26

Protein-Protein Interactions Extraction from Biomedical Literatures

Hongfei Lin, Zhihao Yang and Yanpeng Li
Dalian University of Technology
China

1. Introduction

Protein-protein interactions (PPI) play a key role in various aspects of the structural and functional organization of the cell. Knowledge about them unveils the molecular mechanisms of biological processes. A number of databases such as MINT (Zanzoni et al., 2002), BIND (Bader et al., 2003), and DIP (Xenarios et al., 2002) have been created to store protein interaction information in structured and standard formats. However, the amount of biomedical literature regarding protein interactions is increasing rapidly and it is difficult for interaction database curators to detect and curate protein interaction information manually. Thus, most of the protein interaction information remains hidden in the text of the papers in the literature. Therefore, automatic extraction of protein interaction information from biomedical literature has become an important research area.

Existing PPI works can be roughly divided into three categories: Manual pattern engineering approaches, Grammar engineering approaches and Machine learning approaches.

Manual pattern engineering approaches define a set of rules for possible textual relationships, called patterns, which encode similar structures in expressing relationships. The SUISEKI system uses regular expressions, with probabilities that reflect the experimental accuracy of each pattern to extract interactions into predefined frame structures (Blaschke & Valencia, 2002). Ono et al. manually defined a set of rules based on syntactic features to preprocess complex sentences, with negation structures considered as well (Ono et al., 2001). The BioRAT system uses manually engineered templates that combine lexical and semantic information to identify protein interactions (Corney et al., 2004). Such manual pattern engineering approaches for information extraction are very hard to scale up to large document collections since they require labor-intensive and skill-dependent pattern engineering.

Grammar engineering approaches use manually generated specialized grammar rules that perform a deep parse of the sentences. Sekimizu et al. used shallow parser, EngCG, to generate syntactic, morphological, and boundary tags (Sekimizu et al., 1998). Based on the tagging results, subjects and objects were recognized for the most frequently used verbs. Fundel et al. proposed RelEx based on the dependency parse trees to extract relations (Fundel et al., 2007).

Machine learning techniques for extracting protein interaction information have gained interest in the recent years. In most recent work on machine learning for PPI extraction, the PPI extraction task is casted as learning a decision function that determines for each

unordered candidate pair of protein names occurring together in a sentence whether the two proteins interact or not. Xiao et al. used Maximum Entropy models to combine diverse lexical, syntactic and semantic features for PPI extraction (Xiao et al., 2005). Zhou et al. employed a semantic parser using the Hidden Vector State (HVS) model for protein-protein interactions which can be trained using only lightly annotated data whilst simultaneously retaining sufficient ability to capture the hierarchical structure (Zhou et al., 2006). Yang et al. used Support vector machines to combine rich feature sets including word features, Keyword feature, protein names distance feature, Link path feature and Link Grammar extraction result feature to identify protein interactions (Yang et al., 2010).

A wide range of results have been reported for the PPI extraction systems, but differences in evaluation resources, metrics and strategies make direct comparison of the numbers presented problematic (Airola et al., 2008). Further, PPI extraction methods generate poorer results compared with other domains such as newswire. In general, biomedical IE methods are scored with F-measure, with the best methods scoring about 0.85 without considering the limitation of test corpus, which is still far from users' satisfaction.

This chapter introduces three different protein-protein interactions extraction approaches which represent the state-of-the-art research in this area.

2. Methods

2.1 Multiple kernels learning mehtod

Among machine learning approaches, kernel-based methods (Cristianini & Taylor, 2000) have been proposed for PPI information extraction. Kernel-based methods retain the original representation of objects and use the object only via computing a kernel function between a pair of objects. Formally, a kernel function is a mapping K: X ×X $\rightarrow [0,\infty)$ from input space X to a similarity score $K(x,y) = \phi(x) \cdot \phi(y) = \Sigma_i \phi_i(x) \phi_i(y)$, where $\phi_i(x)$ is a function that maps X to a higher dimensional space without the need to know its explicit representation. Such a kernel function makes it possible to compute the similarity between objects without enumerating all the features.

Several kernels have been proposed, including subsequence kernels (Bunescu & Mooney, 2006), tree kernels (Moschitti, 2006), shortest path kernels (Bunescu & Mooney, 2005a), and graph kernels (Airola et al., 2008). Each kernel utilizes a portion of the structures to calculate useful similarity. The kernel cannot retrieve the other important information that may be retrieved by other kernels.

In recent years researches have proposed the use of multiple kernels to retrieve the widest range of important information in a given sentence. Kim et al. suggested four kernels: predicate kernel, walk kernel, dependency kernel and hybrid kernel to adequately encapsulate information required for a relation prediction based on the sentential structures involved in two entities (Kim et al., 2008). Miwa et al. proposed a method to combine BOW kernel, subset tree kernel and graph kernel based on several syntactic parsers, in order to retrieve the widest possible range of important information from a given sentence (Miwa et al., 2009).

However, these methods assign the same weight to each individual kernel and their combined kernels fail to achieve the best performance: in Kim's method, the performance of the hybrid kernel is worse than that of one of the individual kernels - the walk kernel. In Miwa's method, graph kernels outperform the other individual kernels. When combined with the subset tree kernels, it achieves better performance. However, when further

combined with BOW kernels, the performance deteriorates. In fact, the performance of BOW kernel and graph kernels combination is worse than that of graph kernels alone.
In this chapter, we propose a weighted multiple kernels learning based approach to extracting protein-protein interactions from biomedical literature. The approach combines feature-based kernel, tree kernel, and graph kernel with different weights: the kernel with better performance is assigned higher weight. Experimental results show the introduction of each individual kernel contributes to the performance improvement. The other novelties of our approach include: a) in addition to the commonly used word feature, our feature-based kernel includes the protein name distance feature as well as the Keyword feature. Especially, the introduction of Keyword feature is a way of employing domain knowledge and proves to be able to improve the performance effectively. b) with our tree kernel, we extend Shortest Path-enclosed Tree and dependency path tree to capture richer contextual information.

2.1.1 Methods

A kernel can be thought of as a similarity function for pairs of objects. Different kernels calculate the similarity with different aspects between two sentences. Combining the similarities can reduce the danger of missing important features and produce a new useful similarity measure. In this work, we combine several distinctive types of kernels to extract PPI: feature-based kernel, tree kernel, graph kernel.

2.1.1.1 Feature-based kernel

The following features are used in our feature-based kernel:

Word feature

A bag-of-words kernel takes two unordered sets of words as feature vectors, and calculates their similarity, which is simple and efficient. There are two sets of word features used in our method.
Words between two protein names: These features include all words that are located between two protein names.
Words surrounding two protein names: These features include N words to the left of the first protein name and N words to the right of the second protein name. N is the number of surrounding words considered which is set to be three in our experiment.

Protein name distance feature

The shorter the distance (the number of words) between two protein names is, the more likely the two proteins have interaction relation. Therefore the distance is chosen as a feature. If there are less than three words between two proteins, the feature value is set to "DISLessThanThree"; if there are more than or equal to three words but less than six words between two proteins, the feature value is set to "DISBetweenThreeSix". The other feature values include "DISBetweenSixNine", "DISBetweenNineTwelve" and "DISMoreThanTwelve".

Keyword feature

The existence of an interaction keyword (the verb expressing protein interaction relation such as "bind", "interact", "inhibit", etc) between two protein names or among the surrounding words of two protein names often implies the existence of the protein-protein interaction. Therefore, the existence of the keyword is chosen as a binary feature. To identify the keywords in texts, we built an interaction keyword list of about 500 entries manually, which includes the interaction verbs and their variants (for example, interaction verb "bind" has variants "binding" and "bound", etc. The list can be provided upon request).

2.1.1.2 Tree kernel

A convolution kernel aims to capture structured information in terms of substructures. As a specialized convolution kernel, convolution tree kernel K_C (T_1, T_2) counts the number of common sub-trees (sub-structures) as the syntactic structure similarity between two parse trees T_1 and T_2 (Collins & Duffy, 2001):

$$K_C(T_1, T_2) = \sum_{n_1 \in N_1, n_2 \in N_2} \Delta(n_1, n_2) \quad (1)$$

where N_j is the set of nodes in tree T_j, and $\Delta(n_1,n_2)$ evaluates the common sub-trees rooted at n_1 and n_2.

