A new alignment algorithm to identify definitions corresponding to abbreviations in biomedical text

Yun Xu, ZhiHao Wang, YuZhong Zhao
Anhui Province–MOST Co-Key Laboratory of High Performance Computing and Its Application
University of Science and Technology of China, Hefei, Anhui 230027, P. R. China
xuyun@ustc.edu.cn, {wangzhh, zyzh}@mail.ustc.edu.cn

Yu Xue
School of Life Science
University of Science and Technology of China, Hefei, Anhui 230027, P. R. China
yxue@mail.ustc.edu.cn

Abstract

The exploding growth of the biomedical literature presents many challenges for biological researchers. One such challenge is from the use of a great deal of abbreviations. Extracting abbreviations and their definitions accurately is very helpful to biologists and also facilitates biomedical text analysis. Among existing approaches, text alignment algorithms are simple, effective and require no training data. However, state of the art alignment algorithms could not identify the definitions of irregular abbreviations (e.g., <CNS1, cyclophilin seven suppressor>). We propose an algorithm analogous to pairwise sequence alignment, in which it is given a penalty score if there are two unmatched characters separately from the abbreviation and definition, and in this way some irregular abbreviations are found.

1. Introduction

The volume of published biomedical papers is expanding at an increasing rate each year. With biomedical knowledge expanding so quickly, it is very challenging for biologists to keep up to date with their own field of biomedical research. Thus, an automatic method for biomedical knowledge text mining is urgently needed [1, 2]. In biomedical text mining, one special issue is the exploding use of new abbreviations [3]. It would be a great help for literature retrieval to collect these abbreviations automatically. Furthermore, other text mining tasks could be done more efficiently if all the abbreviations for an entity could be mapped to a single term representing the concept [2]. Generally, an abbreviation is a short form of a word or phrase called "definition" or "long form". Our task is to identify <"short form", "long form"> pairs where there exists a mapping from characters in the short form to characters in the long form [4].

Existing approaches fall into four broad categories: statistically based, rule based, text alignment, and machine learning based. Statistically based approaches always tend to extract abbreviations that appear frequently in biomedical text, and demand a large corpus for the statistics. Zhou *et al.* [5] created an abbreviation database ADAM that analyzed statistical information about collocations of the type "long-form (abbreviation)" in MEDLINE, and Okazaki N and Ananiadou [6] built an abbreviation dictionary from the whole MEDLINE. Although Statistically based methods show a high precision, they would not find rare abbreviations, and need a great deal of time and effort.

Rule based approaches attempt to use the best recognition rule, and good rules would result in good results. Pustejovsky *et al.* [4] presented a regular expression algorithm based on hand-built regular expressions, and syntactic information was considered to identify boundaries of noun phrases. Ao and Takagi [7] constructed a system called ALICE based on heuristic pattern-matching rules. Yu *et al.* [8], Park and Byrd [9] also put forward their own pattern matching rules separately. The shortcoming for these rule based approaches is that the performance of them is determined by the completeness of the rules.

Machine learning based approaches generally comprise of a learner and a predictor, and fit in with all kinds of biomedical text by learning. Chang *et al.* [10] presented a method for identifying abbreviations using supervised machine learning. Generally speaking, machine learning based approaches depend on the learning model and the training

0-7695-3090-7/08 $25.00 © 2008 IEEE
DOI 10.1109/WKDD.2008.53

data, and require a lot of labor and time.

Text alignment based approaches always try to find the optimal alignment between the definition and abbreviation. Schwartz and Hearst [11] presented a simple algorithm for identifying the definitions of abbreviations with only two indices, lIndex for the long form, and sIndex for the short form. The two indices are initialized to point to the end of their respective strings. For each character sIndex points to, lIndex is decremented until a matching character is found. Chang *et al.* [10] and Taghva and Gilbreth [12] utilized the Longest Common Subsequence algorithm in their methods. However, the state of the art alignment algorithms can not find irregular abbreviations.

In this paper a dynamic programming algorithm analogous to pairwise sequence alignment [13, 14] is applied. For the comparison of biological sequences, Needleman and Wunsch [13] and Smith and Waterman [14] both succeeded in getting a sub-optimize result based on a dynamic programming algorithm. Then we improve the algorithm to make it applicable to the identification of definitions corresponding to abbreviations. In the alignment between the abbreviation and definition, gap insertions are allowed in abbreviations, and forbidden in definitions. It is given a penalty score if there are two unmatched characters separately from the abbreviation and definition, and in this way some irregular abbreviations are found.

