDAs) of the selected articles. In this way we focus solely on the performance of *PPIEor* as a standalone system and avoid the errors derived from protein name recognition.

2.4 Preprocessor

Before passing on the input articles that have been annotated with the protein names to the SVM classifier in order to extract the features, they will first go through the preprocessor containing 1) a clause parsing module to split complex sentences into simpler units, 2) a coreference resolution module to find which protein a pronoun refers to, and 3) a pair extraction module to distill the candidate PPI pairs for the SVM classifier. These components are discussed in detail below. The sentence S_{FEBS}

> Our previous results revealed that *Q9HBI1* associates with PIX/ARHG EF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals which activate *P60766* and *P63001*, small Rho GTPases.

taken from the input article DOI:10.1016/j.febslet.2008.01.064 in *FEBS Letters* and annotated with the UniProt identifiers *Q9HBI1*, *Q8K4I3*, *O55222*, *P60766* and *P63001* is used to illustrate the preprocessing steps.

2.4.1 Clause parsing

Sometimes compound sentences are too complex to analyze and may induce too much irrelevant and noisy information. Therefore instead of using the whole sentence, we separate it into several unit structures called *clauses* (Ejerbed, 1988). Based on these simpler structures, interaction relations can be extracted more easily and more efficiently because the candidate PPI pair extraction can be limited to the pairs appearing within the same clause rather than in the same whole sentence. Hence the amount of the false PPI pairs can be reduced a lot. So we propose to use the clause-based representation for the input articles in *PPIEor*.

We design the clause parsing module based on a widely used statistical syntactic parser, *nlparser* (Charniak & Johnson, 2005), to split each compound sentence into a main sentence and several clauses. Charniak & Johnson (2005) reported that the nlparser could obtain a $F_{\beta=1}$ measure of 91.0 on the sentences of length 100 or less and it could produce multiple-best parses. Here we use the first-best parses to decompose the sentences. However, producing a full parse tree sometimes fails due to grammatical inaccuracies. Hence, when the nlparser fails to output a parse tree, we use the original sentence instead.

After this step, the sentence S_{FEBS} is split into 1 main sentence S_{MAIN} and 2 clauses $S_{CLAUSE1}$ and $S_{CLAUSE2}$:

- S_{MAIN}: Our previous results revealed that
- $S_{CLAUSE1}$: *Q9HBI1* associates with PIX/ARHGEF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals
- $S_{CLAUSE2}$: which activate *P60766* and *P63001*, small Rho GTPases.

[4]http://www.uniprot.org/ [accessed on 20/03/2010]

2.4.2 Coreference resolution

After splitting the sentences into clauses another essential step needs to be done, i.e. coreference resolution. For example, in $S_{CLAUSE2}$ "which" refers to "O55222". Now suppose there are 2 PPI pairs in $S_{CLAUSE2}$: "O55222:P60766" and "O55222:P63001". Without using the coreference resolution module, these two pairs can never be retrieved. Unfortunately, because of the ambiguity of natural language in general, it is difficult to get full coreference resolution in practice. As a result, we restrict ourselves to a simple rule-based coreference resolution module that only resolves the Wh-pronominal coreference. Wh-pronouns refer to WHO, WHICH, THAT, WHAT and WHOSE. Detailed information about Wh-pronouns can be found in Santorini (1991).

Because the first word "which" in $S_{CLAUSE2}$ refers to "integrin-*O55222* signals", the coreference resolution module replaces "which" with "integrin-*O55222* signals". Hence, S_{MAIN} and $S_{CLAUSE1}$ remain unchanged and the clause $S_{CLAUSE2}$ becomes:

– $S_{CLAUSE2}$: integrin-*O55222* signals activate *P60766* and *P63001*, small Rho GTPases.

2.4.3 Candidate pair extraction

After preprocessing, we are ready to extract the candidate PPI pairs from the clause-based articles. First those clauses that do not contain any database ID or contain only 1 database ID are ignored. Then for each remaining clause, every two different database IDs are selected to compose one candidate PPI pair. If both database IDs occur only once in that clause, the choice of the candidate PPI pair is straightforward. However, in case one or both database IDs appear more than once in that clause, the nearest two database IDs are chosen to compose a candidate PPI pair.

Therefore, the input sentence S_{FEBS} is finally transformed into a set of *pair-based clauses* from which the SVM classifier will extract the features based on which it will predict whether the candidate PPI pairs are correct or not. For example, in $S_{CLAUSE1}$, the pair-based clauses are

– **Q9HBI1:Q8K4I3** *Q9HBI1* associates with PIX/ARHGEF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals

– **Q9HBI1:O55222** *Q9HBI1* associates with PIX/ARHGEF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals

– **Q8K4I3:O55222** *Q9HBI1* associates with PIX/ARHGEF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals

2.5 Feature extractor

After extracting the candidate pairs, a list of candidate PPI pair-based clauses "(P_1:P_2) C" is obtained where P_1 and P_2 are the protein names (i.e. UniProt identifiers) and C is the clause where P_1 and P_2 are extracted. Then based on these pair-based clauses we derive a set of features including surface features and advanced features to train the SVM model. All the features used in *PPIEor* are listed below and we use the pair-based clause

> S_1 "**Q9HBI1:Q8K4I3** *Q9HBI1* associates with PIX/ARHGEF6/Cool2 (*Q8K4I3*) at the tips of lamellipodia of motile cells and transmits integrin-*O55222* signals"

to illustrate the feature extraction procedure.