Parse tree kernel

A relation instance between two entities is encapsulated by a parse tree. Thus, it is critical to understand which portion of a parse tree is important in the tree kernel calculation.
Zhang et al. explored five tree spans in relation extraction and found that the Shortest Path-enclosed Tree (SPT, an example is shown in Figure 1) performed best (Zhang et al., 2006). SPT is the smallest common sub-tree including the two entities. In other words, the sub-tree is enclosed by the shortest path linking the two entities in the parse tree. But in some cases, the information contained in SPT is not enough to determine two entities' relationship. For example, "interact" is critical to determine the relationship between "ENTITY1" and "ENTITY2" in the sentence "ENTITY1 and ENTITY2 interact with each other." as shown in Figure 1. However, it is not contained in the SPT (dotted circle in Figure 1) to determine their relationship. By analyzing the experimental data, we found in these cases the number of leaf nodes in a SPT is usually less than four, following the pattern like "ENTITY1 and ENTITY2" and including little information except the two entity names.
Here we employ a simple heuristic rule to expand the SPT span. By default, we adopt SPT as our tree span. When the number of leaf nodes in a SPT is less than four, the SPT is expanded to a higher level, i.e. the parent node of the root node of the original SPT is used as the new root node. Thus the new SPT (solid circle in Figure 1) will include richer context information comprising the original SPT. In the above example, the flat SPT string is extended from "(NP (NN PROTEIN1) (CC and) (NN PROTEIN2))" to "(S (NP (NN PROTEIN1) (CC and) (NN PROTEIN2)) (VP (VBP interact) (PP ((IN with) (NP (DT each) (JJ other))))))" and includes richer context information.

Dependency path tree kernel

The other type of tree structure information included in our tree kernel is from parser dependency analysis output. For dependency based parse representations, a dependency path is encoded as a flat tree as depicted as follows: (DEPENDENCY (NSUBJ (interacts ENTITY1)) (PREP (interacts with)) (POBJ (with ENTITY2))) corresponding to the sentence "ENTITY1 interacts with ENTITY2". Because a tree kernel measures the similarity of trees by counting common subtrees, it is expected that the system finds effective subsequences of dependency paths.
Similar to SPT parse tree, in some cases, dependency path tree also needs extension. Taking the sentence "The expression of rsfA is under the control of both ENTITY1 and ENTITY2." as an example (its dependency parse is shown in Figure 2), the path tree between ENTITY1 and ENTITY2 is "(DEPENDENCY (CONJ (ENTITY1, ENTITY2))." Obviously, the information in this path tree is insufficient to determine the relationship between the two entitles. Our solution is to extend the length of dependency path between two proteins to

three when it is less than three. In such case, if there exist two edges in the left of the first protein in the whole dependency parse path, they will be included into the dependency path. Otherwise, the right two edges of the second protein will be included into the dependency path. In the above example, the path tree between ENTITY1 and ENTITY2 is extended from "(DEPENDENCY (CONJ (ENTITY1, ENTITY2))" to "(DEPENDENCY (PREP(control, of)) POBJ((of, ENTITY1)) (CONJ(ENTITY1, ENTITY2)))". The example is shown in Figure 2. The optimal extension threshold three is determined through experiments to achieve the best performance.

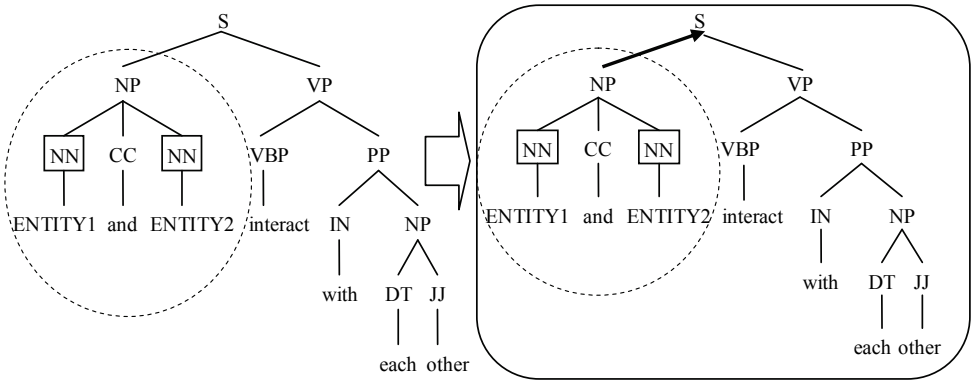

Fig. 1. An example of the extension of Shortest Path-enclosed Tree (the original SPT is in dotted circle and extended SPT in solid circle.)

2.1.1.3 Graph kernel

A graph kernel calculates the similarity between two input graphs by comparing the relations between common vertices (nodes). The graph kernel used in our method is the all-paths graph kernel proposed by Airola et al. (Airola et al., 2008). The kernel represents the target pair using graph matrices based on two subgraphs, and the graph features are all the non-zero elements in the graph matrices. The two subgraphs are a parse structure subgraph (PSS) and a linear order subgraph (LOS). More complete detail about the all-paths graph kernel is presented in (Airola et al., 2008).

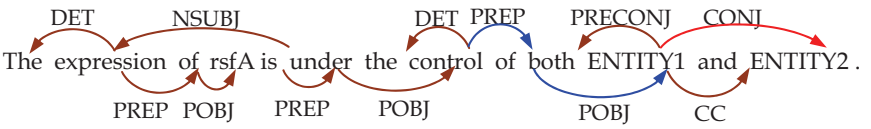

Fig. 2. An example of dependency path tree extension. The edge marked with red color is the original dependency path and the edge marked with blue color is included into the new dependency path.

2.1.1.4 Combination of kernels

Each kernel has its own advantages and disadvantages. The dependency path kernel ignores some deep information, and conversely, the parse tree kernel does not output certain shallow relations. All of them ignore the words. The feature-based kernel is simple and efficient, but can not capture the sentence structure. The graph kernels can treat the parser's

output and word features at the same time. However, they cannot treat them properly without tuning the kernel parameters. They may also miss some distant words, and similarities of paths among more than three elements (Airola et al., 2008).
The kernels calculate the similarity with different aspects between the two sentences. Combining the similarities can reduce the danger of missing important features and produce a new useful similarity measure. To realize the combination of the different types of kernels based on different parse structures, we sum up the normalized output of several kernels K_m as:

$$K(x,x') = \sum_{m=1}^{M} \sigma_m K_m(x,x') \tag{2}$$

$$\sum_{m=1}^{M} \sigma_m = 1,\ \sigma_m \geq 0,\ \forall m \tag{3}$$

where M represents the number of types of kernels, σ_m is the weight of each K_m which is determined through experiments: we tune the weight for each kernel until the overall best results are achieved. We found that each kernel has different performance and only when the kernel with better performance is assigned higher weight can the combination of each individual kernel produce the best result. In our experiments, the weights for feature-based kernel, tree kernel, and graph kernel are set to 0.6, 0.2 and 0.2 respectively in the order of performance rank (the weights of each individual kernel in combined kernels are shown in Table 5). This is a very simple combination, but the resulting kernel function contains all of the kernels' information. Comparatively, the methods in (Kim et al., 2008; Miwa et al., 2009) assign the same weight to each individual kernel and their combined kernels fail to achieve the best performance.

2.1.2 Experiments

2.1.2.1 Experimental setting

We evaluate method using a publicly available corpora AImed (Bunescu et al., 2005b) which is sufficiently large for training and reliably testing machine learning methods. It has recently been applied in numerous evaluations (Airola et al., 2008) and can be seen as an emerging de facto standard for PPI extraction method evaluation. Further, like in (Airola et al., 2008), we do not consider self-interactions as candidates and remove them from the corpora prior to evaluation. In our implementation, we use the SVMLight package (http://svmlight.joachims.org/) developed by Joachims for our feature-based kernel. The polynomial kernel is chosen with parameter d = 4. Tree Kernel Toolkits developed by Moschitti is used for our tree kernel (http://dit.unitn.it/~moschitt/Tree-Kernel.htm) and the default parameters are used. All-paths graph kernel proposed by Airola et al. (http://mars.cs.utu.fi/PPICorpora/GraphKernel.html) is used for our graph kernel.
In the test we evaluate our method with 10-fold document-level cross-validation so that no two examples from the same document end up in different cross-validation folds.

2.1.2.2 Experimental results and discussion

In this section, we firstly discuss the effectiveness of different features used in the feature-based kernel, SPT and dependency tree and their extensions, and different kernels on

AImed corpus. Here AImed is used since it is sufficiently large for training and reliably testing machine learning methods. It has recently been applied in numerous evaluations (Moschitti, 2006) and can be seen as an emerging de facto standard for PPI extraction method evaluation. Then we provide a comprehensive evaluation of our method across five PPI corpora, and compare our results with earlier work.