2. Methods

To find the <short form, long form> pairs can be divided into two sub-tasks. One sub-task is abbreviation recognition, and the other is definition identification. Our work is mainly for the second task. The second sub-task can also be divided into definition searching and definition identification. The flowchart of our work is shown in Figure 1.

2.1. Abbreviation Recognition

For abbreviation recognition, we mainly use the recognition method presented by Park and Byrd [9]. Their method is based on a series of rules: the feature of the abbreviation (or short form) and the syntactic cues in the contexts. The feature of the abbreviation includes: its first character is alphabetic or numeric; it contains at least one letter; its length is between 2 and 10; it contains at most two words.

The syntactic cues we take into account include:

1. long form (short form) or long form [short form]
2. short form (long form) or short form [long form]
3. short form = long form
4. long form = short form
5. short form, ***or*** long form

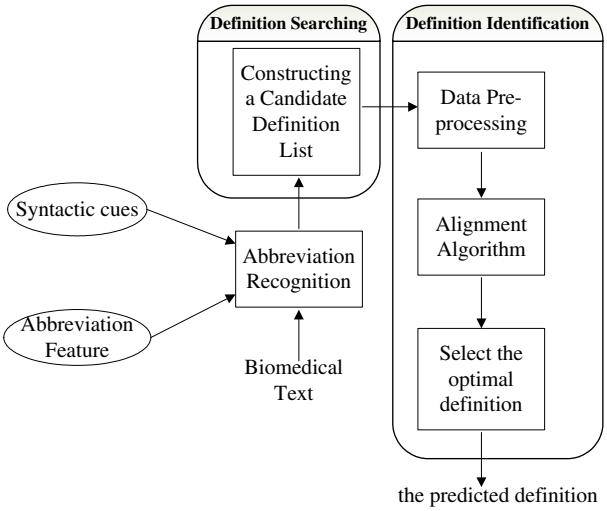

Figure 1. This is the flowchart of extracting abbreviations and their corresponding definitions.

6. long form, ***or*** short form
7. short form. . . ***stands/short/acronym***. . . long form
8. long form, short form ***for short***

In practice, most abbreviations appear with parentheses, i.e. protein kinase C (PKC). We use the similar method for abbreviation recognition as most researchers, and only consider (1) and (2). For pattern (2), the short form is the one or two words before the left parenthesis, and the long form is just the expression inside the parentheses. For pattern (1), the short form is inside the parentheses, but the long form is not easy to be found and it need to be searched for before the left parenthesis.

2.2. Definition Searching

Having recognized the abbreviation, the next step is to find the candidate definition. The candidate definition appears in the same sentence as the abbreviation, and it can be searched for within a search space. The size of the search space is the sum of the maximum length of a definition (the number of the words in the definition) and the maximum offset (the longest distance of a definition from an abbreviation). In our work, the offset is ignored and we consider only definitions adjacent to the abbreviations (as most people do). Park and Byrd [9] analyzed about 4500 abbreviations and their definitions, and then they decided that, for relatively short abbreviations (from two to four characters), the maximum length of a definition should not be greater than twice the abbreviation length (the number of the characters in an abbreviation); for long abbreviations (five or

more characters), the definition should not be longer than the abbreviation length plus 5. Thus, we refer to their work for the maximum length of a definition DEF of an abbreviation ABBR:

$$Max.|DEF| = min\,(\,|ABBR| + 5\,,\,|ABBR| * 2\,) \quad (1)$$

where Max.|DEF| is the maximum length of a definition, and |ABBR| is the number of the characters in an abbreviation.

Then a candidate definition list is constructed from the search space, and the possible definition is just one item of it. The list-constructing algorithm is described below.

Procedure 1 Construct the Candidate Definition List CDL

1: Initiate an empty candidate definition list CDL;
2: Num = the number of words from the beginning of the sentence which contains the abbreviation to the left parenthesis;
3: if (Num $<$ Max.|DEF|) {
SearchSpaceString = the string from the beginning of the sentence to the left parenthesis;
} else {
SearchSpaceString = the string that contains Max.|DEF| words before the left parenthesis;
}
4: WordNum = min (Num, Max.|DEF|);
5: for (N = 0; N $<$ WordNum; N++) {
CandidateDef = SearchSpaceString with the leftmost N words deleted;
insert CandidateDef into CDL; }

2.3. Definition Identification

Now we have a candidate definition list. Each time we retrieve an item from the list, and align it with the abbreviation employing our alignment algorithm. Afterward we select the optimal definition.