2.5.1 Surface features

Surface features are derived from the pair-based clauses describing the explicit properties of candidate PPI pairs and their interactions. In the following feature extraction procedures, all punctuation marks are ignored when counting tokens.

- $Feature_{pair}$: The candidate PPI pair "P_1:P_2". For example, $Feature_{pair}(S_1)$ = ("Q9HBI1: Q8K4I3") in S_1.
- $Feature_{P_1}$: P_1 and the two tokens before and after it. Then $Feature_{P_1}(S_1)$ = ("-", "-", "Q9HBI1", "associates", "with") where"-" stands for the empty token.
- $Feature_{P_2}$: P_2 and the two tokens before and after it. $Feature_{P_2}(S_1)$ = ("ARHGEF6", "Cool2", "Q8K4I3", "at", "the").
- $Feature_{iWord}$: Usually interaction words are good indicators for the occurrence of interactions. We construct a lexicon called *iLexicon* that consists of interaction nouns and verbs similar to the ones proposed by Plake et al. (2005). Then we refine and extend *iLexicon* based on the training data of $Data_{FEBS}$. Due to the spacial limitation, we cannot present *iLexicon* here but it can be found in the PhD thesis of Chen (2009). Declensions of these nouns and conjugations of these verbs are also accepted. Every token in the given clause is matched against the entries of *iLexicon* to find the corresponding interaction words. If several interaction words exist, the one nearest to the proteins consisting of candidate PPI pairs is chosen while if no interaction word exists the value of this feature is NULL. In S_1, only the interaction word "associate" is found. So, $Feature_{iWord}(S_1)$ = ("associate").
- $Feature_{Location}$: Sometimes the importance of a clause is related to the location of that clause in the article. For example, the candidate PPI pairs appearing in the TITLE or the ABSTRACT have a higher probability to be correct than those appearing somewhere else. Usually, authors only write the most essential information in the TITLE or the ABSTRACT. We consider the following 5 categories for the location of the clauses: TITLE, ABSTRACT, FIGURE, TABLE and BODY. As a consequence, $Feature_{Location}(S_1)$ = ("BODY").
- $Feature_{P2Pdistance}$: The distance in tokens between the two proteins P_1 and P_2. So, $Feature_{P2Pdistance}(S_1)$ = (5).
- $Feature_{NP}$: This feature is the number of other identified proteins between the two proteins. $Feature_{NP}(S_1)$ = (0).
- $Feature_{iWordLocation}$: The relative location of iWord within candidate PPI pairs, which is whether iWord appears between, before or after the pairs. In S_1, $Feature_{iWordLocation}(S_1)$ = ("between"). The value of $Feature_{iWordLocation}$ is set to NULL if no iWord exists.
- $Feature_{iWord2P1distance}$: The distance in tokens between the iWord and P1, e.g. from S_1, $Feature_{iWord2P1distance}(S_1)$ = (0). If no iWord exists, the value of $Feature_{iWord2P1distance}$ is NULL.
- $Feature_{iWord2P2distance}$: The distance in tokens between the iWord and P2. For example $Feature_{iWord2P2distance}(S_1)$ = (4). In the same way, the value of $Feature_{iWord2P2distance}$ is NULL if no iWord exists.

2.5.2 Advanced features

As discussed before, surface features only use basic information. In order to improve the performance further, we also incorporate some more advanced features.

