BIOINFORMATICS

Vol. 20 no. 18 2004, pages 3604–3612
doi:10.1093/bioinformatics/bth451

Discovering patterns to extract protein–protein interactions from full texts

Minlie Huang[1], Xiaoyan Zhu[1,], Yu Hao[1], Donald G. Payan[2], Kunbin Qu[2] and Ming Li[3,1]*

[1]State Key Laboratory of Intelligent Technology and Systems (LITS), Department of Computer Science and Technology, University of Tsinghua, Beijing, 100084, China [2]Rigel Pharmaceuticals Inc, 1180 Veterans. Blvd, South San Francisco, CA 94080, USA and [3]Bioinformatics Laboratory, School of Computer Science, University of Waterloo, N2L 3G1, Ontario, Canada

Received on April 2, 2004; revised on June 22, 2004; accepted on July 7, 2004
Advance Access publication July 29, 2004

ABSTRACT

Motivation: Although there are several databases storing protein–protein interactions, most such data still exist only in the scientific literature. They are scattered in scientific literature written in natural languages, defying data mining efforts. Much time and labor have to be spent on extracting protein pathways from literature. Our aim is to develop a robust and powerful methodology to mine protein–protein interactions from biomedical texts.

Results: We present a novel and robust approach for extracting protein–protein interactions from literature. Our method uses a dynamic programming algorithm to compute distinguishing patterns by aligning relevant sentences and key verbs that describe protein interactions. A matching algorithm is designed to extract the interactions between proteins. Equipped only with a dictionary of protein names, our system achieves a recall rate of 80.0% and precision rate of 80.5%.

Availability: The program is available on request from the authors.

Contact: zxy-dcs@tsinghua.edu.cn; mli@uwaterloo.ca

INTRODUCTION

Recently, there have been many accomplishments in literature data mining for biology applications, many of which focus on extracting protein–protein interactions that are scattered throughout the scientific literature. Many research projects have been devised to collect information on protein interactions. Several databases have been constructed to store such data, for example, *Database of Interacting Proteins* (Xenarios *et al*., 2000; Salwinski *et al*., 2004). However, most data in these databases were accumulated manually and inadequately, at high costs. Yet, scientists continue to publish their discoveries on protein interactions in scientific journals, without submitting their data to specific databases. As a result, most protein interactions still exist only in the scientific literature, written in natural languages and hard to be processed with computers.

How to extract protein interaction information has been an active research subject. Among all methods, natural language processing (NLP) techniques are preferred and have been widely applied. These methods can be regarded as parsing-based methods. Both full and partial (or shallow) parsing strategies have been attempted. For example, a general full parser with grammars for biomedical domain was used to extract interaction events by filling sentences into argument structures (Yakushiji *et al*., 2001). No recall or precision rate using this approach was given. Another full parsing method, using bidirectional incremental parsing with combinatory categorial grammar (CCG), was proposed (Park *et al*., 2001). This method first localizes target verbs, and then scans the left and right neighborhood of the verb respectively. The lexical and grammatical rules of CCG are more complicated than those of a general CFG. The recall and precision rate of the system were reported to be 48% and 80%, respectively. Another full parser utilized a lexical analyzer and context-free grammar (CFG), to extract protein, gene and small molecule interactions with a recall rate of 63.9% and precision rate of 70.2% (Temkin and Gilder, 2003). Similar methods such as preposition-based parsing to generate templates were proposed (Leroy and Chen, 2002), processing abstracts with a template precision of 70%. A partial parsing example is the relational parsing for the inhibition relation (Pustejovsky *et al*., 2002), with a comparatively low recall rate of 57%. In conclusion, the above methods are inherently complicated, requiring many resources, and performances are not satisfactory.

Another popular approach uses pattern matching. As an example, a set of simple word patterns and part-of-speech rules are manually coded for each verb in order to extract special kind of interactions from abstracts (Ono *et al*., 2001). Ono's method outperforms parsing-based methods in that it

*To whom correspondence should be addressed.

is based on simple rules. It is able to handle long sentences and achieves high performances with a recall rate of 85% and precision rate of 94% for yeast and *Escherichia coli*. However, manually writing patterns for every verb is not practical for general purpose applications. In GENIES, more complicated patterns with syntactic and semantic constraints are used (Friedman *et al*., 2001). GENIES even uses semantic information. However, GENIES' recall rate is low. In the above methods, patterns are hand-coded without exception. Because there are many verbs and their variants describing protein interactions, manually coding patterns for every verb and its variants is not feasible in practical applications.

Most of the above methods process MEDLINE abstracts (Ng and Wong, 1999; Thomas *et al*., 2000; Park *et al*., 2001; Yakushiji *et al*., 2001; Wong, 2001; Leroy and Chen, 2002). Because there is neither an accurate task definition on this problem nor a standard benchmark, it is hard to compare the results from various methods fairly (Hirschman *et al*., 2002). Furthermore, as MEDLINE has become a standard resource for researchers, the results on the more difficult task of mining full text have been largely ignored.

