Discovering patterns to extract protein-protein interactions from full biomedical texts

[*]Minlie Huang[1], [+]Xiaoyan Zhu[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
[3]Bioinformatics Laboratory, School of Computer Science, University of Waterloo, N2L 3G1, Canada
[*]huangml00@mails.tsinghua.edu.cn
[+]zxy-dcs@tsinghua.edu.cn
[++]mli@uwaterloo.ca

Abstract

Although there have been many research projects to extract protein pathways, most such information still exists only in the scientific literature, usually written in natural languages and defying data mining efforts. We present a novel and robust approach for extracting protein-protein interactions from the 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 protein name dictionary, our system achieves a recall rate of about 80.0% and a precision rate of about 80.5%.

1 Introduction

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

How to extract such 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 used. For example, a general full parser with grammars applied to the biomedical domain was used to extract interaction events by filling sentences into argument structures in (Yakushiji *et al*., 2001). No recall or precision rate 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 the target verbs, and then it scans the left and right neighborhood of the verb respectively. The lexical and grammatical rules of CCG are even more complicated than those of a general CFG. The recall and precision rates of the system were reported to be 48% and 80%, respectively. Another full parser utilizing a lexical analyzer and context free grammar (CFG), extracts protein, gene and small molecule interactions with a recall rate of 63.9% and a precision rate of 70.2% (Temkin *et al*., 2003). Similar methods such as preposition-based parsing to generate templates were proposed (Leroy and Chen, 2002), processing only 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, all the methods are inherently complicated, requiring many resources, and the performances are not satisfactory. Some methods only focus on several special verbs.

Another popular approach uses pattern matching. As an example, a set of simple word patterns and part-of-speech rules were manually coded, for each verb, to extract special kind of interactions from abstracts (Ono *et al*., 2001). The method obtains a recall rate of about 85% and a precision rate of about 94% for *yeast* and *Escherichia coli*, which is the best among all reported results. 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; Marcotte *et al.*, 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 fairly among various methods (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 surprisingly robust method to discover patterns to extract interactions between proteins. It is based on dynamic programming (DP). In the realm of homology search between protein or DNA sequences, 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, the similar parts in sentences could be extracted as patterns. Compared with the previous methods, our proposal is different in the following ways: Firstly, it processes full biomedical texts, rather than only abstracts. Secondly, it automatically mines verbs for describing protein interactions. Thirdly, this method automatically discovers patterns from a set of sentences whose protein names are identified, rather than manually creating patterns as most previous methods. Lastly, our method has low time complexity. It is able to process very long sentences. In contrast, for any full or partial parsing method, it is time- and memory-consuming to process long sentences.

2 Method

2.1 Alignment algorithm

Suppose we have two sequences $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$ which are defined over the alphabet $\Sigma = \{a_1, a_2, ..., a_l, '-'\}$. Each a_i is called as a character, and '-' denotes a white-space or a gap. We want to 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:

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

$$F(i,0) = 0, F(0,j) = 0,\ x_i, y_j \in \Sigma \quad (1b)$$

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

$$s(a,b) = \log[\, p(a,b) / (p(a) * p(b))] \quad (2)$$

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 $p(a)$ and $p(a,b)$ can be easily estimated by calculating appearance frequencies for each pair with pre-aligned training data.

Note that the calculation of scores for a gap will be different. In formula (2), when a or b is a gap, the scores can not be directly estimated by the formula because of two reasons: 1) 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; 2) 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 a sequence of $X = (x_1, x_2, ..., x_n)$ with no gaps, the linear penalty is as follows:

$$\gamma(n) = \sum_{i=1}^{n} s('-', x_i) \quad (3)$$

For sequence X of length n and sequence Y of length m, totally $(n+1)*(m+1)$ scores will be calculated by applying equation (1a-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.

In our method, the alphabet consists of three kinds of tags: 1) part-of-speech tags, as those used by Brill's tagger (Brill *et al*., 1995); 2) tag *PTN* for protein names; 3) tag *GAP* for a gap or white-space. Gap penalties for main tags are shown in Table 1.

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

Table 1. Gap penalties for main tags

2.2 Pattern generating algorithm

For our problem, a data structure called sequence structure, instead of a flat sequence, is used. Sequence structure 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.

Similarly, we also use another data structure called pattern structure 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 in the training corpus. With the structure, the pattern generating algorithm is shown in Figure 1. The filtering rules are listed in Table 2.