	Pattern
1:	**P₁** $W_{1.1}$ **iVerb** $W_{1.2}$ **P₂**
2:	**P₁** $W_{2.1}$ **iVerb** $W_{2.2}$ by $W_{2.3}$ **P₂**
3:	**iVerb** of $W_{3.1}$ **P₁** $W_{3.2}$ by $W_{3.3}$ **P₂**
4:	**iVerb** of $W_{4.1}$ **P₁** $W_{4.2}$ to $W_{4.3}$ **P₂**
5:	**iNoun** of $W_{5.1}$ **P₁** $W_{5.2}$ (by|through) $W_{5.3}$ **P₂**
6:	**iNoun** of $W_{6.1}$ **P₁** $W_{6.2}$ (with|to|on) $W_{6.3}$ **P₂**
7:	**iNoun** between $W_{7.1}$ **P₁** $W_{7.2}$ and $W_{7.3}$ **P₂**
8:	complex between $W_{8.1}$ **P₁** $W_{8.2}$ and $W_{8.3}$ **P₂**
9:	complex of $W_{9.1}$ **P₁** $W_{9.2}$ and $W_{9.3}$ **P₂**
10:	**P₁** $W_{10.1}$ form $W_{10.2}$ complex with $W_{10.3}$ **P₂**
11:	**P₁** $W_{11.1}$ **P₂** $W_{11.2}$ **iNoun**
12:	**P₁** $W_{12.1}$ **P₁** $W_{12.2}$ **iVerb** $W_{12.3}$ with each other
13:	**P₁** $W_{13.1}$ **iVerb** $W_{13.2}$ but not $W_{13.3}$ **P₂**
14:	**P₁** $W_{14.1}$ cannot $W_{14.2}$ **iVerb** $W_{14.3}$ **P₂**
15:	**P₁** $W_{15.1}$ (do|be) not $W_{15.2}$ **iVerb** $W_{15.3}$ **P₂**
16:	**P₁** $W_{16.1}$ not $W_{16.2}$ **iVerb** $W_{16.3}$ by $W_{16.4}$ **P₂**

Table 1. A set of 16 patterns for $Feature_{Pattern}$. Pattern 1−12 indicate the interactions between candidate PPI pairs while Pattern 13−16 indicate that no interaction exists. $W_{i.j}$ means the i^{th} word gaps in Pattern j.

– *Pattern Matching Features ($Feature_{Pattern}$)*: Inspired by Plake et al. (2005), we designed a set of 16 syntactic patterns based on the training data. Each pattern is a syntactic description of sentence parts expressing protein locations, interaction nouns and verbs, and particular words. Two types of semantic information are integrated into these syntactic patterns, i.e. a protein-protein interaction exists or not. 12 patterns are designed to describe interactions between proteins and the remaining 4 patterns describe negations. Hence, in total 16 pattern matching features are designed: $Feature_{Pattern_1}$, $Feature_{Pattern_2}$, $\cdots$, $Feature_{Pattern_{16}}$. If a clause matches a pattern $Pattern_i$, the value of corresponding $Feature_{Pattern_i}$ is "1", otherwise it is "0". The 16 syntactic patterns are listed in Table 1 and they contain five different types of components:

  1. P_1 *and* P_2: P_1 and P_2 refer to the first and second proteins respectively in the PPI pair.
  2. *iNoun*: iNoun refers to the nouns indicating interactions taken from *iLexicon*.
  3. *iVerb*: iVerb refers to the verbs indicating interactions taken from *iLexicon*.
  4. *Fixed words*: Besides PPI pairs and iNouns/iVerbs, some patterns require that particular words occur in the clause. A pattern can require a fixed word like "by" in Pattern 2, or a word from a list, e.g. (with|to|on) in Pattern 6.
  5. *Word gaps*: Word gaps describe an optional sequence of words between the four components above. These gaps are limited in length but they do not require particular words. As recommend in Plake et al. (2005) we have set the maximum length of the gaps equal to 5.

– *Database Matching Features*: We match each candidate PPI pair with the entries of the protein interaction database used to see if this pair has already been recorded. Note that this feature will not be used by *PPIEor* until we have discussed the impact of the interaction databases in Section 3.5. For the moment, the most popular protein interaction databases are MINT and IntAct. Therefore we use the following two database matching features:

  1. $Feature_{MINT}$: Each candidate PPI pair is matched against all the entries in MINT. If matched, $Feature_{MINT}$ = 1 otherwise $Feature_{MINT}$ = 0. For instance, in S_1, the pair "Q9HBI1:Q8K4I3" can be found in MINT, hence $Feature_{MINT}(S_1)$ = (1).
  2. $Feature_{IntAct}$: Each candidate PPI pair is matched against all the entries in IntAct. If matched, $Feature_{IntAct}$ = 1 otherwise $Feature_{IntAct}$ = 0. For instance, in S_1, the pair "Q9HBI1:Q8K4I3" cannot be found in IntAct, hence $Feature_{IntAct}(S_1)$ = (0).

2.6 Feature selector

A feature selection method was used to select a subset of the most relevant features in order to build a robust machine learning model. By removing the most irrelevant and redundant features from the feature set, feature selection helps to improve the learning performance, to reduce the curse of dimensionality, to enhance the generalization ability, to accelerate the learning process and to boost the model interpretability.

The most straightforward method is subset selection with greedy forward search. This method is very simple to use but it has some drawbacks. It is more prone than other methods to get stuck in local optima and computationally it is very expensive (Saeys et al., 2007). Hence, for *PPIEor* we decided in favor of *SVM Recursive Feature Elimination (SVM RFE)* proposed by Guyon et al. (2002) to do the feature selection. SVM RFE interacts with the SVM classifier to search the optimal feature set and is less computationally intensive than the subset selection method. For a more detailed discussion about feature selection methods, the reader is referred to the review paper (Saeys et al., 2007).