2.3.1 Data Preprocessing

Usually a definition is abbreviated with a new addition of a special character (e.g., <Myo3/5p, Myo3p and Myo5p>), and the lowercase letter from a definition may be changed into its corresponding capital letter. Before we identify the definition corresponding to an abbreviation, some data preprocessing steps must be taken. We delete the character that is neither alphabetic nor numeric in the abbreviation and change all capital letters in both the abbreviation and definition into their corresponding lowercase letters. In order to differentiate between the space inserted in the alignment algorithm and the space between words of the definition, we replace the space between words of the definition with the character '\s'.

2.3.2 Alignment Algorithm

The definition identification is a process of comparison between the abbreviation and definition. In the process, the smallest unit of comparison is a pair of characters, one from the abbreviation, and the other from the definition. All possible comparisons are made from the smallest unit while allowing gap insertions in the abbreviation. Among the comparisons the definition with the best match is chosen as the correct definition. The best match can be defined as the largest alignment score of characters of the definition that can be matched with those of the abbreviation.

The largest alignment score can be determined by representing in a two-dimensional array, all possible pair combinations that can be constructed from the abbreviation and definition, A and D, being compared. A[i] is the ith character of the abbreviation string and D[j] is the jth character of the definition string. A[i] and D[j] represent the rows and the columns of the two-dimensional array SCORE respectively. Then the cell, SCORE[i][j], represents a pair combination that contains A[i] and D[j].

With the above definition of A[i], D[j] and SCORE[i][j], now what we need to do is to get the largest value of SCORE[i][j], which represents the best match. Then dynamic programming is used to compute each cell value of SCORE. Unlike the solutions of Needleman and Wunsch [13] and Smith and Waterman [14], gap insertions are allowed in abbreviations but forbidden in definitions in our algorithm, so SCORE[i][j] is determined by SCORE[i][j-1], SCORE[i-1][j-1] and the alignment of A[i] and D[j], and not by SCORE[i-1][j]. The below is the recursion equation for computing the largest value of SCORE[i][j].

Firstly the initial value is assigned:

SCORE[i][j] =0 if i=0 or j=0;

Then, we have

$$SCORE[i][j] = \max_{\substack{0<i\leq length(A)\\ 0<j\leq length(D)}} \begin{cases} SCORE[i-1][j-1] + w(A[i], D[j]) \\ SCORE[i][j-1] \end{cases} \quad (2)$$

where the w(A[i],D[j]) is defined as:

$$w(A[i], D[j]) = \begin{cases} 2, & if\ A[i] = D[j]\ and\ D[j]\ is\ the\ first\ character\ of\ one\ word\ in\ the\ definition; \\ 1, & if\ A[i] = D[j]\ and\ D[j]\ is\ not\ the\ first\ character\ of\ one\ word\ in\ the\ definition; \\ -10, & if\ A[i]\ != D[j] \end{cases} \quad (3)$$

After the matrix SCORE is filled, SCORE[length(A)][length(D)] is just the largest alignment score, the score of the best match. Knowing the largest alignment score is not enough, we need to get the best match pathway by traceback.

The best match pathway can be obtained by beginning at the terminals of both strings (i=length(A), j=length(D))

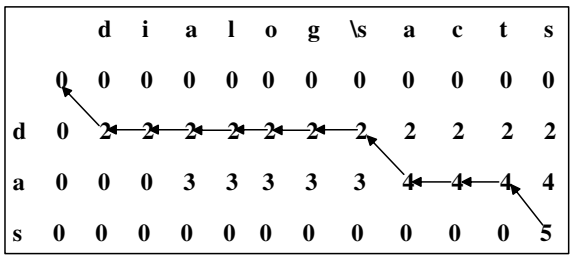

Figure 2. For example, the definition is "Dialog Acts", and the abbreviation is "DAs". All the arrows form the best match pathway.

and proceeding row by row toward the origins. The traceback algorithm checks if SCORE[i][j] is obtained from SCORE[i-1][j-1]. If yes, A[i] is identical to D[j], and both i and j are decremented. If not, a space is inserted before the ith character of the abbreviation, and only j is decremented. The process is repeated until all cells in the matrix SCORE have been operated upon and i=j=0. The operation of successive summations of cell values is illustrated in Figure 2.