Effectiveness of different features in the feature-based kernel

In our method, no feature selection is performed. We tried stemming, but found little decline in performance. The classification performances of different features in the feature-based kernel tested on AImed are shown in Table 1 (the precision, recall, F-score values are achieved with the optimal threshold values obtained from the 10-fold cross-validations).
With the feature-based kernel, an F-score of 50.82% and an AUC of 77.69% are achieved using only word features. With the introduction of protein names distance and Keyword feature the F-score and AUC are improved to 52.69% and 80.71% respectively. Compared with protein names distance feature, the Keyword feature contributes more to the performance improvement (1.39 percentage units' increase in F-score and 2.48 percentage units' increase in AUC). The reason is that the Keyword feature employs domain knowledge, which proves to be able to improve the performance effectively. Exploiting domain knowledge may be a promising method to further improve PPI extraction performance. Similar works have been reported recently. Danger et al. defined a PPI ontology, PPIO, and showed some preliminary results guided by the ontology (Danger et al., 2008). He et al. proposed a novel framework of incorporating protein-protein interactions ontology knowledge into PPI extraction from biomedical literature in order to address the emerging challenges of deep natural language understanding (He et al., 2008).

Effectiveness of SPT parse tree, dependency tree and their extensions

The performances of SPT parse tree, dependency path tree and their extensions tested on AImed are shown in Table 2. Using only SPT achieves an F-score of 50.13% and an AUC of 76.32% while, after the introduction of SPT extension, dependency tree and its extension, the F-score and AUC are improved to 52.24% and 79.19% respectively (2.11 percentage units' increase in F-score and 2.87 percentage units' increase in AUC). Though the performance of dependency tree kernel itself is poor (an F-score of 30.03% and an AUC of 56.11% after extension), when combined with SPT parse tree kernel, it can help improve the total performance (0.68 percentage units' increase in F-score (52.24-51.56) and 1.14 percentage units' increase in AUC (79.19-78.05)).

	P	R	F	σ_F	AUC	σ_{AUC}
Words	42.58	62.9	50.82	2.9	77.69	3.4
Words + Protein names distance	43.65	62.3	51.3	5.4	78.23	3.6
Words + Protein names distance + Keyword	46.32	61.1	52.69	4.9	80.71	4.1

Table 1. Effectiveness of different features in the feature-based kernel and their combinations on AImed

In addition, as discussed in Section 2.1.1.2, SPT and dependency path tree extensions can improve the performance by including richer context information outside SPT and dependency path. They together contribute to the improvement of performance by almost 0.7 percentage units in F-score (52.24-51.52) and 2 percentage units in AUC (79.19-77.24).

	P	R	F	σ_F	AUC	σ_{AUC}
SPT	40.09	66.74	50.13	3.2	76.32	2.7
SPT Extension	42.37	65.8	51.56	3.3	78.05	2.2
Dependency	18.76	58.33	29.17	3.2	54.37	2.3
Dependency Extension	20.49	56.18	30.03	3.6	56.11	2.1
SPT + Dependency	42.29	65.65	51.52	5.1	77.24	2.8
SPT Extension +Dependency Extension	43.71	64.65	52.24	4.8	79.19	2.6

Table 2. Effectiveness of SPT, dependency tree and their extensions on AImed

Effectiveness of different kernels

The performances of different kernels tested on AImed are shown in Table 3. Among the four individual kernels, the performance of the graph kernel is the best. As discussed in Section 2.1.1.4, the reason is that the graph kernels can treat the parser's output and word features at the same time. The performance of the feature-based kernel ranks second since it uses protein names distance and Keyword feature besides words features (otherwise, with only words features, its performance (an F-score of 50.82% and an AUC of 77.69%) is worse than that of the tree kernel). The performance of the tree kernel is almost the same with that of the feature-based kernel.

	P	R	F	σ_F	AUC	σ_{AUC}
Feature-based kernel	46.32	61.1	52.69	3.6	80.71	2.7
BOW(Miwa)			52.8		82.1	
Tree kernel	43.71	64.65	52.24	3.1	79.19	2.6
Tree kernel(Miwa)			58.2		82.5	
Graph kernel	52.66	64.56	57.20	5.6	83.27	2.8
Graph kernel(Miwa)			59.5		85.9	
Tree kernel(0.5)+ Feature-based kernel(0.5)	50.44	68.49	58.05	3.3	84.19	2.3
Tree kernel + BOW (Miwa)			60.5		85.9	
Graph kernel(0.7) +Feature-based kernel(0.3)	51.33	69.58	59.02	4.1	84.68	3.1
Graph kernel + BOW (Miwa)			57.8		85.2	
Graph kernel(0.7)+ Tree kernel(0.3)	53.43	68.57	59.66	5.8	85.51	3.4
Tree kernel+ Graph kernel (Miwa)			61.9		87.6	
Feature-based kernel(0.2) + Tree kernel(0.2)+ Graph kernel(0.6)	57.4	70.75	63.9	4.5	87. 83	2.9
Tree kernel+ Graph kernel + BOW (Miwa)			60.8		86.8	

Table 3. Effectiveness of different kernels and performance comparison with those of Miwa's method on AImed. The weights of each individual kernel in combined kernels are in the parentheses after the kernel name.

The experimental results show that, when two or more individual kernels are combined, better performances are achieved. When the graph kernel is combined with the feature-based kernel, the performance is improved by 1.82 percentage units in F-score and 1.41

percentage units in AUC. When further combined with the tree kernel, the performance is improved by 4.88 percentage units in F-score and 3.15 percentage units in AUC. The results show that the combined kernel can achieve much better performance than each individual kernel. As discussed in Section 2.1.1.4, the different kernels calculate the similarity with different aspects between the two sentences and the combination of kernels covers more knowledge by introducing more kernels and is effective for PPI extraction.

The performance comparison between our kernels and those in (Miwa et al., 2009) is also made in Table 3. Our feature-based kernel, tree kernel, graph kernel corresponds to the BOW, tree kernel, graph kernel in Miwa's method respectively. The performance of the BOW kernel Miwa's method is almost the same as our feature-based kernel in F-score (52.69% to 52.8%). The performance of the tree kernel in Miwa's method is better than our tree kernel (58.2% to 52.4% in F-score and 82.5% to 79.19% in AUC) the reason is that it uses the predicate type information to represent the dependency types (Miwa et al., 2009). The performance of the graph kernel in Miwa's method is also better than our graph kernel (59.5% to 57.2% in F-score and 85.9% to 83.27% in AUC). The reasons are: First, each word in the shortest path has two labels, and the relations in the shortest path are not replaced, but duplicated in the first subgraph. Second, the shortest path is calculated by using the constituents in the PAS structure. The words in the constituents in the shortest path are distinguishably marked as being "in the shortest path" (IP). Finally, the POS information for protein name is not attached (Miwa et al., 2009).

However, different from our results, the combination of kernels in (Miwa et al., 2009) doesn't always contribute to performance improvement. Among their kernels, the graph kernel performs best. When it is combined with the tree kernel, the performance is improved by 2.4 percentage units in F-score and 1.7 percentage units in AUC. However, when further combined with the BOW kernel, the performance drops by 1.1 percentage units in F-score and 0.8 percentage units in AUC. In fact, the performance drops when the graph kernel itself is combined with the BOW kernel. That shows the introduction of the BOW kernel into the graph kernel leads to the deterioration of the performance. Similarly, the performance of the hybrid kernel in (Kim et al., 2008) is worse than that of one of the individual kernels - the walk kernel. The reason may be that in their methods each kernel is assigned the same weight when combined. As discussed in Section 2.1.1.4, we found that only when the kernel with better performance is assigned higher weight can the combined kernel produce the best result. In our experiments the weights for feature-based kernel, tree kernel, and graph kernel are set to 0.6, 0.2, and 0.2 respectively in the order of performance rank.

Performance compared to other methods

Method	P	R	F	AUC
Our: Combined Kernel	57.4	70.75	63.9	87. 83
Miwa et al., 2008			63.5	87.9
Miwa et al., 2009	58.7	66.1	61.9	87.6
Miyao et al., 2009	54.9	65.5	59.5	
Airola et al., 2008	52.9	61.8	56.4	84.8

Table 4. Comparison on AImed. Precision, recall, F-score and AUC results for methods evaluated on AImed.

The comparison with relevant results reported in related research is summarized in Table 4. The best performing system combines multiple layers of syntactic information by using a combination of multiple kernels based on several different parsers and achieves an F-score of 63.5% and an AUC of 87.9% (Miwa et al., 2008). Our method uses only the Stanford parser output to achieve parse tree, dependency structure (path and graph) information and the performance is comparable to the former. This is due to the following three key reasons: 1) with feature-based kernel, besides the commonly used word feature, protein names distance and Keyword feature are introduced to improve the performance. Especially, the introduction of Keyword feature is a way of employing domain knowledge and proves to be able to improve the performance effectively. With the appropriate features, feature-based kernel performs best among three individual kernels. 2) the tree kernel can capture the structured syntactic connection information between the two entities. Our tree kernel combines the information of parse tree and dependency path tree and introduces their extensions to capture richer context information outside SPT and dependency path when necessary. 3) different kernels calculate the similarity with different aspects between the two sentences. Our combined kernel can reduce the danger of missing important features and, therefore, produce a new useful similarity measure. Especially, we use a weighted linear combination of individual kernel instead of assigning the same weight to each individual kernel and experimental result show the introduction of each kernel contributes to the performance improvement.