Note that a threshold *d* is used in the algorithm. If a pattern appears less than *d* times in the corpus, it will be discarded; otherwise those infrequent patterns will cause many matching errors. Through adjusting this parameter, generalization and usability of patterns can be controlled. The larger the threshold is, the more general and accurate patterns are.

Tags like *JJ* (adjective) and *RB* (adverb) are too common and can appear at 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 to pattern generation. Therefore, just as the first step in our algorithm shown in Figure 1, we remove directly the 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 2 are adapted. Verb or noun tags define interactions between proteins, thus they are indispensable for a pattern, 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. Actually more rigid or looser filtering rules than those shown in Table 2 can be applied to meet special demands, which will affect the forms of patterns.

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 neighborhood of the tag in a pattern, reject the pattern.*

Table 2. Filtering rules.

2.3 Pattern matching algorithm

Because one pattern possibly matches a sentence at different positions, we have to explore an algorithm that is able to find out multiple matches.

Input: an integer *d*,
a sequence set $S = (s_1, s_2, ..., s_n)$
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
   a) Do local alignment for s_i and s_j. Aligned output is X_a and Y_b;
   b) Extract the identical characters at the same positions in X_a and Y_b as pattern *p*. Add the corresponding word indices to pattern structure;
   c) Judge whether *p* is legal, using the filtering rules. If it is illegal, go to step 2;
   d) 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*.

Figure 1. Pattern generating algorithm. Time complexity is $O(n^2)$ in the corpus size *n*.

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.

Suppose that $X=(x_1, x_2, ..., x_n)$ is the sequence of tags for a sentence in which we look for multiple matches, and $Y=(y_1, y_2, ..., y_m)$ is a pattern. We still use a score matrix *F*, while the recurrence, defined by formulas (4a-b), is different from that of pattern generating algorithm. Formula (4a) only allows matches to end when they have score at least *T*.

$$F(0,0) = 0$$
$$F(i,0) = \max \begin{cases} F(i-1,0) \\ F(i-1, j) - T,\ j = 1,2, \cdots m \end{cases} \quad (4a)$$

$$F(i, j) = \max \begin{cases} F(i,0), \\ F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + s(x_i, '-') \\ F(i, j-1) + s('-', y_j) \end{cases} \quad (4b)$$

The total score of all matches is obtained by adding an extra cell to the matrix, *F(n+1,0)*, using (4a). By tracing back from cell (n+1,0) to (0,0), the individual match alignments will be obtained. Threshold *T* should not be identical for different patterns. Threshold *T* is calculated as follows:

$$T = \eta \sum_{i=1}^{m} s(y_i, y_i) \quad (5)$$

where η is a factor, in our method we take η=0.5. The right hand of formula (5) is the maximum score when a pattern matches a sentence perfectly.

A match is accepted only when three conditions are satisfied: 1) a pattern has a local optimal match with the sentence; 2) words in matching part of the sentence can be found in the word set of the pattern; 3) decision rules are satisfied.

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

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

To show details how well a pattern matches a sentence, a measurement data structure is defined, which is formalized as a vector. It will be referred to as *mVector*:

$$mVector = (cLen, cMatch, cPtn, cVb) \quad (6)$$

where *cLen* is the length of a pattern; *cMatch* is the number of matched tags; *cPtn* is the number of protein name tag (*PTN*) skipped by the alignment in the sentence; *cVb* is the number of skipped verbs. Based on the structure, decision rules shown in Table 3 are used in the pattern matching. 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*.

Input: two parameters *P* and *V*
1. *If cMatch ≠ cLen, reject the match;*
2. *if cPtn > P, reject the match;*
3. *if cVb > V, reject the match;*

Table 3. Decision rules.

3 System overview

Our system uses the framework of *Pathway-Finder* (Yao *et al*., 2004). It consists of several modular components, as 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 which will be explained with details in the next section. Patterns generated at the training phase are stored in the pattern database.

For an input sentence, firstly some filtering rules are adapted to remove useless expressions at the pre-processing phase. For example, remove citations, such as '[1]', and listing figures, such as '(1)'. 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 *et al*., 1995), where the tag of protein names is changed to tag *PTN*. Last, since a sequence of tags is obtained, it can be added into the corpus at the training phase or it can be used by the matching algorithm at the testing phase.