2.3.3 Select the optimal definition

From the candidate definition list CDL, we get at most Max.|DEF| candidate definitions, each of which corresponds to an alignment score provided by the alignment algorithm. Despite the alignment score, it is not enough to determine the optimal definition. For example,

1. In the text "little is known, however, about how such dialog acts (DAs) can be automatically classified in truly natural conversation", "DAs" will be recognized as an abbreviation. The optimal definition is "dialog acts", but "dialog acts", "such dialog acts", "how such dialog acts" and "about how such dialog acts" will have the same alignment score;
2. In the text "the mutations map across most of the Bicoid protein, with some located within the DNA-binding domain (homeodomain)", "homeodomain" will be recognized as an abbreviation wrongly. Then the alignment algorithm will identify the string "with some located within the DNA-binding domain" as its definition. In this alignment, "within the DNA-binding" in the definition is unmatched (Figure 3 illustrates what "unmatch" means). With too many unmatched words in the middle of the definition this abbreviation must be thrown away;
3. In the text "a questionnaire was sent to them as well as to 246 physicians who had residency-level teaching responsibilities but who were not named (controls)", "controls" will be recognized as an abbreviation wrongly. The alignment algorithm will identify the string "physicians who had residency-level teaching responsibilities but who were not named" as its definition. In this alignment, "but who were not named" in the definition is unmatched. With too many unmatched words in the end of the definition this abbreviation must be thrown away;

So we propose a new concept "the redundant word penalty":

Definition 1 *The redundant word penalty is a penalty against the candidate definitions having several continuous unmatched words.*

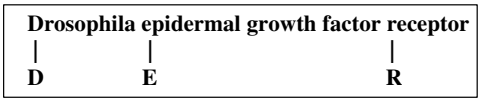

Figure 3. This is an alignment for <DER, Drosophila epidermal growth factor receptor>. In the alignment, "growth" and "factor" in the definition are unmatched. Adjacent to each other, they are called "continuous unmatched words". The number of the continuous unmatched words is 2.

The penalty depends on the number of the continuous unmatched words in the candidate definition (Figure 3). If the number is small, the penalty is low, otherwise the penalty is high. One unmatch word often appears in true definitions, for example, for the pair <FMDV, foot and mouth disease virus>, there is only one unmatched word "and" in the definition. The penalty should be very low in this case. Based on the analysis, the redundant word penalty (RWP) is divided into the beginning word penalty (BP, a low penalty) and the extended word penalty (EP, a high penalty). In N continuous unmatched words, the first C words are given a penalty score, BP for each word, and the other N-C words are given another penalty score, EP for each word. Thus, the equation of RWP is as follows:

$$RWP = C * BP + EP * (N - C) \quad (4)$$

There are three cases that the redundant word penalty occurs:

1. The first character of the abbreviation does not match the first word of the candidate definition (RWP1);
2. Two adjacent characters in the abbreviation match two separated words in the candidate definition respectively (RWP2);
3. The last character of the abbreviation does not match the last word of the candidate definition (RWP3).

For example, for the alignment <DER, Drosophila epidermal growth factor receptor> in Figure 3, RWP1, RWP2, and RWP3 are computed as follows:

1. Because the first character "D" of the abbreviation matches the first word "Drosophila" of the definition, RWP1=0;
2. For RWP2, any two adjacent characters in the abbreviation must be considered. "D" and "E" match two adjacent words "Drosophila" and "epidermal" separately, so RWP2("DE")=0; "E" and "R" match two separated words "epidermal" and "receptor" separately, and the number of the continuous unmatched words is 2, so RWP2("ER") = C*BP + EP*(2-C). In sum, RWP2 = RWP2("DE") + RWP2("ER");
3. Because the last character "R" of the abbreviation matches the last word "receptor" of the definition, RWP3=0.