2.2 Uncertainty sampling based active learning method

One problem of applying machine learning approaches to PPI extraction is that large amounts of data are available but the cost of correctly labeling it prohibits its use. For example, MEDLINE is the most authoritative bibliographic database which has covered over 17 million references to articles from over 4800 journals, newspapers and magazines and updates weekly in the Web of knowledge. On the other hand, though the amount of unlabeled data is increasing fast, the existing labeled data can not meet research needs, for which people have to tag a lot of samples manually. However, corpus annotation tends to be costly and time consuming. People would like to minimize human annotation effort while still maintaining desired accuracy.

To accomplish this, we turned to the uncertainty sampling method of active learning. Active learning is a research area in machine learning that features systems that automatically select the most informative examples for annotation and training (Angluin, 1988).

The primary goal of active learning is to reduce the number of examples for annotation that the system is trained on, while maintaining the accuracy of the acquired information. It may construct their own examples, request certain types of examples, or determine which of a set of unsupervised examples are most usefully labeled (Cohn et al., 1994). The last approach is particularly attractive in text mining since there is an abundance of data and we would like to tag the samples as few as possible (i.e. selecting only the most informative ones for tagging). The basic idea is to combine obtaining samples and model, not like passive learning which considers each part separately. The method has been applied to text classification (McCallum & Nigam, 1998), natural language parsing (Thompson et al., 1999), name entity recognition (Shen et al., 2004) and information extraction (Thompson et al., 1999).

To reduce annotation effort in PPIs from biomedical text, we present an uncertainty sampling based method of active learning in a lexical feature-based SVM model. To verify

the effectiveness the AImed corpus and the CB corpus (Krallinger et al., 2007) are used and a 10-fold cross validation is applied.

2.2.1 Methods

The process flow of uncertainty sampling based active learning (USAL) method includes two stages. Firstly, the corpus is divided into three parts: the initial training set, the unlabeled the training set and the test set. Secondly, USAL method is introduced to select the most informative samples and add them into training set. The details are described in the following sections.

2.2.1.1 Lexical feature and preprocessing

The words surrounding the tagged protein names are used as lexical features. We divide lexical features into three types: left words, middle words and right words. Left words are the words to the left of the first protein name, middle words are the words between the first protein name and the second protein name, and right words are the words to the right of the second protein name.

A few preprocessing steps are performed before the lexical feature extraction including stopword elimination and stemming. Stopword elimination can reduce the noise, and stemming can relieve the sparse problem.

2.2.1.2 Uncertainty sampling

Uncertainty sampling (Lewis & Catlett, 1994) is an active learning method. It iteratively requests informative examples to label from unlabeled samples. Comparing to random sampling which randomly selects samples to label and train, the idea of USAL is that people only find the most informative unlabeled samples to tag.

In our method the "most informative" unlabeled samples are defined as those with the lowest absolute value of the predict scores outputted by our lexical feature-based SVM model (the lexical features used are discussed in Section 2.2.1.1). We think the smaller a sample's absolute value of the predict score is, the more uncertainty the sample has and, therefore, is more informative. Learning begins with a small pool of annotated samples and a large pool of unannotated samples. The USAL attempts to choose the most uncertain additional samples. The iterative process will not stop until the pool of unlabeled samples is empty or any other indicator reaches a threshold.

2.2.2 Experiment and discussion

2.2.2.1 Datasets

One problem in current PPI extraction research is the lack of defined criteria for evaluating the PPI systems: researchers develop and test on their own corpus and, therefore, their results are not comparable. In our experiments we used two standard datasets: AImed corpus and CB corpus. CB corpus is provided by as BioCreAtIvE II (Krallinger et al., 2007) challenge evaluation.

In our experiments, each corpus is divided into three parts. The first part is initial training set composed of 400 randomly selected samples, the second is unlabeled training set and the third is the test set composed of 400 samples which are also randomly selected.

We use Precision, Recall, F-score and Accuracy as metrics to evaluate the performance. Three group experiments are designed to verify the effectiveness and efficiency of USAL method. In the first group USAL is evaluated on using how much of the training set can

achieve the best performance; In the second group how much the learning process could be accelerated is tested by only considering one of the same uncertainty samples while keeping the PPI performance; In the third group a threshold is used to restrict the uncertainty so as to further speed up learning process, i.e. samples whose uncertainties are within the threshold are picked up to label, and the other samples are ignored.

During every round in uncertainty sampling, samples selected by the classifier from the unlabeled train dataset are added into the initial dataset. In the last round the final actual training set is formed. Assuming that e denotes the proportion between the sizes of the actual and total training set, the values of Precision, Recall, F-score and Accuracy are observed on the test set with increasing e. A 10-fold cross-validation is applied to verify the effectiveness of USAL method.

2.2.2.2 Results and discussion

First, on AImed dataset, USAL is put up as N=10 and N=100 (N denotes the number of samples which are picked up in each round). In each round a prediction is done on the test set. As shown in the Fig. 3 and Fig. 4, the performance is steadily improved by increasing the amount of training data, and when e=0.6 almost each evaluation metric (Precision, Recall, F-score and Accuracy) reaches its optimal value. It shows that labeling cost can be reduced by 40% using USAL while the performance doesn't decline.

USAL selects the unlabeled samples with most uncertainty to label (i.e. the samples with the lowest absolute value of predict scores outputted by our lexical feature-based SVM model), adds them into training set and re-trains the SVM model to pick up another N informative samples. It is an iterative process that gradually makes the training model rich and perfect. As shown in Fig. 3 and Fig. 4, on AImed dataset, no matter how many samples are selected in each round, almost each evaluation metric reaches its optimal values as e=0.6. However, sometimes the result may decline a little when e is increasing. It is a self-improvement process in which the model improves itself constantly.

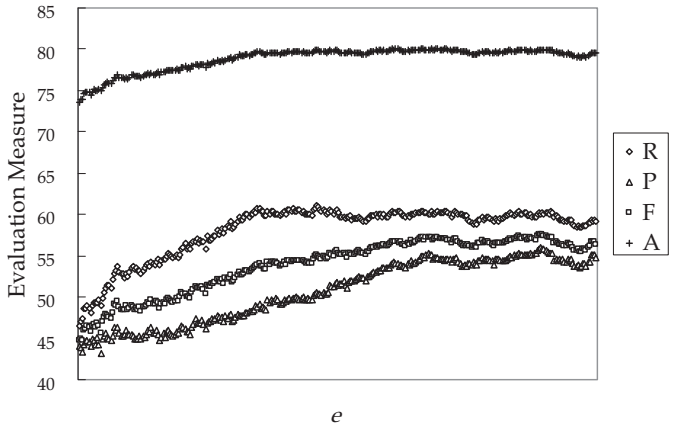

Fig. 3. Performance on AImed dataset when N is set to 10

From the above discussion, we can draw a conclusion that USAL could reduce the labeling cost without sacrificing the PPI performance. Besides, as shown in Fig. 3 and Fig. 4, Accuracy is much higher than F-score. By analyzing the result we found that as the number

of positive instances is much less than that of negative instances, F-score (which is calculated in allusion to the number of positive instances) can not be as high as Accuracy(which is calculated in allusion to the number of instances classified correctly including positive and negative instances).

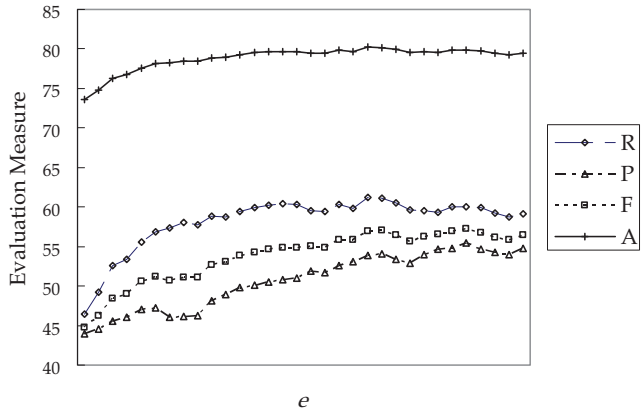

Fig. 4. Performance on AImed dataset when N is set to 100

The experiment results on CB dataset are similar to those on AImed dataset: the performance is steadily improved by increasing the amount of training data, and when e=0.8 almost each evaluation metric reaches its optimal value. It shows that annotation effort can be reduced by 20% using USAL. In addition, Accuracy is almost the same as F-score since the positive instances are almost as many as the negative instances.