In case of a linear kernel, SVM RFE uses the weights w_i appearing in the decision boundary to produce the feature ranks. The best subset of r features is the one that generates the largest margin between the two classes when the SVM classifier is using this subset. Stated in Guyon et al. (2002), the criteria $(w_i)^2$ estimates the effect on the objective function of removing one feature at a time. The feature with the smallest $(w_i)^2$ is removed first and as a result it has the lowest rank. In this way a corresponding feature ranking can be achieved. However, the features that are top ranked (eliminated last) are not necessarily the ones that are individually the most relevant. In some sense the features of a subset are optimal only when they are taken together. For computational reasons, it may be more efficient to remove several features at a time but at the expense of possible classification performance degradation.

In this chapter we use the toolbox *Java-ML* designed by Abeel et al. (2009) to implement the SVM RFE algorithm. Java-ML is a collection of machine learning and data mining algorithms and it has a usable and easily extensible API used by *PPIEor*. The library is written in Java and is available from http://java-ml.sourceforge.net/ under the GNU GPL license.

2.7 Classification model

After extracting features for the candidate PPI pair-based clauses $(P_1{:}P_2)C$, a binary classifier is needed to decide whether the candidate PPI pairs are correct or not. $Model_{SVM\_linear}$ is proposed using a linear kernel and the features described above. The toolbox LIBSVM (Chang & Lin, 2001) is used to train and tune $Model_{SVM\_linear}$ using 5-fold cross validation.

2.8 Post-processor

PPIEor makes an implicit assumption, i.e. the proteins in PPI pairs are different. This assumption leads to the problem that we cannot find self-interaction proteins. In the article DOI:10.1016/j.febslet.2008.12.036, the only correct PPI pair is "P64897:P64897", i.e. "P64897" interacts with itself. However, usually self-interactions are not stated explicitly in the articles. Therefore, we develop a post-processor to recover some self-interaction protein pairs and it consists of three steps:

- First, recall from Section 2.4.3 that the clauses that contain only 1 protein are ignored. Now we want to see if these proteins can interact with themselves. Therefore, for each article the proteins that do not consist of any candidate PPI pair are picked out.
- Second, for each protein obtained in the first step, search the MINT and IntAct databases to see if it can interact with itself.
- Finally, if the answer is yes, regard this protein as a self-interaction protein and add the corresponding pair to final PPI pair list.

Same as the database matching features discussed in Section 2.5.2, this component will not be used by *PPIEor* until the impact of the two databases on its performance is discussed in Section 3.5.

3. Results and discussion

3.1 Experimental purpose

Before discussing the experimental results, we would like to state the two purposes of *PPIEor*. The first purpose is to extract PPI pairs in the articles as accurately as possible to help the researchers avoid reading all the available articles. In this case, the performance of the system can be improved a lot by making use of interaction databases like MINT and IntAct. The second purpose is to help database curators who want to extract newly discovered PPI pairs from the articles that have not been recorded yet in databases. In this case it is not realistic to use existing interaction databases.

In the following we first focus on the second purpose, i.e. to build *PPIEor* without using any interaction database. First, in Section 3.2 we compare the fine-tuned *PPIEor* with other leading protein-protein interaction pair extraction systems built on similar data sets. Then in Sections 3.3 and 3.4 we show the impact of these components on *PPIEor* including the contribution of the preprocessor, the features and the feature selection method. Finally, in Section 3.5 we turn to the first purpose mentioned above and discuss the impacts of the databases, i.e. MINT and IntAct.

3.2 Results

PPIEor is developed and tuned on the training data of $Data_{FEBS}$ by doing 5-fold cross validation. After finding the optimal parameter value $C = 2^{-7}$ for the box constraint in the SVM the system is applied to the test data of $Data_{FEBS}$ and evaluated using the precision, the recall and the $F_{\beta=1}$ measure (Van Rijsbergen, 1979). The confidence intervals shown here are obtained by the *bootstrap resampling* method (Efron & Tibshirani, 1994) making use of 1,000 samples and for a confidence level of $\alpha = 0.05$.

The performance of *PPIEor* using $Model_{SVM\_linear}$ is compared with some of the leading protein-protein interaction extraction systems in Table 2. *PPIEor* is built by using the optimal feature set obtained by the SVM RFE feature selection method in Section 2.6. All systems

	Precision	Recall	$F_{\beta=1}$
PPIEor ($Model_{SVM\_linear}$)	72.66%	75.61%	74.10 ± 2.11
Syntax Pattern-based System	60.00%	46.00%	52.00
MDL-based System	79.80%	59.50%	68.17

Table 2. Evaluation results of *PPIEor* compared with other systems.