In this paper, we propose a novel and robust method to discover patterns to extract protein interactions. It is based on dynamic programming (DP). In the realm of homology search between protein or DNA sequences, a global and local alignment algorithm has been thoroughly researched (Needleman and Wunsch, 1970; Smith and Waterman, 1981). In our method, by aligning sentences using dynamic programming, similar parts in sentences could be extracted as patterns. Compared with the previous methods, our proposal is different in the following ways: first, it processes full biomedical texts, rather than only abstracts; second, it automatically mines verbs for describing protein interactions; third, this method automatically discovers patterns from a set of sentences whose protein names are identified, rather than manually creating patterns as the previous methods do; last, our method has low time complexity and it is able to process very long sentences.

METHOD

In this section, we first discuss the sequence alignment algorithm. Then the pattern generating algorithm is presented. At last, the matching algorithm for extracting interactions between proteins is described.

Alignment algorithm

Suppose we have two sequences $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_m)$. They are defined over the alphabet $\Sigma = \{a_1, a_2, \ldots, a_{l-1}, \text{'–'}\}$ where each a_i is called a character, and '–' denotes a white-space or a gap. We assign a score to measure how similar X and Y are. Define $F(i, j)$ as the score of the optimal alignment between the initial segment from x_1 to x_i of X and the initial segment from y_1 to y_j of Y. $F(i, j)$ is recursively calculated as follows:

$$F(i,0) = 0, F(0,j) = 0, x_i, y_j \in \Sigma \tag{1a}$$

$$F(i,j) = \max \begin{cases} 0, \\ F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + s(x_i, \text{'–'}) \\ F(i, j-1) + s(\text{'–'}, y_j) \end{cases} \tag{1b}$$

where $s(a, b)$ is defined as follows:

$$s(a,b) = \log\left[\frac{p(a,b)}{(p(a) * p(b))}\right] \tag{2}$$

Here, $p(a)$ denotes the appearance probability of character a, and $p(a, b)$ denotes the probability that a and b appear at the same position in two aligned sequences. Probabilities $p(a)$, $p(a, b)$ can be estimated by formula (3*a*–*b*) with pre-aligned training data:

$$p(a) = [C(a) + 1] \Big/ \sum_{\text{all } x} [C(x) + 1] \tag{3a}$$

$$p(a,b) = [C(a,b) + 1] \Big/ \sum_{\text{all pairs}(x,y)} [C(x,y) + 1] \tag{3b}$$

where $C(a)$ denotes the count of character a appearing in the training corpus, and $C(a, b)$ denotes the number of aligned pair (a, b) being observed in the training set. Note that $(C(\cdot) + 1)$ instead of $C(\cdot)$ is used since characters or pairs are possibly never observed in the training data because of data sparseness. Thus, formula (3*a*–*b*) is a smoothed estimation.

However, the calculation of scores for a gap will be different. In formula (2), when a or b is a gap, the scores cannot be directly estimated by formula (3*a*–*b*) because of two reasons: first, the case that a gap aligns to another gap will never happen in the alignment algorithm since it is not optimal, therefore, what s('–','–') exactly means is unclear; second, gap penalty should be negative, but it is unclear what p('–') should be. In DNA sequence alignment, these gap penalties are simply assigned with negative constants. Similarly, we tune each gap penalty for every character with some fixed negatives. Then a linear gap model is used.

Given a sequence of gaps with length n which aligns to sequence $X = (x_1, x_2, \ldots, x_n)$ with no gaps, the linear penalty is as follows:

$$\gamma(n) = \sum_{i=1}^{n} s(\text{'–'}, x_i). \tag{4}$$

For sequence X of length n and sequence Y of length m, totally $(n + 1)*(m + 1)$ scores will be calculated by applying formula (1*a*–*b*) recursively. Store the scores in a matrix $F = F(x_i, y_i)$. Through back-tracing in F, the optimal local alignment can be found. The local alignment algorithm can be found at http://learn.tsinghua.edu.cn/homepage/2000315648/local_align.pdf.

Table 1. Main tags used in the method

Tag name	Tag description
PTN	Special tag for protein name
GAP('–')	Tag for the gap
NN	Noun, singular or mass
NNS	Noun, plural
IN	Preposition, subordinating conjunction
CC	Coordinating conjunction
TO	to
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-third person singular present
VBZ	Verb, 3rd person singular present
RB	Adverb
JJ	Adjective

Table 2. Gap penalties for main tags

Tag	Penalty	Tag	Penalty	Tag	Penalty
PTN	−10	IN	−6	VB	−7
NN	−8	CC	−6	VBD	−7
NNS	−7	TO	−1	VBG	−7
VBN	−7	VBP	−7	VBZ	−7
RB	−1	JJ	−1		

In our method, the alphabet consists of three kinds of tags: part-of-speech tags, as those used by Brill's tagger (Brill, 1995); tag *PTN* for protein names; and tag *GAP* for a gap. The main tags are listed in Table 1. Gap penalties for these tags are shown in Table 2.

Pattern generating algorithm

In our method, a data structure called sequence structure is used. It consists of a sequence of tags (including *PTN* and *GAP*) and word indices in the original sentence for each tag (for tag *PTN* and *GAP*, word indices are set to −1). Through the structure, we are able to trace which words align together.