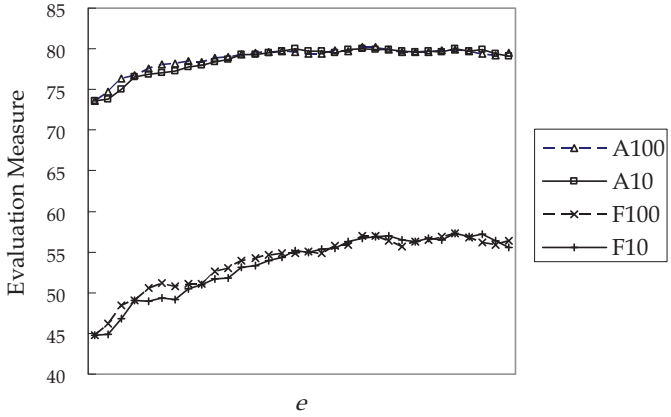

Fig. 5. F-score and accuracy comparison when N=100 and N=10 on AImed dataset

The experiment results on both AImed and CB datasets verify the effectiveness of USAL method. In addition, some experiments are designed to verify the effect of accelerating the learning process by only considering one of the same uncertainty samples. F100 and A100 denote F-score and Accuracy when N is set to 100; F10 and A10 denote F-score and Accuracy of when N is set to 10. They are compared on AImed and CB datasets respectively.

As shown in Fig. 5, the results on AImed dataset show that there is no obvious difference between N=10 and N=100 which means that N can be set to a large value to speed up learning process with less training time while keeping the performance. The similar result is found on CB dataset.

During the process of USAL, many unlabeled samples have the same uncertainty in each round. We only consider one of these samples and the others are ignored. In this way a faster learning process could be achieved while still maintaining desired performance. In order to further speed up, a threshold is used to restrict the uncertainty. Samples whose uncertainties are within the threshold are picked up to tag, and the others are ignored.

Assuming that the method used in the phase is denoted by f (T, IU) where T is the threshold and IU denotes to whether the same uncertainty samples are combined as one. RT (rounds of training) is used to measure the speed of learning process. FTS is the number of samples in final training set after USAL. The achieved F-score and accuracy of different strategies are shown in Table 5.

f (T, IU)	AImed				CB			
	RT	FTS	F	A	RT	FTS	F	A
f (∞, *False*)	32	3626	56.43	79.47	31	3656	84.34	84.23
f (∞, *True*)	28	3066	56.22	79.18	28	3042	84.5	84.39
f (3, *True*)	21	2411	56.96	79.83	27	2934	84.25	84.12
f (2, *True*)	14	1769	55.49	79.34	23	2552	83.44	83.38
f (1, *True*)	9	1087	51.4	79.05	12	1456	77.8	77.94

Table 5. Comparison of different strategies based on four indicators: RT, FTS, F and A.

In Table 5 f (∞, False) is used as the baseline in which all the training samples are used to predict the test set and the samples with same uncertainty are not combined as one. There are four group experiments with varying T and IU. Compared with f (∞, False), f (∞, True), in which all the training samples are used and the samples with same uncertainty are combined as one, reduces 3 RT and more than 600 FTS while maintaining the performance. Further, when the threshold T is introduced, f (3, True) reduces 7 RT and more than 600 FTS in AImed dataset while it reduces 1 RT and more than 100 FTS in CB dataset. While when T is set to smaller values, the performance begins to decline, and when T is set to 1 the performance degrades sharply. If T is set to an optimal threshold value (e.g. 2) keeping only one sample with the same uncertainty and using a threshold could help to reduce much training time with slight loss of performance.

2.3 Feature coupling generalization method

Many recent works (Airola et al., 2008; Bunescu et al., 2005a; Miwa et al., 2008; Miyao et al., 2009) focus on the syntactic-based methods where examples are represented by features or kernels derived from the outputs of syntactic parsers.These methods are capable of capturing syntactic relationships between entities, and show over 10% better performance than lexical features (Miwa et al., 2008; Miyao et al., 2009).

One could wonder whether methods without using syntactic information can also achieve state-of the-art performance or not. In this work, we present a novel feature representation method for the PPIE task, which is an application of our recently proposed semi-supervised learning strategy – feature coupling generalization (FCG) (Li et al., 2009). The general idea of

FCG is to learn a novel feature representation from the co-occurrences of two special types of raw features: example-distinguishing features (EDFs) and class-distinguishing features (CDFs). EDFs and CDFs refer to strong indictors for examples and for classes respectively. Intuitively, their co-occurrences in huge unlabeled data will capture indicative information that could not be obtained from labeled training data due to data sparseness. We used this method to learn an enriched representation of entity names from 17GB unlabeled biomedical texts for a gene named entity classification (NEC) task (Li et al., 2009) and found that the new features outperformed elaborately designed lexical features.

It is natural to think of applying FCG to PPIE as well as the NEC task, since there are huge amount of biomedical literatures available online which provide rich unlabeled resources. Our primary work here is to design proper EDFs, CDFs and other settings of FCG framework for the PPIE task. We also compare the performance of our methods with other syntactic-based methods proposed in previous researches on AImed corpus.

2.3.1 Feature coupling generalization

2.3.1.1 The general framework

In short, feature coupling generalization is a framework for creating new features from old features (referred to as "prior features" (Li et al., 2009)). We introduced two types of prior features: example-distinguishing features (EDFs) and class-distinguishing features (CDFs). EDFs are intuitively defined as "strong indicators" for the current examples, and CDFs are "strong indicators" for the target classes. The relatedness degree of an EDF fe and a CDF fc estimated from the unlabeled data U is defined as feature coupling degree (FCD), denoted by FCD (U, fe, fc). The FCG algorithm describes how to convert FCDs into new features. The assumptions behind this idea are: 1) the relatedness of an EDF and a CDF provides indicative information for classifying the current examples that contains the EDF. 2) Given more unlabeled data, more FCDs that cannot be obtained from labeled data can be estimated from unlabeled data.

Assuming that $F = \{f_1, \ldots, f_n\}$ is the feature vocabulary of "raw data" that contains every Boolean feature one could enumerate to describe an example, and $\mathbf{X} \subseteq \mathbf{R^n}$ is the vector space of the raw data, where each example is represented by a n-dimensional vector $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{X}$. The algorithm process of FCG can be summarized as follows:

1. Select the "example-distinguishing" part of F as EDFs, denoted by $E \subseteq F$.
2. Map each element in E to a unique higher-level concept (EDF root) in the set H, denoted by root (e): $E \rightarrow H$.
3. Select the "class-distinguishing" part of F as CDFs, denoted by $C \subseteq F$.
4. Define the set of FCD types T to measure the relatedness of EDFs and CDFs.
5. Let the vocabulary of FCD features be $H \times C \times T$ so that each FCD feature maps a tuple (h, c, t), where $h \in H$, $c \in C$, and $t \in T$.
6. Calculate FCDs from unlabeled data and convert each example from the old representation x to a new feature vector $\tilde{\mathbf{x}}$ by the equation:

$$\tilde{x}_i = \tilde{x}_{(h,c,t)} = \sum_{root(e)=h} band(e, \mathbf{x}) * FCD_t(U, e, c) \tag{4}$$

where $e \in E$, $\tilde{x}i \in \tilde{x}$, i indexes each triple (h, c, t) in $H \times C \times T$. The operator band(e, x) equals 1 if the feature e appears in the example x, and 0 otherwise.

For simplicity, here we assume that EDFs and CDFs are all extracted from F. In a broader sense, we can use the transformed feature set of original data to generate EDFs or CDFs. For

example, the "CDF II" used in the NEC task is the combination of local context words by a classifier. In the above algorithm, we assume F contains all the "feasible" combinations of original features derived from the data, and all the EDFs and CDFs are limited to be generated from this set.

2.3.1.2 Why it works

In supervised learning, usually only a subset of elements in F can be utilized. This means features that don't lead to performance improvement are regarded as irrelevant ones which are either removed before training or assigned very small weights during training to degrade their impact. In FCG framework, we also need to select a subset of F as EDFs or CDFs, but the criterion for feature selection is rather different. Here "good" EDFs or CDFs mean the performance of FCD features generated by them is good, although the single performances of them might be poor in a supervised setting. In other words, irrelevant features in supervised learning may be good EDFs or CDFs that produce indicative FCD features, so that FCG could utilize the features discarded by supervised learning.