are evaluated on similar data sets, i.e. biological literature annotated with golden standard protein names.
Since we cannot reproduce the other systems, we compare the performance of *PPIEor* on the test set of $Data_{FEBS}$ with the ones reported in the literature by these competitors.
The *Syntax Pattern-based System* proposed by Plake et al. (2005) matched sentences against syntax patterns describing typical protein interactions. The syntax pattern set was refined and optimized on the training set using a genetic algorithm. This system was evaluated on the corpus of the BioCreAtIvE I challenge, Task 1A (Yeh et al., 2005) and got a $F_{\beta=1}$ measure of 52.00.
Another leading system, *MDL-based System*, proposed by Hao et al. (2005) used a minimum description length (MDL)-based pattern-optimization algorithm to extract protein-protein interactions and used a manually selected corpus from biological literature consisting of 963 sentences. This system got a $F_{\beta=1}$ measure of 68.17.
From Table 2, it can be seen that *PPIEor* using $Model_{SVM\_linear}$ gets a comparable $F_{\beta=1}$ measure with the above two leading systems, which is 74.10 ± 2.11. Therefore we can conclude that *PPIEor*'s performance is quite promising.

3.3 Contribution of preprocessor

First, we discuss the contribution of the preprocessor. It transforms the original sentences into a clause-based representation consisting of main sentences and a number of clauses followed by a coreference resolution module that resolves the Wh-pronominal coreference in order to facilitate the extraction of the candidate PPI pairs. Table 3 shows the performance of *PPIEor* without and with preprocessor. In the former case, the original sentences themselves are used as input data. It can be seen that with the preprocessor *PPIEor* performs much better, i.e the precision is increased by 4.08, the recall by 1.62 and the $F_{\beta=1}$ measure by 2.89. And the difference of the $F_{\beta=1}$ measures is significant for a confidence level $\alpha = 0.05$.
Another advantage of the preprocessor is that less candidate PPI pairs are extracted especially negative ones, which is illustrated in Example 3.1.

Example 3.1 *Consider that the sentence S consists of 2 clauses, C_1 and C_2. In C_1 2 proteins P_1 and P_2 are recognized and in C_2 also 2 proteins P_3 and P_4 are recognized. Only the PPI pair "P_1:P_2" are correct.*

$$S_1 : \underbrace{\cdots P_1 \cdots P_2 \cdots}_{C_1}, \underbrace{\cdots P_3 \cdots P_4 \cdots}_{C_2}.$$

With using the preprocessor, only 2 candidate PPI pairs are extracted: 1 positive PPI pair "P_1:P_2" and 1 negative PPI pair "P_3:P_4". However, without using the preprocessor, 6 candidate PPI pairs are extracted: 1 positive PPI pair "P_1:P_2" and 5 negative PPI pairs: "P_1:P_3", "P_1:P_4", "P_2:P_3", "P_2:P_4" and "P_3:P_4".

From Example 3.1, we see that on one hand, the preprocessor can handle the unbalance in the distribution of candidate positive and negative pairs to some extent and hence avoid the problems caused by such unbalance when building machine learning based models. On the

Data set	Precision	Recall	$F_{\beta=1}$
The sentence-based data set	50.51%	80.49%	62.69 ± 1.27
The clause-based data set	54.59%	82.11%	65.58 ± 1.31

Table 3. Comparison of the performance of *PPIEor* without (the data set consists of sentences) and with (the data set consists of clauses) preprocessor.

other hand, *PPIEor* becomes more efficient since less candidate PPI pairs has to be considered. Hence, it is better to use the preprocessor as a component of *PPIEor*.

3.4 Contribution of feature selection

The SVM Recursive Feature Elimination (SVM RFE) algorithm is designed specifically for SVMs and hence this feature selection method interacts directly with the SVM model. Since the core component of *PPIEor* is the SVM-based binary classifier $Model_{SVM\_linear}$, we think that the SVM RFE algorithm is to be prefered over other feature selection techniques like χ^2 (White & Liu, 1994), information gain (Quinlan, 1986) and gain ratio (Quinlan, 1993) which ignore interactions with the classifier. Hence the SVM RFE algorithm is applied to the original feature set (except the two database matching features $Feature_{MINT}$ and $Feature_{IntAct}$) discussed in Section 2.5 to select the optimal subset of features.
However, it is important to note that based on the specifications of the PPIE task, the performance of $Model_{SVM\_linear}$ is not exactly same as the performance of the final *PPIEor*. We will illustrate this in Example 3.2.

Example 3.2 *Consider the snippet of the input instances from the article DOI: 10.1016/j.febslet.200 8.01.064 shown below. The first item is the class label and the rest are the extracted features.*
1|O55222:Q9HBI1|DISTANCE:CLOSE|P2P:0|NULL|...
1|Q9ES28:Q9HBI1|DISTANCE:CLOSE|P2P:0|NULL|...
1|O55222:Q9HBI1|DISTANCE:CLOSE|P2P:0|LOC:BETWEEN|...
1|O55222:Q9HBI1|DISTANCE:MIDDLE|P2P:0|LOC:RIGHT|...