The three cases may appear in the same alignment, so the total redundant word penalty (TotalRWP) is:

$$\texttt{TotalRWP} = \texttt{RWP1} + \texttt{RWP2} + \texttt{RWP3} \quad (5)$$

Then for each alignment, we have

$$\texttt{total score=the alignment score-TotalRWP} \quad (6)$$

At last the optimal definition can be selected from the candidate definition list by selecting the largest total score. If the total score of the optimal definition is larger than the predefined cutoff score, the optimal definition is identified.

2.4. Evaluating Performance

We use the harmonic mean (F-measure) of precision (accuracy) and recall (coverage) that are commonly used in the field to evaluate our results. The precision measures the number of correct <short form, long form> pairs in the answer file over the total number of the pairs in the answer file, and the recall measures the number of correct pairs in the answer file over the total number in the key file. With "TP" labeling true positives, "FP" the false positives and "FN" the false negatives, the measures are:

$$\begin{aligned} \texttt{Precision} &= \frac{TP}{TP+FP} \\ \texttt{Recall} &= \frac{TP}{TP+FN} \\ \texttt{F-measure} &= \frac{2 * Precision * Recall}{Precision + Recall} \end{aligned} \quad (7)$$

2.5. Scoring scheme

Before our evaluation, the w(A[i],D[j]), the redundant word penalty, the cutoff score and the variable C (in Equation (4)) need to be adjusted. They can be regarded as four parameters, and different value for every parameter will bring different results. So it is necessary to find the best value of them to make our algorithm perform well.

Here we provide a simple method to adjust the parameters. We first give the four parameters an initial value separately, and then we search for the best value for each of the four parameters to maximize F-measure when we keep the other three unchanged. In the procedure the gold standard DEVELOPMENT corpus [15] is used.

For w(A[i],D[j]), we need to adjust the penalty score that A[i] differs from D[j]. We run our algorithm against all possible values of the penalty score for the maximum F-measure. The relation between F-measure and the penalty score is illustrated in Figure 4. From the figure we know the maximum F-measure is got when the penalty score is -7. Then the penalty score is assigned the value, -7. In

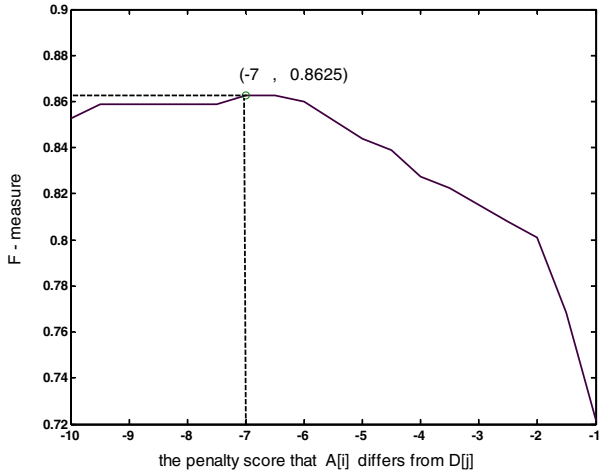

Figure 4. Relation between F-measure and the penalty score that A[i] differs from D[j].

a similar way, the redundant word penalty is also adjusted for the maximum F-measure. According to the above description, the redundant word penalty includes the beginning word penalty (BP) and the extended word penalty (EP). One unmatched word often appears in the true definition, and therefore, BP is assigned a small value, 0.5. Here we need to adjust EP. The relation between F-measure and EP is illustrated in Figure 5. The maximum F-measure is got when EP is 4.5. Then EP is assigned the value, 4.5. With the w(A[i],D[j]) and the redundant word penalty identified, the cutoff score is adjusted next. In Figure 6, the cutoff score ranges from -5 to 4, and the maximum F-measure is got when it is 0.5. Then the cutoff score is assigned the value, 0.5. There are some true definitions that contain several continuous unmatched words. If we want to find this kind of definitions, we need to adjust the variable C upward

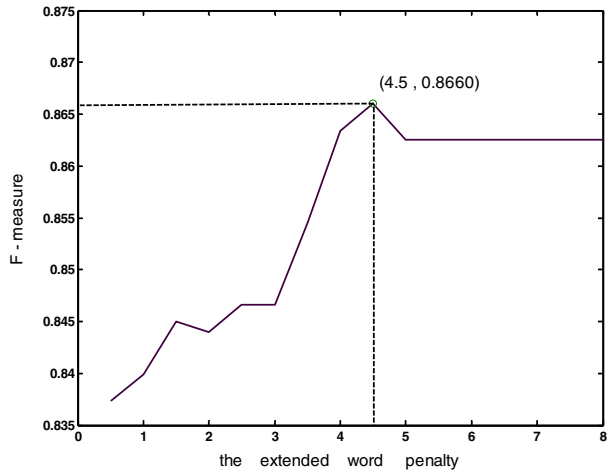

Figure 5. Relation between F-measure and the extended word penalty.

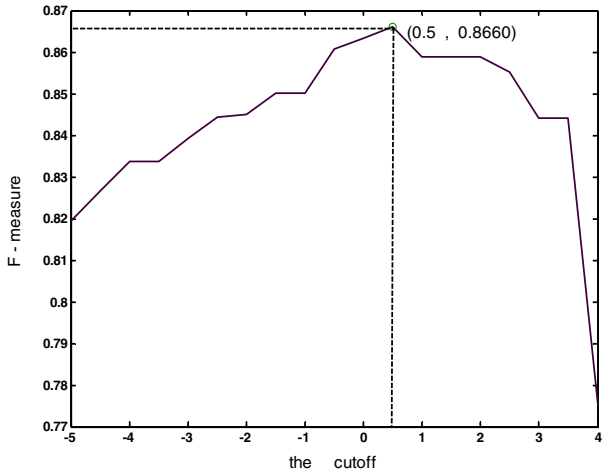

Figure 6. Relation between F-measure and the cutoff score.

(allowing more continuous unmatched words). Then a table is created for comparison when C = 1, 2 and 3. Table 1 shows that the maximum F-measure is got when C = 1.

Table 1. Relation Table between F-measure and C

	F-measure
C = 1	0.8660
C = 2	0.1261
C = 3	0.1204

2.6. Evaluation

To evaluate the algorithm, we have run it against a publicly available tagged corpus, the Medstract Gold Standard EVALUATION Corpus [15], which contains 168 <short form, long form> pairs. We compare our algorithm with three downloadable algorithms for acronym identification on the corpus: the Chang *et al.* [10] algorithm (obtained from `http://bionlp.stanford.edu/webservices.html`) at the three cutoff scores: 0.03, 0.14 and 0.88; the SLICE [7] algorithm (obtained from `http://uvdb3.hgc.jp/ALICE/program_download.html`); the S & H [11] algorithm (obtained from `http://biotext.berkeley.edu/software.html`).

Our result is strictly based on the corpus without corrections, and the extracted pairs must match the marked pairs exactly. In the result, our algorithm identified 153 <short form, long form> pairs. Out of these, 139 pairs are correct, resulting in a recall of 83% and a precision of 91%. Table 2 indicates the result of that comparison with other algorithms on the gold-standard corpus. In our result 14 pairs are incorrect. 6 of them are not biomedical items: lethal of scute (l'sc), basic helix-loop-helix (bHLH) , primary ethylene response element (PERE), Ca2+-sensing receptor (CaSR), intermediate neuroblasts defective (ind), eliminates an AP180 homolog (LAP). The other 8 pairs are only partially matched: for the true definition "general transcription factor IIB (TFIIB)", we get "transcription factor IIB (TFIIB)"; "RNA polymerase I (Pol I)", we get "polymerase I (Pol I)"; "RNA polymerase I (Pol II)", we get "polymerase I (Pol II)"; "multiprotein von Hippel-Lindau (VHL)", we get "von Hippel-Lindau (VHL)"; "cAMP-dependent protein kinase A (PKA)", we get "protein kinase A (PKA)"; "protein Hedgehog (Hh)", we get "Hedgehog (Hh)"; "transcription factor Cubitus interruptus (Ci)", we get "Cubitus interruptus (Ci)".

Table 2. Comparing with other algorithms on the gold-standard EVALUATION corpus

	Precision	Recall	F-measure
Chang (score=0.88)	93%	23%	0.37
Chang (score=0.14)	89%	76%	0.82
Chang (score=0.03)	87%	81%	0.84
ALICE	90%	77%	0.83
S & H	91%	77%	0.83
Our algorithm	91%	83%	0.87

Our algorithm misses 29 pairs: 8 of them are only par-

tially matched, that is to say, the true definition includes an additional word, for example, "RNA polymerase I (Pol I)", our algorithm misses the word "RNA". For the other 21 pairs there are several unmatched characters in the abbreviation, or the definition and abbreviation are not separated by parentheses.