Because the pattern acquisition algorithm is aligning sequences of tags, the accuracy of part-of-speech tagging is crucial. However, Brill's tagger only obtained overall 83% accuracy for biomedical texts. This is because biomedical texts contain many unknown words. Here we propose a simple and effective approach called pre-tagging strategy to improve the accuracy, just as the method used by (Huang *et al.*, 2004).

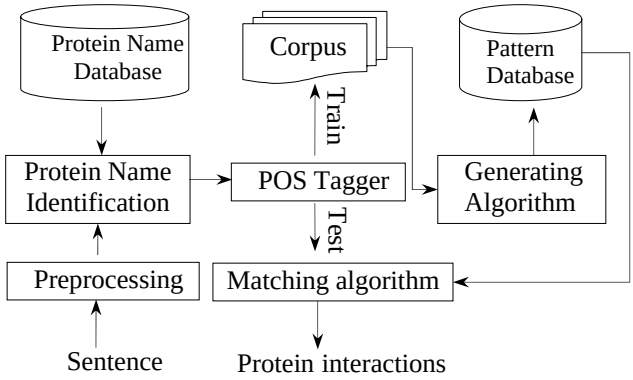

Figure 3. Architecture of our system.

4 Results

Our evaluation experiments are made up of three parts: mining verbs for patterns, extracting patterns and evaluating precision and recall rates.

4.1 Mining verbs

The algorithm shown in Figure 1 is performed on the whole corpus and one more filtering rule as follows, is used, besides those in Table 2:

If the pattern has no verb tag, reject it.

With this rule, only patterns that have verbs are extracted. Here the threshold *d* is set to 10 to obtain high accurate verbs for the subsequent

experiments. Totally 94 verbs are extracted from 367 verbs for describing interactions. Note that different tense verbs that have the same base form are counted as different ones. There are false positives which do not define interactions semantically at all, such as 'affect', 'infect', 'localize', amounting to 16. Hence the accuracy is 83.0%.

These verbs and their variants, particularly the gerund and noun form, (obtained from an English lexicon) are added into a list of filtering words, which is named as *FWL* (Filtering Word List). For example, for verb '*inhibit*', its variants including '*inhibition*', '*inhibiting*', '*inhibited*' and '*inhibitor*' are added into *FWL*. At the current phase, we add all verbs into *FWL*, including false positives because we think these verbs are also helpful to understand pathway networks between proteins.

4.2 Extracting patterns

Pattern generating algorithm is performed on the whole corpus with *FWL*. The threshold *d* is 5 here. The filtering rules in Table 2, plus the following rule, are applied.

If a pattern has any verb or noun that is not in FWL, reject it.

This ensures that the patterns have a good form and all their words are valid. In other word, this rule guarantees that the main verbs or nouns in every pattern exactly describe protein interactions. The experiment runs on about 1200 sentences, with threshold *d=5*, and 134 patterns are obtained. Some of them are listed in Figure 4.

4.3 Evaluating precision and recall rates

In this part, three tests are performed. The first test uses 383 sentences that only contain keyword *interact* and its variants. 293 of them are used to extract patterns and the rest are tested. The second one uses 329 sentences that only contain key word *bind* and its variants. 250 of them are used to generate patterns and the rest are tested. The third one uses 1205 sentences with all keywords, where 1020 are used to generate patterns, the rest for test. As described before, we do not exclude those verbs such as '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 4. Some matching examples are shown in Figure 5. Simple sentences as *sen1-2* are matched by only one pattern. But it is more common that several patterns may match one sentence at different positions, as in *sen3-4*. In examples *sen5*, the same pattern matches repeatedly at different positions since we used a 'multiple matches' algorithm.

Keywords	Recall	Precision	F-score
Interact	80.5%	84.6%	82.5%
Bind	81.7%	82.8%	82.2%
All verbs	79.9%	80.3%	80.2%

Table 4. The recall and precision experiments.

5 Discussion

We have proposed a new method for automatically generating patterns and extract 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; second, it is competent to process long and complicated sentences from full texts.

In our method, a threshold *d* is used to control both the number of patterns and the generalization power of patterns. Although infrequent patterns are filtered by a small threshold, a glance to these patterns is meaningful. For example, on 293 sentences containing keyword 'interact' and its variants, patterns whose count equals one are shown in Figure 6. Among the results, some are reasonable, such as '*PTN VBZ IN PTN IN PTN* ' (protein$_1$ interacts with protein$_2$ through protein$_3$). These kinds of patterns are rejected because of both insufficient training data and infrequently used expressions in natural language texts. Some patterns 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 patterns 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 filtering rule in Table 2 is used.