It is clear that there are two different candidate PPI pairs "O55222:Q9HBI1" and "Q9ES28:Q9HBI1". Because the candidate PPI pair "O55222:Q9HBI1" is discussed many times in the article DOI: 10.1016/j.febslet.2008.01.064, e.g. it appears in the figure, the title and the abstract, three instances are created, i.e the first, third and fourth instance above.

First, we use Example 3.2 to see the performance changes in $Model_{SVM\_linear}$:

- *Step 1*: $Model_{SVM\_linear}$ classifies all these 4 instances correctly, the recall is $4/(3+1) \times 100\% = 100\%$.
- *Step 2*: If $Model_{SVM\_linear}$ classifies the first instance incorrectly but the other three ones correctly, this gives 3 true positives and 1 false negative and hence the recall is $3/(3+1) \times 100\% = 75\%$.
- *Step 3*: If $Model_{SVM\_linear}$ classifies the first and second instances incorrectly but the other two ones correctly, the recall is $2/(3+1) \times 100\% = 50\%$.
- *Step 4*: If $Model_{SVM\_linear}$ only classifies correctly the fourth instance, the recall is $1/(3+1) \times 100\% = 25\%$.
- *Step 5*: If $Model_{SVM\_linear}$ misclassifies all the instances, this gives a recall of $0/(3+1) \times 100\% = 0\%$.

Hence it can be seen that as the number of misclassifications increases, the performance of $Model_{SVM\_linear}$ decreases.
However, the purpose of *PPIEor* is to find *distinct* PPI pairs in each article. For example, in the article DOI: 10.1016/j .febslet.2008.01.064, the correct PPI pairs are "O55222:Q9HBI1" and "Q9ES28:Q9HBI1". Again using Example 3.2, the performance of *PPIEor* changes as follows:

- *Step 1*: All these 4 instances are classified correctly, the recall is $2/2 \times 100\% = 100\%$.
- *Step 2*: If the first instance is misclassified but the other three ones are classified correctly, the recall is still $2/2 \times 100\% = 100\%$.
- *Step 3*: If the first and second instances are classified incorrectly but the other two ones correctly, the recall becomes $1/2 \times 100\% = 50\%$.
- *Step 4*: If only the fourth instance is classified correctly, the recall is $1/2 \times 100\% = 50\%$.
- *Step 5*: If all the instances are misclassified, this gives a recall of $0/2 \times 100\% = 0\%$.

As in the case of $Model_{SVM\_linear}$, as the number of misclassifications increases, the performance of *PPIEor* also decreases but differences are not the same. Therefore, we can conclude that the performances of $Model_{SVM\_linear}$ and *PPIEor* are *different* but *closely related*.
Since the performance of *PPIEor* is our final purpose, we decide to tune the feature selection based on the $F_{\beta=1}$ measure of *PPIEor*. Fig. 2 shows the contribution of the SVM RFE algorithm to the performances of both *PPIEor* and $Model_{SVM\_linear}$.
As explained above, $Model_{SVM\_linear}$ and *PPIEor* perform differently. In Fig. 2 one can see that their best $F_{\beta=1}$ measures are achieved for a different number of highest ranked features.
However, it can also be seen that the performances of $Model_{SVM\_linear}$ and *PPIEor* are closely related. Using the SVM RFE algorithm to rank the features by interacting with $Model_{SVM\_linear}$ also imposes the positive effect on the performance of *PPIEor*. The $F_{\beta=1}$ measure of *PPIEor* with all the features is 65.58 ± 1.31. When the top ranked 143 features (4.04%) are used, which are obtained by 5-fold cross validation on the training data, *PPIEor* achieves a $F_{\beta=1}$ measure of 74.10 ± 2.11. After applying the SVM RFE algorithm, the $F_{\beta=1}$ measure of *PPIEor* is increased by 8.52. This is significantly different for a confidence level $\alpha = 0.05$.
Finally, we look at types of the 143 best features. In Table 4, the types of the designed features are listed in a descending order according to their relative importance for *PPIEor*. Here importance means the more features are selected from a certain feature type, the more important that type is. It can be seen that the most important feature types are $Feature_{pair}$ and $Feature_{Pattern}$ with 122 and 9 features among the 143 best ones. In contrast, $Feature_{iWord}$, $Feature_{Location}$ and $Feature_{iWord2P1distance}$ are not important since no features of this type are selected.

3.5 Impact of the interaction databases

In this section we turn to the first purpose of *PPIEor*, i.e. to extract the PPI pairs in the articles as accurately as possible to help the researchers avoid reading all the available articles. For this situation, the interaction databases used by the database matching features and the post-processor give the positive contribution.
First, the two interaction databases, MINT and IntAct, are used to extract the database matching features, $Feature_{MINT}$ and $Feature_{IntAct}$. Second, based on MINT and IntAct, the post-processor will recover some self-interaction pairs. Table 5 shows the comparison of the performance of *PPIEor* without the interaction databases, with the database matching features and with the post-processor.