A pattern structure is also devised, which is made up of three parts: a sequence of tags; an array of word index lists for each tag, where each list defines a set of words for a tag that can appear at the corresponding position of a pattern; a count of how many times the pattern has been extracted out from the training corpus. With the structure, the pattern generating algorithm is shown in Figure 1. The filtering rules used in Figure 1 are listed in Table 3.

In the first step of the algorithm, *useless* tags are removed from each sequence. Tags like *JJ* (adjective) and *RB* (adverb) are too common and can appear at almost every position in a sentence; hence, if patterns include such kind of tags, they lose the generalization power. Some tags such as *DT* (determiner) only play a functional role in a sentence and they are useless for pattern generation. Therefore, as illustrated in the first step of Figure 1, we remove directly useless tags such as *JJ*, *JJS* (superlative adjective), *JJR* (comparative adjective), *RB*, *RBS* (superlative adverb), *RBR* (comparative adverb) and *DT* from the sequences. Furthermore, to control the form of a pattern, filtering rules shown in Table 3 are adapted. Verb or noun tags define the action type of interactions, thus they are indispensable, as the first rule shows. The second rule guarantees the integrality of a pattern because tags like *IN* and *TO* must be followed by an object. The last one requires symmetry between the left and right neighborhood of *CC* tag. Certainly more rigid or looser filtering rules can be applied to meet special demands.

Furthermore, we use a threshold d in the algorithm. If a pattern appears less than d times, it will be discarded; otherwise they will cause many matching errors. Through tuning the threshold, the generalization and usability of patterns can be controlled. The larger the threshold is, the more accurate patterns are.

Input: an integer d, a sequence set $S = (s_1, s_2, \cdots, s_n)$
Output: pattern set P
1. Remove useless tags from each s_i in S
2. For any $(s_i, s_j) \in S(i \neq j)$ do
   ①.Do alignment for s_i and s_j with formula (1a-b). Suppose aligned output is X_a and Y_b;
   ②.Extract the identical characters at the same positions in X_a and Y_b as pattern p. Add word indices to pattern structure;
   ③.Judge whether p is legal, using the filtering rules. If it is illegal, go to step 2;
   ④.If p exists in P, increase the count of p with 1. If not, add p to P with a count of 1;
3. For every p in P , do
   If the count of p is less than d, discard p;
4. Output P.

Fig. 1. Pattern generating algorithm. It has a time complexity of $O(n^2)$ in the corpus size n.

Table 3. Filtering rules

1. If a pattern has neither verb tag nor noun tag, reject it.
2. If the last tag of a pattern is IN or TO, reject it.
3. If the left neighborhood of a CC tag is not equal the right one in a pattern, reject the pattern.

Pattern matching algorithm

To evaluate patterns generated previously, a matching algorithm is explored. Because one pattern possibly matches a sentence at different positions, the algorithm must be capable of finding out multiple matches. Here if we think a pattern as

a motif, and sentence as a protein sequence, then our task is similar to finding out all motifs in the sequence.

First of all, matches scoring less than a threshold are useless because there is always short local alignment with a small score even between entirely unrelated sequences. An alignment is short if the length of aligned segments is less than three, because a protein interaction requires at least three tags (a subject, an action word and an object). We would look for multiple matches in a tag sequence $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_m)$ is a pattern. The recurrence defined by formulae (5a–b), is different from that of the pattern generating algorithm. Formula (5a) only allows matches to end when they score at least T.

$$F(0,0) = 0$$

$$F(i,0) = \max \begin{cases} F(i-1,0) \\ F(i-1,j) - T, j = 1,2,\ldots m \end{cases} \quad (5a)$$

$$F(i,j) = \max \begin{cases} F(i,0), \\ F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + s(x_i, \text{`–'}) \\ F(i,j-1) + s(\text{`–'}, y_j) \end{cases} \quad (5b)$$

The total score of all matches is obtained by adding an extra cell $F(n+1,0)$ to the score matrix F, using (5a). By tracing back from cell $(n+1,0)$ to (0,0), the individual match alignment will be obtained.

Because each pattern has different length, threshold T in formula (5a) should not be identical for different patterns. Assume the sequence for a pattern p_i is $Y = (y_1, y_2, \ldots, y_m)$, then threshold T is calculated as follows:

$$T = \eta \sum_{i=1}^{m} s(y_i, y_i) \quad (6)$$

Here, we take the factor $\eta = 0.5 \times \sum s(y_i, y_i)$ is the maximum score when a pattern matches a sentence perfectly (Fig. 2).

From the algorithm, a match is accepted only when three conditions are satisfied: first, a pattern has a local optimal match with the sentence; second, words of the matching part are inside the word set of the pattern; third, decision rules are satisfied.

To illustrate the matching details, a measurement data structure *m Vector* is defined as follows:

$$mVector = (cLen, cMatch, cPtn, cVb), \quad (7)$$

where *cLen* is the length of the pattern; *cMatch* is the number of matched tags; *cPtn* is the number of protein name tag (*PTN*) skipped by the alignment; and *cVb* is the number of skipped verbs. Based on the structure, decision rules shown in Table 4 are used. There are two parameters P and V used in the decision rules, which can be adjusted according to the performance of the experiments. Here, we take $P = 0$ and $V = 2$. The first rule shows that the deletion of elements in a pattern is inhibited. The second and third rules limit the maximum numbers of skipped tag *PTN* and *VB*.

Input: a pattern set $P = (p_1, p_2, \ldots, p_n)$, a sequence X
Output: aligned result set R
1. For every pattern p_i in P, do
   ①.Set threshold T for pattern p_i, using formula (6);
   ②.For X and the sequence of pattern p_i, build score matrix F using formula (5a-b);
   ③.Trace back to find multiple matches. Suppose the results are $A_r = \{X_{a1}, X_{a2}, \ldots, X_{ar}\}$;
   ④.For every result X_{ai} in A_r
   (a) Check whether every word in X_{ai} aligned to p_i appears in the corresponding position of p_i, if not, go to step ④;
   (b) Fill all the data in *mVector* as Formula (7);
   (c) Decide to accept or reject the match according to decision rules. If reject, go to step ④;
   (d) Add X_{ai} to the result set R;
2. Output R.

Fig. 2. Pattern matching algorithm. Its time complexity is $O(|P| * (|X| * |\bar{p}|))$ in pattern set size $|P|$, sequence length $|X|$ and average length of pattern $|\bar{p}|$.

Table 4. Decision rules

Input: two parameters P and V
1. if cMatch $\neq$ cLen, reject the match;
2. if cPtn $>$ P, reject the match;
3. if cVb $>$ V, reject the match;

SYSTEM OVERVIEW

Our system uses the framework of *PathwayFinder* (Yao *et al.*, 2004). Its architecture is shown in Figure 3.

The external resource required in our method is a dictionary of protein names, where about 60,000 items are collected from both databases of *PathwayFinder* and several web databases, such as *TrEMBL*, *SWISSPROT* (O'Donovan *et al.*, 2002) and *SGD* (Cherry *et al.*, 1997), including many synonyms. The training corpus contains about 1200 sentences. It is explained in detail in the section on results.

For an input sentence, first some filtering rules are adapted to remove useless expressions such as citations ('[1]') at the pre-processing phase. Then protein names in the sentence are identified according to the protein name dictionary and the names are replaced with a unique label. Subsequently, the sentence is part-of-speech tagged by Brill's tagger (Brill, 1995), and then the tag of protein names is changed to tag *PTN*. There are about 36 tags in the original Brill's tagger, the majority of which have been listed in Table 1. Last, the tag sequence is added into the corpus at the training phase or processed by the matching algorithm at the testing phase.

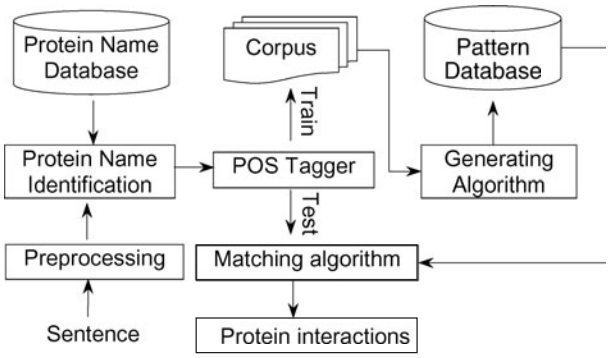

Fig. 3. System architecture.

Because the pattern generating algorithm is aligning tag sequences, the accuracy of part-of-speech tagging is crucial. However, Brill's tagger only got overall 83% accuracy for biomedical texts because biomedical texts contain many unknown words. Here we propose a simple and effective approach called pre-tagging strategy to improve the accuracy. It is described in the section on results at length.

We do not include the parameter estimation module in Figure 3. About 200 aligned sentence pairs are collected in advance, and formula (3a–b) and (4) are simply applied to estimate the scores.

RESULTS

The corpus consists of about 1200 sentences. They are collected by the following steps: first a web crawler program is used to download biomedical papers of interest from the internet with the keyword 'protein–protein interaction', and the papers are sorted automatically according to their relevance to the query; then the first 50 papers are selected, and full texts are segmented into sentences, where the number of sentences is about 65,536; then protein names in these sentences are identified; finally, sentences with fewer than two protein names are discarded. Note that the sentences which contain at least two protein names may include no protein interactions at all. Our corpus does not exclude such kind of sentences.

Our evaluation experiments include four tests: part-of-speech tagging, mining verbs, extracting patterns and evaluating the precision and recall rate.

Part-of-speech tagging

As mentioned before, the accuracy of part-of-speech tagging is crucial for our method. Thus, high accuracy of tagging is necessary. Brill's tagger consists of three parts: a lexicon, a lexical rule set and a contextual rule set. To tag a sentence, first of all each word is assigned an initial tag according to the lexicon for known words. For unknown words, lexical rules are applied and each rule gives a tag. Then contextual rules are used to tune the initial tags. However, most lexical rules are not suitable for unknown words in biomedical texts, causing a large number of tagging errors. Our idea is: first recognize an unknown word using morphological features and assign a tag to it; then apply Brill's tagger with a pre-tagging strategy. Table 5 shows features used in the morphology-based tagging. The last feature of Table 5 indicates that if a word is hyphenated by several words and the last word can be tagged as *VBD*, *VBG*, *VBN*, *IN* and *JJ*, it will be tagged as *JJ*. If an unknown word satisfies feature listed in the first column of Table 5, the corresponding tag in the second column is assigned to it. We call the tag a pre-tag. Then the following pre-tagging strategy is applied:

(1) Do not apply lexical rules or contextual rules on pre-tagged words;

(2) For the neighborhoods of pre-tagged word, apply contextual rules according to the pre-tagged word.