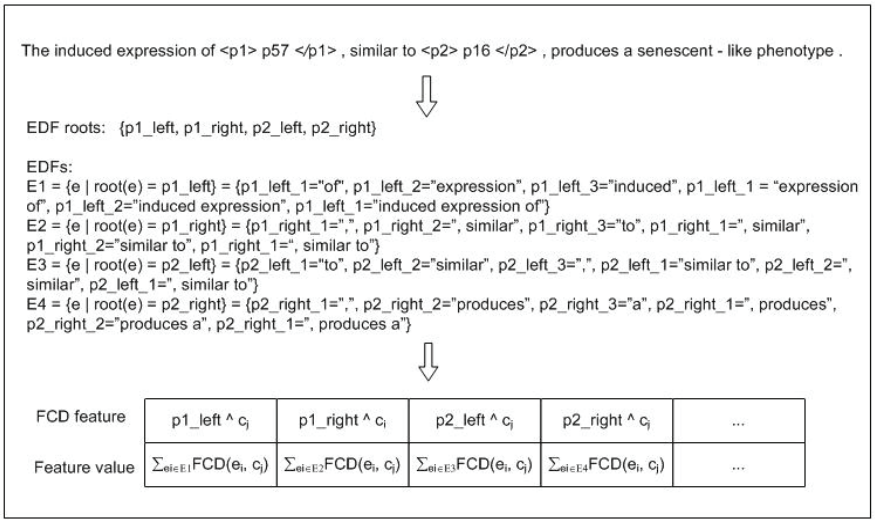

Fig. 6. An example that shows how FCG generates new feature for the PPIE task. Here only SP-EDFs are considered, and they are divided into four groups according to different EDF roots. A CDF is denoted by cj. Since only one FCD type is used here, the FCD features are indexed by the conjunction of EDF roots and CDFs.

The selection of EDFs and CDFs plays a central part in this framework. We suggested that when selecting these features, a trade-off between "indicative" and "informative" should be considered (Li et al., 2009). In the NEC task (Li et al., 2009) for determining whether an entity is a gene or protein name, the EDFs were selected as the whole entities and boundary word-level n-grams, and the CDFs were context patterns (such as "X gene" and "the expression of X") and the discretized scores of a SVM trained by local contexts. The experiments show that good results can be achieved when various types of EDFs together with hundreds of CDFs are used. We also found that these FCD features performed better in non-linear classifiers than linear ones.

2.3.2 Methods for protein-protein interaction extraction

Similar to the research methodology in our previous work (Li et al., 2009), we first designed an enhanced lexical feature set considering words and n-grams in specific positions of sentences, and then proposed several types of EDFs and CDFs for the PPIE task. We also combined lexical features and FCD features to get a higher performance.

2.3.2.1 Corpus and preprocessing

We used AImed corpus to examine our methods. We converted each sentence to lowercase, replaced XML tags like """ by their standard ASCII characters, and then tokenized a sentence by splitting tokens from non-letter or digit characters, e.g., "wild-type (d)" -> "wild - type (d)". We replaced the two focus proteins in the current example by "prot1" and "prot2", and the other proteins in the same sentence by "prot0". We also replaced all the examples with overlapping "prot1" and "prot2" (self-interactions) by the same sentence "prot1 prot2 ."

Before introducing lexical features and FCD features, we give some notions that describe words, n-grams, areas or positions in sentences with regard to interacting proteins.

Vocabularies of words: LW = {words in labeled data}, and UW = {words in unlabeled data}.

Vocabularies of n-grams: LN = {1-3 grams in labeled data}, and UN = {1-3 grams in unlabeled data}.

General areas: *GA* = {*left_area*, *inner_area*, *right_area*} –text snippets split by "*prot1*" and "*prot2*" in each sentence denoted by "*left_area prot1 inner_area prot2 right_area*".

Surrounding areas: *SA* = {*p1_left*, *p1_right*, *p2_left*, *p2_right*} – texts surrounding "*prot1*" or "*prot2*" within a 3-word window.

Specific positions: $SP = SA \times \{1, 2, 3\} = \{p1\_left\_1, p1\_left\_2, \ldots\}$ – words or n-grams that appear in certain positions of *SA*. See also the example in Figure 6.

Cross patterns: CP = {p1_dirction_offset ^ p2_direction_offset ^ distance | direction ∈ {left, right}, offset ∈ {1}, distance ∈ {0, 1, 2, 3, 4, 5, (6~7), (8~10), (11~15), (16~20), (21~30), (31~40), (40~) } } – "cross-entity" conjunctions of partial elements in SP and the discretized word count between the two proteins.

2.3.2.2 Lexical features

Note that the lexical features used in the recent works (Miwa et al., 2008; Miyao et al., 2009) based on AImed corpus only involved bag-of-words or simple variants. Here we attempt to enhance lexical-level representation by incorporating n-gram and position information and give a detailed evaluation of the contribution of each feature type. Four types of features are investigated in our work:

Bag-of-words (GA-BOW): features derived from *LW*×*GA*, e.g., *word_in_left_area ="expression"*. These features ignore word positions in the current area, which are almost the same as features of the baselines used in the works (Miwa et al., 2008; Miyao et al., 2009).

Bag-of-n-grams (GA-Lex): features from *LN*×*GA*. It simply enriches the bag-of-words representation by bigrams and trigrams.

Surrounding n-grams (SA-Lex): features from *LN*×*SA*, e.g., *p1_right="interacts with"*. They are used to highlight n-grams in the "indicating areas" since intuitively features surrounding candidate protein pairs are more indicative.

Specific n-grams (SP-Lex): features from *LN*×*SP*, which gives the information of the specific distances from protein candidates to n-grams in *SA*, e.g., *p1_right_1="interacts"*, and *p1_right_2="with"*. It provides more specific information than the "surrounding n-grams".

Our classifier for all the lexical features is SVM light (http://svmlight.joachims.org/) with linear kernel and default parameters.

2.3.2.3 FCD features

FCD measure and unlabeled data

In this work, we consider one type of FCD measure:

$$FCD1 = \frac{\log_{10}\left(co(x,y)+b\right)}{\log_{10}\left(count(x)+b\right) * \log_{10}\left(count(y)+b\right)} \quad (5)$$

where x is an EDF, y is a CDF, and co(x, y) is the co-occurrence count of x and y. The smoothing factor b is assigned 1. We log the term count to avoid highly biased values in very large corpus. This measure can be viewed as a variant of pointwise mutual information (PMI). We discussed its advantage in our previous work (Li et al., 2009).

The experimental results in our previous work (Li et al., 2009) show that the performance of FCD features increases when more unlabeled data are added. So in this work, we downloaded more data, which include all the PubMed abstracts (up to 2009) and data collection of TREC genomics track 2006 (Hersh et al., 2006; Li et al., 2009), with the total size of 20GB. We tokenized the texts in the same way as the method in Section 2.3.2.1 and tagged the protein names using the gene/protein mention tagger developed in our previous work (Li et al., 2009). We used the "dictionary-based" method because it is very fast. The method for the dictionary construction was also based on FCG and it achieved an F-score of 86.2 on BioCreative 2 Gene Mention test corpus (Wilbur et al., 2007). Note that in the PPIE task, "unlabeled" means no need to label the protein-protein interactions, but the protein names should be recognized first.

EDF selection

We examine the performances of two types of EDFs which are also derived from the lexical information introduced in Section 2.3.2.1:

SP-EDF: EDFs derived from UN×SP. It can be viewed as the extension of SP-Lex features to the vocabulary of UN. Obviously it has stronger discriminating ability than features derived from GA or SA. But the set of EDF roots was selected as SA not SP because SP resulted in higher feature dimension and space cost but the performance varied little (not reported in this work due to page limitation).

CP-EDF: text patterns derived from the set UN×CP. The set of EDF roots is CP′ = {p1_direction ^ p2_direction ^ distance | direction ∈ {left, right}, distance ∈ {0, 1, 2, 3, 4, 5, (6~7), (8~10), (11~15), (16~20), (21~30), (31~40), (40~)}}. One SP-EDF only considers a text snippet surrounding one protein, which may limit its discriminating ability, while a CP-EDF incorporates information from both sides across "prot1" and "prot2".

CDF selection

The method for generating CDFs is very simple. We used information gain – a popular feature selection technique – to rank lexical features introduced in Section 2.3.2.2 and selected top 400 ones as CDFs. This idea is similar to our prior wok on named entity classification (Li et al., 2009).

Note that rather different from lexical features, these EDFs and CDFs are not elements of the input vectors of the target classifiers. They are used only for generating FCD features which

belong to part of the final feature vectors. Figure 6 shows an example of the generation of FCD features for the PPIE task, where only SP-EDFs and one type of FCD measure are considered, so the FCD features are indexed by the conjunction of EDF roots and CDFs. It can be seen clearly that the "sparse" EDFs are "generalized" to a "higher-level" representation.