3. Discussion

In this paper, we develop a new alignment algorithm to identify the definitions corresponding to abbreviations. In preparation for the alignment algorithm, we need to recognize the abbreviation and define the search space for the definition. From the search space we construct a candidate definition list, each item of which is aligned with the abbreviation using the alignment algorithm, and then the optimal definition is selected. Our evaluation demonstrate that the algorithm performs better than the others. It can find the true definitions exactly for regular abbreviations, even for some irregular abbreviations (e.g., <CNS1, cyclophilin seven suppressor>) our algorithm can also find their definitions.

One problem for this algorithm is from the procedure of optimizing parameters. We use a simple method to optimize these parameters: the w(A[i],D[j]), the redundant word penalty, the cutoff score and the variable C (in Equation (4)) based on a hypothesis that they are mutually independent. In fact there are some correlations between them. Many optimizing methods exist, such as Genetic Algorithm, Simulated Annealing Algorithm and so on. We have been trying to optimize the parameters with these methods, and this is the topic of our current research.

Our future work is to build a database of biomedical abbreviations and set up a web server for abbreviation retrieval. We will consider more syntactic clues in the contexts, the replacement problem between definitions and abbreviations, and how to recognize both acronyms and non-acronyms accurately.

In conclusion, a new alignment algorithm is developed to identify the definitions corresponding to abbreviations. The algorithm could find some irregular abbreviations and achieves a good result in biomedical text. It can also be used in the general text, or applied in other research areas.

4. Acknowledgements

We thank Ariel Schwartz and Marti Hearst for providing us the data and the early version of their algorithm. This work is supported by The Key Project of The National Nature Science Foundation of China under the grant No. 60533020.

References

[1] Jensen LJ, Saric J, Bork P: Literature mining for the biologist: from information retrieval to biological discovery. *Nat. Rev. Gen.* 2006, 7:119-129.

[2] Cohen AM, Hersh WR: A survey of current work in biomedical text mining. *Briefings in Bioinformatics* 2005, 6:57-71.

[3] Fred HL, Cheng TO: Acronymesis: the exploding misuse of acronyms. *Tex Heart Inst J.* 2003, 30:255-257.

[4] Pustejovsky J, Castano J, Cochran B: Automatic extraction of acronym-meaning pairs from medline databases, *Medinfo* 2001, 10:371-375.

[5] Zhou W, Torvik VI, Smalheiser NR: ADAM: another database of abbreviations in MEDLINE. *Bioinformatics* 2006, 22:2813-2818.

[6] Okazaki N, Ananiadou S: Building an Abbreviation Dictionary using a Term Recognition Approach. *Bioinformatics* 2006, 22:3089-3095.

[7] Ao H, Takagi T: Alice: An Algorithm to Extract Abbreviations from MEDLINE. *J. AM. Med. Inform. Assoc.* 2005, 12:576-586.

[8] Yu H, Hripcsak G, Friedman C: Mapping abbreviations to full forms in biomedical articles. *J Am Med Inform Assoc* 2002, 9:262-272.

[9] Park Y, Byrd RJ: Hybrid Text Mining for Finding Abbreviations and Their Definitions. *In Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing: 03-04 June 2001; Pittsburgh*. Edited by Lee L, Harman D: Association for Computational Linguistics Press; 2001:126-133.

[10] Chang JT, Schutze H, Altman RB: Creating an Online Dictionary of Abbreviations from MEDLINE. *J. Am. Med. Inform. Assoc* 2002, 9:612-620.

[11] Schwartz AS, Hearst MA: A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text. *In Proceedings of the 8th Pacific Sym-posium on Biocomputing: 03-07 January 2003; Lihue, Hawaii*. Edited by Altman RB, Dukner AK, Hunter L, Jung TA, Klein TE: World Scientific Press; 2003:451-462.

[12] Taghva K, Gilbreth J: Recognizing Acronyms and Their Definitions, *Technical Report, Information Science Research Institute, University of Nevada*, 1995.

[13] Needleman S, Wunsch C: A general method applicable to the search for simi-larities in the amino acid sequence of two proteins. *J Mol Biol* 1970, 48:443-453.

[14] Smith TF, Waterman MS: Identification of Common Molecular Subsequences. *J Mol Biol* 1981, 147:195-197.

[15] `http://www.medstract.org`