The tagging results are shown in Table 6. The experiment is performed on GENIA corpus which is the largest

Table 5. Features used in morphology-based tagging

Feature	Tag	Examples
AllCaps	NN	CSN, APC/C
FirstLowRestCaps	NN	mHOS, rMGMT
ContainCapsAndGreek	NN	TNF-alpha
ContainCapsAndRoman	NN	TbR-I, CaMKII
ContainCapsDigit	NN	E3, Mdm2
FirstLowRestDigit	NN	p22, p24
ContainCapsSymbols	NN	APP+1, UbL[R23], p42/44, (H)Mdm2
TwoMoreCaps	NN	DHFRts, RNAi
Last_s_and_prev_Caps	NNS	SnRKs, AIPs
∗-VBD_or_VBG_or_ VBN_or_IN_or_JJ (∗ means any word)	JJ	IIbeta-binding, B-bound, ubiquitin-like

Table 6. Part-of-speech tagging accuracy

Tag	Without pre-tagging (%)	With pre-tagging (%)	Total count
NN	68.3	92.4	139883
IN	98.8	99.7	60523
JJ	80.1	84.2	47988
NNS	94.7	96.9	32930
CC	91.9	92.3	17496
VBN	93.0	93.7	14664
RB	95.4	94.4	10464
VBD	85.4	85.5	10350
TO	99.5	99.5	7417
VB	96.9	97.1	5495
VBG	90.6	90.6	4750
VBZ	94.4	94.5	9213
VBP	79.2	79.1	6843
Overall	82.8	92.8	368016

Table 7. Mined verbs describing protein interactions

activate	conjugate	mediate
abolish	down-regulate	modify
accelerate	enhance	prevent
affect	inactivate	phosphorylate
alter	induce	regulate
amplify	infect	stimulate
assemble	inhibit	suppress
associate	interact	transactivate
bind	ligate	ubiquitinate
block	localize	upregulate

annotated corpus in molecular biology domain available to public (Ohta *et al*., 2000). We use the file GENIAcorpus-3.02.pos.txt in GENIA-v3.02. The pre-tagging strategy with morphology-based tagging improves the accuracy of *NN*, *JJ* and *NNS* remarkably, while the accuracy of other tags is not affected much. The overall accuracy is improved to 92.8%.

Mining verbs

First, we will mine as many verbs as possible from the corpus. The algorithm shown in Figure 1 is performed on the whole corpus and one more filtering rule is used, besides those in Table 3:

If the pattern has no verb tag, reject it.

With this rule, only patterns that have verbs are extracted. Here the threshold d is 10. It is comparatively high because we want to obtain very accurate verbs for the subsequent experiments. Totally 94 verbs are extracted as action words defining interactions, out of 367 different verbs in the sentences. Different tense verbs that have the same base form are counted as different ones. Among the extracted verbs, there are false positives which do not define interactions at all, such as 'prevent', 'affect', 'infect', 'localize'. Such words describe a relation between proteins, but do not semantically define the interactions. Our algorithm cannot perceive the difference. There are 16 such false positives. Hence the accuracy is 83.0%. Among the 273 eliminated words, there are no false negatives which still define interactions but are not extracted.

Table 7 shows some verbs obtained. These verbs and their variants, particularly the gerund and noun form (obtained from an English lexicon), are added into a filtering words list (*FWL* for short). For example, for verb '*inhibit*', its variants including '*inhibition*', '*inhibiting*', '*inhibited*' and '*inhibitor*' are added into *FWL*. At the current phase, we add all mined verbs into *FWL*, including false positives because these verbs are also helpful in understanding protein interaction networks. More verbs are listed at http://learn.tsinghua.edu.cn/homepage/2000315648/verbs.htm.

Extracting patterns

The pattern generating algorithm is performed on the whole corpus with *FWL*. The threshold d is 5 here. The rules in Table 3, plus the following rule, are applied.

If the action word of a pattern is not in FWL, reject it.

This rule guarantees that the main verb or noun in each pattern exactly describe protein interactions. The experiment runs on about 1200 sentences, with threshold $d = 5$, and 134 patterns are obtained (Fig. 4). Some of them are listed in Figure 3. More patterns can be found at http://learn.tsinghua.edu.cn/homepage/2000315648/pattern.htm.

Evaluating precision and recall rate

In this part, three tests are performed. The first test uses 383 sentences that include keyword *interact* or its variants. 293 of them are used to extract patterns and the rest are tested. The second one uses 329 sentences that contain the key word *bind* or its variants. 250 of them are used to generate patterns and the rest are tested. The third one uses 1205 sentences with all the keywords. 1020 are used to generate patterns, and the rest are tested. As described before, we do not exclude verbs such as 'prevent', 'affect', 'infect' and so on, therefore, relations between proteins defined by these verbs or nouns are thought to be interactions. Note that the testing and training sentences are randomly partitioned, and they do not overlap in all these tests. The results are shown in Table 8. *TP* is the number of correctly extracted interactions, $(TP + TN)$ is the number of all interactions in test sentences, and $(TP + FP)$ is the number of all extracted interactions. $F_{\beta=1}$ is defined as:

$$F_{\beta=1} = 2 * \text{recall} * \text{precision}/(\text{recall} + \text{precision})$$

Some matching examples are shown in Figure 5. Simple sentences as *sen* 1–3 are matched by only one pattern. But it is more common that several patterns may match one sentence at different positions, as in *sen4* and *sen5*. In the examples *sen6* and *sen7*, the same pattern matches repeatedly at different positions since we used a 'multiple matches' algorithm. More examples can be found at http://learn.tsinghua.edu.cn/home-page/2000315648/examples.htm.

DISCUSSION

We have proposed a new method for automatically generating patterns and extracting protein interactions. In contrast, our method outperforms the previous methods in two main aspects: first, it automatically mines patterns from a set of sentences whose protein names are identified. In the state of the art methods, patterns play an important role in extracting pathways. However, writing patterns for each verb one by one is impractical. Our method provides an approach to generate such patterns. A pattern can be extracted with certainty as long as it appears a sufficient number of times in the training set.

Pattern Count	Pattern Form	Word lists of pattern
1914	*PTN VBZ PTN*	* ;modifies promotes inhibits activates mediates blocks enhances forms ;* ;
758	*PTN VBZ IN PTN*	* ;interacts associates ;with within ;* ;
199	*PTN VBZ TO PTN*	* ;binds ;to ;* ;
99	*PTN VBZ IN PTN CC PTN*	* ;assembles interacts associates ;of in with from ;* ;and but ;* ;
94	*PTN VBN PTN*	* ;linked modified promoted activated stimulated regulated enhanced ;* ;
84	*PTN VBZ PTN CC PTN*	* ;assembles activates phosphorylates ubiquitinates prevents ;* ;and but ;* ;
52	*PTN VBN IN PTN*	* ;phosphorylated modified promoted ubiquitinated regulated activated induced mediated expressed stimulated localized ;with on by of upon ;* ;
26	*PTN VB IN PTN*	* ;interact associate stimulate catalyze ;with in of ;* ;
24	*NNS IN PTN CC PTN*	interactions activations ;with of between through from ;* ;and ;* ;
21	*PTN VBP PTN*	* ;modify activate recognize catalyze ;* ;
20	*PTN CC PTN VBP IN PTN*	* ;and ;* ;regulate interact associate ;of with ;* ;
16	*NN IN PTN TO PTN*	conjugation binding activation ;of ;* ;to ;* ;
13	*PTN VBD IN PTN*	* ;interacted associated co-localized ;with via by of ;* ;
6	*PTN VBZ PTN PTN CC PTN*	* ;assembles binds disrupts ubiquitinates activates ;* ;* ;and;* ;
5	*NN IN PTN IN PTN*	association interaction binding ;of ;* ;with within via through ;* ;

Fig. 4. Pattern examples extracted from about 1200 sentences. The star symbol denotes a protein name. Words for each component of a pattern are separated by a semicolon. Action words are not completely listed.

Table 8. The recall and precision experiments

Keyword	TP	TP+TN	TP+FP	Recall (%)	Precision (%)	$F_{\beta=1}$ (%)
Interact	66	82	78	80.5	84.6	82.5
Bind	58	71	70	81.7	82.8	82.2
All verbs	183	229	228	79.9	80.3	80.2

Given a small threshold d, even infrequent patterns could be extracted. The more the data, the better the patterns are. In essence, our method is a data-driven method.

Second, it is competent to process long and complicated sentences from full texts. The performance of most parsers falls sharply when processing long sentences. Some can only process a limited number of words in a sentence. In contrast to the previous methods (except Ono's method), our method based on dynamic programming is able to process such long sentences fast and efficiently.

In our method, a threshold d is used to control both the number and the generalization of patterns. It is meaningful to probe into the infrequent patterns filtered by a small threshold. For example, on the 293 sentences containing keyword 'interact' or its variants, patterns whose count equals one are shown in Figure 6. Some patterns are reasonable, such as '*PTN VBZ IN PTN IN PTN*' (protein$_1$ interacts with protein$_2$ through protein$_3$). Such patterns are rejected either because of insufficient training data or infrequently used expressions in natural language texts. Some are not accurate, such as '*NNS IN PTN PTN PTN*', because there must be a coordinating conjunction between the three continuous protein names, otherwise it will cause many errors. Some are even wrong, such as '*PTN NN PTN*' because there are never such segment 'protein$_1$ interaction protein$_2$' defining a real interaction between protein$_1$ and protein$_2$. Some patterns, such as '*PTN VBZ IN CC IN PTN*' which should be '*PTN VBZ IN PTN CC IN PTN*' (protein$_1$ interacts with protein$_2$ and with protein$_3$), are not precise because the last decision rule in Table 3 is used.

Nevertheless, these patterns can be filtered out by a small threshold. However, how to evaluate and maintain patterns becomes a real problem. For example, when the pattern generating algorithm is applied on about 1200 sentences, with a threshold $d = 0$, about 800 patterns are generated, most of which appeared only once. It is necessary to reduce such a large number of patterns. A MDL-based algorithm that measures the confidence of each pattern is under development. The patterns which cause many errors will be deleted. Furthermore, some similar patterns can be merged and those patterns that are not observed can potentially be generated from similar patterns.

Because our matching algorithm utilizes part-of-speech tags, and our patterns exclude adjectives (*JJ*), interactions defined by adjectives, such as '*inducible*' and '*inhibitable*', cannot be extracted correctly by our method currently. This can be illustrated by the example below, where the words in bold are protein names.

"*The **class II proteins** are expressed constitutively on B-cells and EBV-transformed B-cells, and are inducible by **IFN-gamma** on a wide variety of cell types.*"

In the sentence, interaction between *class II proteins* and *IFN-gamma* that is defined by adjective *inducible* (tagged as *JJ*) does not match any pattern.

To solve this problem, we are considering using word stemming and morpheme recognition to convert adjectives into their corresponding verbs with context.

*Sen1: Here, we show that **HIPK2** is regulated by a ubiquitin-like protein, **SUMO-1**.*
Pattern: PTN VBN IN PTN result: ***HIPK2*** regulated by ***SUMO-1***
*Sen2: Our studies show that **UBC9** binds to **TEL** exclusively through the **HLH** domain of **TEL**.*
Pattern: PTN VBZ TO PTN result: ***UBC9*** binds to ***TEL***
*Sen3: In the absence of **Mad2**, **BubR1** inhibits the activity of **APC** by blocking the binding of **Cdc20** to **APC**.*
Pattern: PTN VBZ PTN result: ***BubR1*** inhibits ***APC***
Pattern: NN IN PTN TO PTN result: binding of ***Cdc20*** to ***APC***
*Sen4: All proteins of this family have **Cdk**-binding and anion-binding sites, but only mammalian **Cks1** binds to **Skp2** and promotes the association of **Skp2** with **p27** phosphorylated on **Thr-187**.*
Pattern: PTN VBZ TO PTN result: ***Cks1*** binds to ***Skp2***
Pattern: NN IN PTN IN PTN result: association of ***Skp2*** with ***p27***
*Sen5: In the nucleus, **A1Up** co-localized with mutant **ataxin-1**, further demonstrating that **A1Up** interacts with **ataxin-1**.*
Pattern: PTN VBD IN PTN result: ***A1Up*** co-localized with ***ataxin-1***
Pattern: PTN VBZ IN PTN result: ***A1Up*** interacts with ***ataxin-1***
*Sen6: **MUL** interacts with **USP7** in vitro via their TDs to all of the previously identified TRAF family proteins, whereas the **TD of SPOP** interacts with **TRAF1**.*
Pattern: PTN VBZ IN PTN result: ***MUL*** interacts with ***USP7***
Pattern: PTN VBZ IN PTN result: ***TD of SPOP*** interacts with ***TRAF1***
*Sen7: Evidence is also provided that, in vivo, **E6** can interact with **p53** in the absence of **E6-AP** and that **E6-AP** can interact with **p53** in the absence of **E6**.*
Pattern: PTN VB IN PTN result: ***E6*** interact with ***p53***
Pattern: PTN VB IN PTN result: ***E6-AP*** interact with ***p53***

Fig. 5. Examples of protein interactions extracted from sentences. Words in bold are protein names. For every sentence, the matched patterns are listed, followed by the corresponding results.

Pattern Count	Pattern Form	Word lists of pattern
1	*NNS IN PTN PTN*	interactions ;with between ;* ;* ;
1	*PTN VB IN PTN IN PTN*	* ;interact ;with ;* ; through ;* ;
1	*PTN VBD IN PTN CC PTN*	* ;interacted ;with ;* ;and ;* ;
1	*PTN VBZ IN CC IN PTN*	* ;interacts ;with ;and ;with ;* ;
1	*PTN VBZ IN PTN IN PTN*	* ;interacts ;with ;* ;through;* ;
1	*PTN NN PTN*	* ;interaction ;* ;
1	*NNS IN PTN PTN CC PTN*	interactions ;between ;* ;* ;and or ;* ;
1	*NNS IN PTN PTN PTN*	interactions interaction ;with between ;* ;* ;* ;

Fig. 6. Some patterns whose count equals one are generated by our algorithm. More examples can be found at http://learn.tsinghua.edu.cn/homepage/2000315648/pattern_1.htm.

By analyzing the results, errors in the experiments can be classified into three categories: (1) protein name identification error. Although we use a dictionary-based tagging method, there are also errors because of wrong lexical items and lack of correct ones. (2) part-of-speech tagging errors. Incorrect tags directly cause a failure in extracting interactions. (3) match errors from the matching algorithm itself. We find that the current matching algorithm is not optimal and causes approximately one-third of total errors. This partially derives from the simple decision rules used in the matching algorithm. These rules may work well for some texts but partially fail for others because the natural language texts are multifarious. With these considerations, a more accurate and complicated matching algorithm is under development.

CONCLUSION

In this paper, a method for automatically generating patterns to extract protein–protein interactions is proposed and implemented. The method is capable of discovering verbs and patterns in biomedical texts. The algorithm is fast and able to process long sentences. Experiments show that a recall rate of about 80% and precision rate of about 80% are obtained. The approach is powerful, robust, and applicable to real and large-scale full texts.

Our future work will focus on pattern maintenance, and new matching algorithms.

ACKNOWLEDGEMENTS

The work was supported by Chinese Natural Science Foundation under grant No.60272019 and 60321002, the Canadian NSERC grant OGP0046506, CRC Chair fund, and the Killam Fellowship. We would like to thank Jinbo Wang and Daming Yao for their collaboration on the *PathwayFinder* system.

REFERENCES

Brill,E. (1995) Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguistics*, **21**(4), 543–565.

Cherry,J.M., Ball,C., Weng,S., Juvik,G., Schmidt,R., Adler,C., Dunn,B., Dwight,S., Riles,L., Mortimer,R.K. and Botstein,D. (1997) Genetic and physical maps of *Saccharomyces cerevisiae. Nature*, **387(6632 Suppl)**, 67–73.

Friedman,C., Kra,P., Yu,H., Krauthammer,M. and Rzhetsky,A. (2001) Genies: a natural-language processing system for the extraction of molecular pathways from journal articles.*Bioinformatics*, **17** (**suppl. 1**), S74–S82.

Hirschman,L., Park,J.C., Tsujii,J., Wong,L. and Wu,C.H. (2002) Accomplishments and challenges in literature data mining for biology. *Bioinformatics*, **18**, 1553–1561, December 2002.

Leroy,G. and Chen,H. (2002) Filling preposition-based templates to capture information from medical abstracts. In *Pacific Symposium on Biocomputing*, Hawaii, USA, Vol. 7, pp. 350–361.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Ng,S.K. and Wong,M. (1999) Toward routine automatic pathway discovery from on-line scientific text abstracts, *Proceedings of 10th International Workshop on Genome Informatics*, Tokyo, December 1999, pp. 104–112.

O'Donovan,C., Martin,M.J., Gattiker,A., Gasteiger,E., Bairoch,A. and Apweiler,R. (2002) High-quality protein knowledge resource: Swiss-Prot and TrEMBL. *Briefings Bioinform.*, **3**(3), 275–284.

Ohta,T., Tateishi,Y., Collier,N., Nobata,C. and Tsujii,J. (2000) Building an annotated corpus from biology research papers. *Proceedings of the COLING-2000 Workshop on Semantic Annotation and Intelligent Content*, Luxembourg, pp. 28–34.

Ono,T., Hishigaki,H., Tanigami,A. and Takagi,T. (2001) Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, **17**(2), 155–161.

Park,J.C., Kim,H.S. and Kim,J.J. (2001) Bidirectional incremental parsing for automatic pathway identification with combinatory categorical grammar. In *Proceedings of the Pacific Symposium Biocomputting*, Hawaii, USA, pp. 396–407.

Pustejovsky,J., Castano,J., Zhang,J., Kotecki,M. and Cochran,B. (2002) Robust relational parsing over biomedical literature: extracting inhibit relations. In *Proceedings of the seventh Pacific Symposium on Biocomputing (PSB 2002)*, pp. 362–373.

Salwinski,L., Miller,C.S., Smith,A.J., Pettit,F.K., Bowie,J.U. and Eisenberg,D. (2004) The database of interacting proteins: 2004 update. *NAR*, **32**, *Database issue*: D449–D451.

Smith,T.F. and Waterman,M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Temkin,J.M. and Gilder,M.R. (2003) Extraction of protein interaction information from unstructured text using a content-free grammar, *Bioinformatics*, **19**, 2046–2053.

Thomas,J., Milward,D., Ouzounis,C., Pulman,S. and Carroll,M. (2000) Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of the Pacific Symposium on Biocomputing*, Hawaii, USA, Jan 2000, pp. 541–551.

Wong,L. (2001) A protein interaction extraction system. In *Proceedings of Pacific Symposium on Biocomputing 2001*, Hawaii, January 2001, pp. 520–530.

Xenarios,I., Rice,D.W., Salwinski,L., Baron,M.K., Marcotte,E.M. and Eisenberg.D. (2000) DIP: The database of interacting proteins. *NAR* **28**, 289–291.

Yakushiji,A., Tateisi,Y., Miyao,Y. and Tsujii,J. (2001) Event extraction from biomedical papers using a full parser. In *Proceedings of the sixth Pacific Symposium on Biocomputing* (PSB 2001), Hawaii, USA, pp. 408–419.

Yao,D., Wang,J., Lu,Y., Noble,N., Sun,H., Zhu,X., Lin,N., Payan,D.G., Li,M. and Qu,K. (2004) Pathway-Finder: paving the way towards automatic pathway extraction. In Yi-Ping Phoebe Chen (eds), *Bioinformatics 2004: Proceedings of the 2nd Asia-Pacific Bioinformatics Conference (APBC)*, **29** volume of *CRPIT*, pp. 53–62, Dunedin, New Zealand, January 2004. Australian Computer Society.