Classification model

In the work (Li et al., 2009), we found the density of FCD features was much higher than lexical features widely used in NLP and was somewhat like the feature spaces for image recognition, which inspired us to make use of non-linear classifiers. We used SVD plus RBF kernel and achieved better results than linear kernel. Similarly, for the PPIE task we also investigated the two models: linear SVM and RBF kernel based SVM. For the RBF model, we first used SVD to get a sub-space of FCD features and then used the new features as the inputs of SVM with RBF kernel. In our experiments, SVD was done on the entire AImed corpus and top 300 most significant features in left-singular matrix were selected. The parameter "-c" and "-g" of SVM light were set at 3.0 and 20.0 respectively. Then we combined the prediction scores of lexical features and FCD features given by SVMs using a simple weighted linear function, where their weights were set at 0.5 and 0.5 respectively.

2.3.3 Results and discussion

2.3.3.1 Evaluation metrics

We attempt to keep our evaluation metrics as the same as most recent works (Airola et al., 2008; Miwa et al., 2008; Miyao et al., 2009). We used F-score as the primary evaluation measure and also reported AUC. They suggested that for this task, abstract-level cross validation should be done to avoid sentences in the same abstract are both used for training and testing. We also performed abstract-wise 10-cross validation, where abstracts were divided into 10 groups, and one was used for testing and the others for training in each turn. We also extracted CDFs from each training data separately to avoid the use of answers in testing data at training time.

2.3.3.2 Lexical features

Features	P	R	F	AUC
F1	41.9	62.8	50.0	78.7
F1+F2	46.5	61.6	52.1 (+2.1)	80.5
F1+F2+F3	54.3	61.5	57.2 (+7.2)	83.6
F1+F2+F3+F4	56.8	63.1	59.0 (+9.0)	84.9

Table 6. Performance of lexical features. F1: GA-BOW, F2: GA-Lex, F3: SA-Lex, F4: SP-Lex.

Table 6 shows the performances of various combinations of lexical features. We can see the F-score of GA-BOW features is 50.0, which is similar to the results reported in the recent work (Miyao et al., 2009), where the F-score of a similar feature set is 51.1. The discrepancy may be caused by lemmatization they used, or the detailed methods in data preprocessing and splitting stages. It can be seen that features derived from n-grams, surrounding areas and specific position information improve the performance significantly and produce a surprisingly good result – 59.0 F-score and 84.9 AUC, which is competitive to most of the recent works based on syntactic parsing (Airola et al., 2008; Miwa et al., 2008; Miyao et al.,

2009) (see also Table 8). Note that this run only used simple Boolean lexical features, so it is much faster and easier to implement than syntactic based methods, which will make it of great value in practice. To our best knowledge, the features (F2, F3, and F4) are not explicitly used as Boolean lexical features in the PPIE task and their contribution is not examined well on the AImed corpus. The simple idea of creating these lexical features is similar to our work on entity classification (Li et al., 2009), just following the cue: "word -> n-grams -> n-grams in specific positions".

2.3.3.3 FCD features

Table 7 shows the performances of runs with FCD features. It can be seen that the F-score of SP-EDFs is lower than CP-EDFs, possibly because features derived from SP only consider specific n-grams surrounding one protein, so they have lower example-discriminating ability than CP, thus produce weaker FCD features. However, SP-EDFs yield higher recall and seem benefit more from non-linear classifier than CP-EDFs. From Run 2 to Run 5, we cannot see the significant advantage of non-linear classifiers, but Run 7 outperforms Run 6 by near 3 points in F-score, since in our experiments we fixed the parameters of SVD and RBF kernels for all the runs, which were tuned to optimize the performance of FCD features with both EDFs. So the results also support the fact that RBF kernel performs better for these FCD features. In Table 7, we also find that the runs with both the two EDFs (Run 6 and 7) produce a further improvement over each single feature (Run 2-5). Note that Run 7 doesn't use any classical features, while its performance is competitive to any single type of features proposed by previous researchers.

ID	Features (Models)	P	R	F	AUC
1	lexical (linear)	56.8	63.1	59.0	84.9
2	SP-EDF (linear)	43.2	58.1	49.9	76.7
3	SP-EDF (SVD + RBF)	47.9	58.5	51.4	79.8
4	CP-EDF (linear SVM)	49.5	55.9	52.2	80.3
5	CP-EDF (SVD + RBF)	50.1	54.4	52.3	78.6
6	SP+CP EDF (linear)	49.7	62.9	54.2	81.3
7	SP+CP EDF (SVD + RBF)	59.9	57.8	58.1	83.1
8	lexical +FCD (1 + 7)	59.3	68.2	62.9	87.3

Table 7. Performance of FCD features

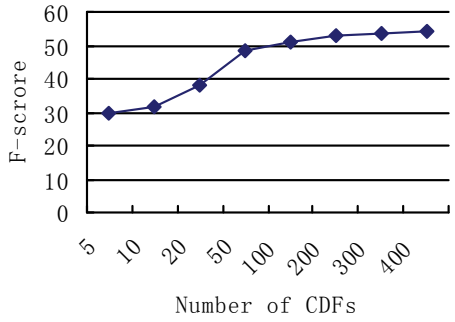

Fig. 7. Relationship between the number of CDFs and the performance of FCD features.

It is promising to see that Run 8 (Table 7) that combines the results of lexical and FCD features by simple average produces a significant improvement over the lexical features on both F-score and AUC, although the baseline is rather strong. The results in Table II have proved the success of applying FCG to the PPIE task. However, the improvement of FCD features is not as huge as the entity classification task (Li et al., 2009), where the improvement is over 6 points in F-score. We think it is mainly because the EDFs and CDFs investigated here are relatively simple. For example, only information of words surrounding the interacting proteins within a 3-word window is considered in EDFs. Also the EDFs in the PPIE task are not so "obvious" as that in the NEC task, since it is more difficult to select the "example-distinguishing" part of a sentence concerned with two interacting proteins. In addition, the introduction of more types of CDFs would also enhance the performances, since in Figure 7, the performance of PPI extraction increases when more CDFs are incorporated.

2.3.3.4 Comparison with other systems

Methods	F	AUC
Our method (Combined)	62.9	87.3
(Miwa et al., 2008)	62.7 (64.3)	83.2 (87.9)
(Miyao et al., 2009)	59.5	
Our method (Lex)	59.0	84.9
Our method (FCD)	58.1	83.1
(Airola et al., 2008)	56.4	84.8
Bag-of-words	50.0	78.7

Table 8. Comparison with other systems

For the work in (Miwa et al., 2008), the scores in brackets are their reported results obtained by removing all the examples with self-interaction protein pairs in AImed corpus.

In Table 8, we compare the performances of our methods with other results reported in previous researches evaluated on AImed corpus. Although it is difficult to make strict comparison due to different methods for data splitting and pre-processing, it can be seen that our combined method is among the state-of-the-art systems. It is an important finding for both biomedical text mining and NLP community, because unlike other methods, not any syntactic information is used in this run. Another interesting finding is that our simple lexical features create a strong baseline for other methods to challenge, since it not only achieves good results but is much more efficient than syntactic based methods.

In our previous work (Li et al., 2009), we discussed the efficiency of FCG in real world applications. In summary, for real-time application, it needs the support of "feature-level" search engine. Alternatively if the task can be divided into non-real-time sub-tasks, we can run FCG on each sub-task in an offline manner. For example, in this task, we can generate a huge number of lexical patterns indicating for PPI and used FCG to remove noisy patterns. Then the refined patterns are used as features integrated into the lexical feature-based method (Section 2.3.2.2). The idea is similar to the dictionary construction for the NER task (Li et al., 2009). In this way, we can both utilize the information from unlabeled data and make the system efficient.

3. Conclusions

In this chapter, three different protein-protein interactions extraction approaches representing the state-of-the-art research in this area are introduced.

Firstly, we present a multiple kernels learning based approach to extracting protein-protein interactions from biomedical literature. The approach combines feature-based kernel, tree kernel, and graph kernel with different weights and achieves much better performance than each individual kernel. This indicates that the features in individual kernels are complementary and the combined kernel can well integrate them: 1) the flat entity information captured by the feature-based kernel; 2) the structured syntactic connection information between the two entities captured by the tree kernel, graph kernel and POS path kernel.

Secondly, we present a lexical feature-based SVM model to extract PPI information from biomedical literature. During the supervised learning process, to reduce annotation effort while maintaining the PPI extraction performance, we employ an uncertainty sampling based method of active learning to tag the most informative unlabeled samples. The experiment results show that our method can reduce the labeling cost by 40% and 20% on the two corpora respectively without degrading the performance. In addition, the number of samples picked up in each round can be set to a large value to speed up learning process. Besides, to further accelerate USAL process, our method only reserves one of the same uncertainty samples. To further speed up learning process, a threshold is used to restrict the uncertainty. The samples whose uncertainties are within the threshold are picked up to tag, and the other samples are ignored. The experiment results show these methods reduce much annotation effort and training time with slight loss of performance.