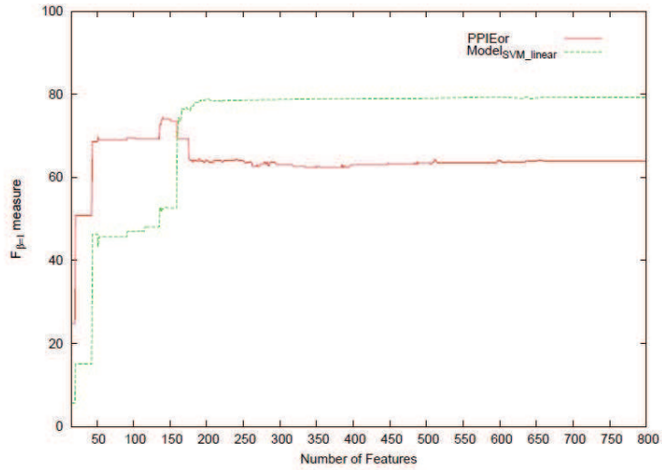

Fig. 2. Contributions of the SVM Recursive Feature Elimination (SVM RFE) algorithm to the performances of both *PPIEor* and $Model_{SVM\_linear}$. Note that x-axis is restricted to the range from 0 to 800 features since the performances do not change anymore when more features are added. The SVM RFE algorithm ranks the total of 3,543 features in a descending order according to their weights. The $F_{\beta=1}$ measure of *PPIEor* with all the features is 65.58 ± 1.31. When the optimal parameter values obtained by 5-fold cross validation on the training data are used, *PPIEor* achieves the $F_{\beta=1}$ measure of 74.10 ± 2.11 when the top 143 features (4.04%) are used. For $Model_{SVM\_linear}$, the best $F_{\beta=1}$ measure of 79.35 is obtained when the top 635 features are used while the $F_{\beta=1}$ measure for all the features is 79.17.

From the results shown in Table 5, it can be seen that the database matching features can greatly improve the performance. They increase the $F_{\beta=1}$ measures from 74.10 to 88.99. This makes sense because the PPI pairs that are recorded in MINT and IntAct have been verified by biological experiments and hence the matched candidate pairs have higher probabilities to be correct PPI pairs. The post-processor which has to recover some self-interaction pairs also contributes to the performance of *PPIEor*. It is clear that with post-processor the recall and the $F_{\beta=1}$ measure increase significantly although the precision drops slightly. Therefore we

Feature Type	Number of Selected Features after the feature selection
$Feature_{pair}$	122/290 (42.07%)
$Feature_{Pattern}$	9/37 (24.32%)
$Feature_{NP}$	3/6 (50.00%)
$Feature_{P1}$	3/1532 (0.20%)
$Feature_{P2Pdistance}$	2/3 (66.67%)
$Feature_{P2}$	2/1566 (0.13%)
$Feature_{iWordLocation}$	1/4 (25.00%)
$Feature_{iWord2P2distance}$	1/4 (25.00%)
$Feature_{iWord}$	0/92 (0.00%)
$Feature_{Location}$	0/5 (0.00%)
$Feature_{iWord2P1distance}$	0/4 (0.00%)

Table 4. Importance of the different types of features in descending order according to the number of selected features.

	Precision	Recall	$F_{\beta=1}$
Without Using Databases	72.66%	75.61%	74.10 ± 2.11
+Database Matching Features	97.12%	82.11%	88.99 ± 0.81
+Post-Processor	96.36%	86.18%	90.99 ± 0.81

Table 5. Contribution of the interaction databases to *PPIEor*.

conclude that the post-processor reduces to some extent the self-interaction problem.
However it should be noted that using the interaction databases makes *PPIEor* hardly capture newly discovered PPI pairs that are not recorded in the databases yet. Hence it is better not to use the interaction databases when searching for new PPI pairs.

4. Conclusion

In this chapter we presented a protein-protein interaction pair extractor (*PPIEor*), which used a binary SVM classifier as the core component. Its purpose was to automatically extract protein-protein interaction pairs from biological literature. During the preprocessing phase, the original sentences from the articles were transformed into clause-based ones and the candidate PPI pairs were distilled. Then we derived a rich and informative set of features including surface features and advanced features. In order to improve the performance further, we used a feature selection method, the SVM Recursive Feature Elimination (SVM RFE) algorithm, to find the features most relevant for classification. Finally, the post-processor recovered some of the self-interaction proteins which could not be identified by our SVM model. The experimental results has proved that *PPIEor* can achieve the quite promising performance.
However, PPI pairs that appear in the figures, span different sentences or interact with themselves cannot be handled well for the moment. More advanced techniques need to be exploited in the future, like anaphora resolution used for semantic analysis to detect the inter-sentence PPI pairs, or specifically designed patterns to recover more self-interaction PPI pairs, etc.