Nevertheless, these patterns can be filtered out by the 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*, approximate 800 patterns are generated, most of which appeared only once in the corpus. It is necessary to reduce such large amount of patterns. A MDL-based algorithm that measures the confidence of each pattern and maintains them without human intervention is under development.

Because our matching algorithm utilizes part-of-speech tags, and our patterns do not contain any adjective (*JJ*), interactions defined by adjectives, such as '*inducible*' and '*inhibitable*', cannot be extracted correctly by our method currently.

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 in within ;* ;
402	*NN IN PTN CC PTN*	interaction association activation modification degradation ;between with of from by ;* ;and or but ;* ;
270	*PTN NN IN PTN*	* ;interaction complex conjugation modification association ;with of on by in within between ;* ;
199	*PTN VBZ TO PTN*	* ;binds; to ;* ;
99	*PTN VBZ IN PTN CC PTN*	* ;assembles interacts associates; of in with from ;* ;and but ;* ;
16	*PTN CC PTN NN IN PTN*	* ;and or ;* ;interaction conjugation complex ubiquitination degradation modification activation recognition ;between of with by ;* ;
5	*PTN VBP IN PTN*	* ;interact ;with ;* ;

Figure 4. Pattern examples. The star symbol denotes a protein name. Words for each component of a pattern are separated by a semicolon. For simplicity, words in a pattern are partially listed.

*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: **SPB** association of **Plo1** is the earliest fission yeast mitotic event recorded to date.*
Pattern: PTN NN IN PTN — result: ***SPB*** association of ***Plo1***
*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: 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***

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

Pattern Count	Pattern Form	Word lists of pattern
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 PTN*	interactions interaction ;with between ;* ;* ;* ;

Figure 6. Some patterns whose count equals one are generated by our algorithm. 293 sentences containing keyword 'interact' and its variants are used in the training.

This can be demonstrated by the following sentence, where 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 this sentence, interaction between ***class II proteins*** and ***IFN-gamma*** is defined by an 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.

By analyzing our experimental results, 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.

6 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 a precision rate of about 80% are obtained. The approach is powerful, robust, and applicable to real and large-scale full texts.

7 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 that thank Jinbo Wang and Daming Yao for their collaboration on the *PathwayFinder* system.

References

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

Cherry,JM, Ball,C, Weng,S, Juvik,G, Schmidt,R, Adler,C, Dunn,B, Dwight,S, Riles,L, Mortimer,RK, 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:S74–82.

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

Huang,M., Zhu,X. and Li,M. (2004) A new method for automatic pattern acquisition to extract information from biomedical texts. In the *Seventh International Conference on Signal Processing*, August, Beijing, China. Accepted.

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

Marcotte,EM, Xenarios,I., and Eisenberg,D. (2001) Mining literature for protein-protein interactions. *Bioinformatics*, **17(4)**, 359–363.

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,MJ, Gattiker,A., Gasteiger,E., Bairoch,A. and Apweiler,R. (2002) High-quality pro-tein knowledge resource: Swiss-Prot and TrEMBL. *Briefings in Bioinformatics 2002 Sep;* **3(3)**, 275-284.

Ohta,T., Tateishi,Y., Collier,N., Nobata,C., and Tsujii,J. (2000) Building an annotated corpus from biology research papers. *Proc. 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,JC, Kim,HS, and Kim,JJ (2001) Bidirectional incremental parsing for automatic pathway identify-cation with combinatory categorical grammar. In *Proceedings of the Pacific Symposium Biocom-putting*, 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,CS, Smith,AJ, Pettit,FK, Bowie,JU, Eisenberg,D (2004) The database of interacting proteins: 2004 update. *NAR* **32** *Database issue*: D449-51.

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

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, *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., Eisenberg.D. (2000) DIP: The data-base of interacting proteins. *NAR* **28**, 289-91.

Yakushiji,A., Tateisi,Y., Miyao,Y., 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., Qu,K. (2004) Pathway-Finder: paving the way towards automatic pathway extraction. In Yi-Ping Phoebe Chen, ed., *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.