Finally, we present the application of FCG semi-supervised learning strategy to the PPI extraction task and show that FCD features derived from simple lexical information can achieve good results and produce further improvement over a high baseline.

Currently, PPI extraction methods generate poorer results compared with other domains such as newswire and there is still much room for performance improvement. In the future work, we will focus on designing EDFs and CDFs that cover more lexical or linguistic information (e.g., from shallow or syntactic parsing) of the whole sentences. Since many experiments show that FCD features perform well in non-linear classifiers, we will examine other popular learning techniques in pattern recognition. It is encouraging to see that FCG can perform well in the two different NLP tasks: entity classification and relation extraction, so we will continuously examine this method in more tasks on natural language processing and machine learning. In addition, as discussed in Section 2.1.2.2, introducing more domain knowledge such as protein-protein interactions ontology into PPI extraction from biomedical literature may help address the emerging challenges of deep natural language understanding.

4. References

Airola, A., Pyysalo, S.; Björne, J.; Pahikkala, T.; Ginter, F. & Salakoski, T. (2008). All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC Bioinformatics*, no.9(Suppl 11):S2.

Angluin, D. (1988). Queries and concept learning. *Mach Learn*, vol. 2, no. 4, pp., 319-342.

Bader, G.D.; Betel, D. & Hogue, C.W. (2003). BIND: the biomolecular interaction network database, *Nucl. Acids Res*, vol. 31, no. 1, pp. 248-250.

Blaschke, C. & Valencia, A. (2002). The frame-based module of the Suiseki information extraction system. *IEEE Intelligent Systems*, vol. 17, no. 2, pp.14-20.

Bunescu, R.C. & Mooney, R.J. (2005a). A shortest path dependency kernel for relation extraction. In: *Proceedings of Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 724-731, British Columbia, Canada.

Bunescu, R.C.; Ge, R.; Kate, R.J.; Marcotte, E.M.; Mooney, R.J.; Ramani, A.K. & Wong, Y. (2005b). Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. *Artif Intell Med*, vol. 33, no. 2, pp. 139-155.

Bunescu, R.C. & Mooney, R.J. (2006). Subsequence kernels for relation extraction. In: *Advances in Neural Information Processing Systems 18*, Weiss Y, Schölkopf B, Platt J, (Ed.), pp. 171-178. MIT Press, Cambridge, MA.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, vol. 15, no. 2, pp. 201-221.

Collins, M. & Duffy, N. (2001). Convolution Kernels for Natural Language", In: *Proceedings of 14th Conference on Neural Information Processing Systems*, pp. 625-632, Cambridge, MA.

Corney, D.P.; Buxton, B.F.; Langdon, W.B. & Jones D.T. (2004). BioRAT: extracting biological information from full-length papers. *Bioinformatics*, vol. 20, no. 17,pp.3206-3213.

Cristianini, N. & Taylor, J.S. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press, 0521780195, New York.

Danger, R.; Rosso, P.; Pla, F. & Molina, A. (2008). PPIEs: Protein-protein interaction information extraction system. *Journal of Procesamiento del lenguaje natural,* no. 40, pp. 137-143.

Fundel, K.; Küffner, R. & Zimmer, R. (2007). RelEx-Relation extraction using dependency parse trees. *Bioinformatics*, vol. 23, no. 3, pp.365-371.

He, Y.L.; Nakata, K. & Zhou, D. (2008). Ontology-Based Protein-Protein Interactions Extraction from Literature Using the Hidden Vector State Model. In: *Proceedings of 2008 IEEE International Conference on Data Mining Workshops*, pp.736-743,Pisa, Italy.

Hersh, W.; Cohen, A.; Roberts P. & Rekapalli, H.K. (2006). TREC 2006 genomics track overview, In: *Proceedings of 15th Text REtrieval Conference (TREC)*, Gaithersburg, Maryland.

Kim, S.; Yoon, J. & Yang, J. (2008). Kernel approaches for genic interaction extraction, *Bioinformatics*, vol. 24, no. 1, pp. 118-126.

Krallinger, M.; Leitner, F. & Valencia, A. (2007). Assessment of the second BioCreative PPI task: automatic extraction of protein–protein interactions. In: *Proceedings of the 2nd BioCreAtIvE Workshop*, pp. 41-54,Madrid, Spain: CNIO.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In: *Proceedings of the Eleventh International Conference on Machine Learning*, pp.148-156. New Brunswick, USA.

Li, Y. P.; Lin, H.F. & Yang.Z.H. (2009). Incorporating Rich Background Knowledge for Gene Named Entity Classification and Recognition, *BMC Bioinformatics*, no. 10:223.

McCallum, A. & Nigam, K. (1998). Employing EM and Pool-Based Active Learning for Text Classification. In: *Proceedings of the Fifteenth International Conference on Machine Learning*, pp.350-358. Madison, USA.

Miwa, M.; Sætre, R.; Miyao, Y.; Ohta, T. & Tsujii, J. (2008). Combining Multiple Layers of Syntactic Information for Protein-Protein Interaction Extraction. In: *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine*, pp.101-108, Turku, Finland: Centre for Computer Science.

Miwa, M.; Soetre, R.; Miyao, Y. & Tsujii, J. (2009). Protein-protein interaction extraction by leveraging multiple kernels and parsers, *Int. J. Med. Inform.* vol. 78, no. 12, pp. e39-46.

Miyao, Y.; Sætre, R.; Sagae, K.; Matsuzaki; T. & Tsujii J. (2009). Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, vol. 25, no. 3, pp. 394-400.

Moschitti, A. (2006). Making tree kernels practical for natural language processing. In: *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy.

Ono, T.; Hishigaki, H.; Tanigam, A. & Takagi, T. (2001). Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, vol. 17, no. 2, pp.155-161.

Sekimizu, T.; Park, H.S. & Tsujii, J. (1998). Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. *Genome Inform*, vol. 9,pp. 62-71.

Shen, D.; Zhang, J.; Su, J.; Zhou, G. D. & Tan, C. L. (2004). Multi-Criteria-based Active Learning for Named Entity Recognition. In: *Proceedings of the 42nd Association of Computational Linguistic*, pp.589-596. Barcelona, Spain.

Thompson, C. A.; Cali, M. E. & Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. In: *Proceedings of the 16th International Conference on Machine Learning*, pp.406-414. Bled, Slovenia.

Wilbur, J.; Smith, L. & Tanabe, L. (2007). BioCreative 2. gene mention task, In: *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, pp. 7–16, Madrid, Spain: CNIO.

Xenarios, I.; Rice, D.W.; Salwinski, L.; Baron, M.K.; Marcotte, E.M.; Eisenberg, D. (2000). DIP: the Database of Interacting Proteins, *Nucleic Acids Res*, vol. 28, no. 1, pp. 289-291.

Xiao, J.; Su, J.; Zhou, G.D. & Tan, C.L. (2005). Protein-Protein Interaction Extraction: A Supervised Learning Approach. In: *Proceedings of the First International Symposium on Semantic Mining in Biomedicine*, pp. 10-13, Hinxton, Cambridge, UK.

Yang, Z.H.; Lin, H.F. & Li, Y.P. (2010). BioPPISVMExtractor: A protein-protein interaction extractor for biomedical literature using SVM and rich feature sets. *J Biomed Inform,* vol. 43, no. 1, pp.88-96.

Zanzoni, A.; Montecchi-Palazzi, L.; Quondam, M.; Ausiello, G.; Helmer-Citterich, M. & Cesareni, G. (2002). Mint: A molecular interaction database, *FEBS Lett*. vol. 513, no. 1, pp. 135-140.

Zhang, M.; Zhang, J.; Su, J. & Zhou, G.D. (2006). A Composite Kernel to Extract Relations between Entities with both Flat and Structured Feature. In: *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 825-832, Sydney, Australia.

Zhou, D.; He, Y. & Kwoh, C.K. (2006). Extracting Protein-Protein Interactions from the Literature Using the Hidden Vector State Model. In: *Proceedings of the International Workshop on Bioinformatics Research and Applications*, pp. 718-725, Reading, UK. LNCS 3992, Springer-Verlag.

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.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Hongfei Lin, Zhihao Yang and Yanpeng Li (2011). Protein-Protein Interactions Extraction from Biomedical Literatures, Biomedical Engineering, Trends, Research and Technologies, Dr. Sylwia Olsztynska (Ed.), ISBN: 978-953-307-514-3, InTech, Available from: http://www.intechopen.com/books/biomedical-engineering-trends-research-and-technologies/protein-protein-interactions-extraction-from-biomedical-literatures

INTECH
open science | open minds

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