5. References

Abeel T.; Van de Peer Y. & Saeys Y. (2009). Java-ML: A Machine Learning Library. *Journal of Machine Learning Research*, Vol.10, 931-934

Airola A.; Pyysalo S.; Björne J.; Pahikkala T.; Ginter F. & Salakoski T. (2008). A Graph Kernel for Protein-Protein Interaction Extraction, *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing*, pp. 1-9

Ananiadou S. & McNaught J. (2006). *Text Mining for Biology and Biomedicine*, Artech House, Inc., ISBN:158053984X, London.

Baumgartner Jr. W.; Lu Z.; Johnson H.; Caporaso J.; Paquette J.; Lindemann A.; White E.; Medvedeva O.; Cohen K. & Hunter L. (2007). An integrated approach to concept recognition in biomedical text, *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pp. 257-271

Bunescu R. & Mooney R. (2005). Subsequence kernels for relation extraction, *Proceedings of the 19th Conference on Neural Information Processing Systems*, pp. 171-178

Ceol A.; Chatr-Aryamontri A.; Licata L. & Cesareni G. (2008). Linking entries in protein interaction database to structured text: the FEBS Letters experiment. *FEBS Letters*, Vol.582, No.8, 1171-1177

Chang C. & Lin C. (2001). LIBSVM : a library for support vector machines.

Charniak E. & Johnson M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking, *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 173-180

Chen Y. (2009). *Biological Literature Miner: Gene Mention Recognition and Protein-Protein Interaction Pair Extraction*, Vrije Universiteit Brussel.

Craven M. (1999). Learning to extract relations from MEDLINE, *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, pp. 25-30

De Bruijn B. & Martin J. (2002). Literature mining in molecular biology, *Proceedings of the EFMI Workshop Natural Language*, pp. 1-5

Efron B. & Tibshirani R. (1994). *An Introduction to the Bootstrap*, Chapman & Hall/CRC.

Ejerbed E. (1988). Finding Clauses In Unrestricted Text By Finitary And Stochastic Methods, *Proceedings of the second conference on Applied Natural Language Processing*, pp. 219-227

Grover C.; Haddow B.; Klein E.; Matthews M.; Neilsen L.; Tobin R. & Wang X. (2007). Adapting a Relation Extraction Pipeline for the BioCreAtIvE II Tasks, *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pp. 273-286

Guyon I.; Weston J.; Barnhill S. & Vapnik V. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, Vol.46, No.1-3, 389-422

Hao Y.; Zhu X.; Huang M. & Li M. (2005). Discovering patterns to extract protein-protein interactions from the literature: Part II. *Bioinformatics*, Vol.21, No.15, 3294-3300

Hakenberg J.; Plake C.; Royer L.; Strobelt H.; Leser U. & Schroeder M. (2008). Gene mention normalization and interaction extraction with context models and sentence motifs, *Genome Biology*, Volume 9, Suppl 2, Article S14

Krallinger M.; Leitner F. & Valencia A. (2007). Assessment of the second BioCreative PPI task: Automatic Extraction of Protein-Protein Interactions, *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pp.41-54

Plake C.; Hakenberg J. & Leser U. (2005). Optimizing syntax patterns for discovering protein-protein interactions, *Proceedings of the 2005 ACM symposium on Applied computing*, pp. 195-201

Quinlan J. (1986). Induction of decision trees. *Machine Learning*, Vol.1, No.1, 81-106

Quinlan J. (1993). *C4.5: Programs for Machine Learning*, Morgan Kaufmann.

Ray S. & Craven M. (2001). Representing Sentence Structure in Hidden Markov Models for Information Extraction, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pp. 1273-1279

Saeys Y.; Inza I. & Larrañaga P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, Vol.23, No.19, 2507-2517

Santorini B. (1991). *Part-of-Speech Tagging Guidelines for the Penn Treebank Project*, Department of Computer and Information Science, University of Pennsylvania.

Van Rijsbergen C. (1979). *Information Retrieval*, Butterworth-Heinemann.

White A. & Liu W. (1994). Bias in Information-based measures in decision tree induction. *Machine Learning*, Vol.15, No.3, 321-329

Yeh A.; Morgan A.; Colosimo M. & Hirschman L. (2005). BioCreAtIvE Task 1A: gene mention finding evaluation. *BMC Bioinformatics*, Vol.6(Suppl 1), S2

Zelenko D.; Aone C. & Richardella A. (2003). Kernel methods for relation extraction. *The Journal of Machine Learning Research*, Vol.3, 1083-1106

Biomedical Engineering, Trends, Research and Technologies
Edited by Dr. Sylwia Olsztynska

ISBN 978-953-307-514-3
Hard cover, 644 pages
Publisher InTech
Published online 08, January, 2011
Published in print edition January, 2011

This book is addressed to scientists and professionals working in the wide area of biomedical engineering, from biochemistry and pharmacy to medicine and clinical engineering. The panorama of problems presented in this volume may be of special interest for young scientists, looking for innovative technologies and new trends in biomedical engineering.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen