Sequence Alignment Based Citation Parser using BLAST and Smith-Waterman algorithm

G. Guru Brahmam, R. Rakesh Kumar

Abstract— The dramatic increase in the number of academic publications has led to a growing demand for efficient organization of the resources to meet researchers' specific needs. As a result, a number of network services have compiled databases from the public resources scattered over the Internet. However, publications in different conferences and journals follow different citation formats, so the problem of accurately extracting metadata from a publication string has also attracted a great deal of attention in recent years. In this paper, we propose a citation parser called BibPro for extracting metadata from citation strings by using a gene sequence alignment tool. The main enhancement of BibPro to previous tools is that BibPro does not need knowledge databases (e.g., an author name database) to generate feature indices for citation strings. Instead, only the order of punctuation marks in a citation string is used to represent its format. Second, BibPro employs the Basic Local Alignment Search Tool (BLAST) to find the most similar citation formats in database and then uses the Smith-Waterman algorithm to choose the best-fit citation format as the extraction template.

Index Terms— Data integration, digital libraries, information extraction, sequence alignment.

1 INTRODUCTION

PARSING citation information is essential for integrating bibliographical information published on the Internet, and many related applications, such as field-based searching, academic searching and analysis, and citation analysis [5]. However, it is difficult to design a system to automatically parse citation strings scattered over the Internet because, in addition to the problem of technical typing errors, there are a lot of different citation styles/formats. A citation string usually contains many fields (such as fields of author, title, publication information) arranged in many different formats depending on the type and venues to publish (e.g., for books, journals, conference papers, or technical reports). Hence, it is still challenging to design an automatic system for extracting metadata from citation strings.

Numerous studies on extracting metadata from citation strings are proposed in recent years [1-11, 15]. Those approaches can be classified into three categories: learning-based, template-based and rule-based approaches. Learning-based methods utilize machine learning techniques (e.g.,. the Hidden Markov Model (HMM) [7, 8], Support Vector Machines (SVM) [6], and Conditional Random Fields (CRF) [5]). Among them, CRF achieves the best performance with an overall word accuracy of 95.37% on the Cora reference dataset [5, 16]. Second, template-based methods work by using several template databases with various styles of citation templates (e.g., ParaCite [15] and INFOMAP [11]).

- *G.Guru Brahmam is currently pursuing masters degree program in computer science and engineering in JNTUH, india, PH-8121617726. E-mail: guru.vce@gmail.com*
- *R.Rakesh Kumar is currently pursuing masters degree program in computer science and engineering in JNTU, india, E-mail: rrakesh548@gmail.com*
(This information is optional; change it according to your need.)

ParaCite has been integrated with the EPrints.org software, and links between it with CiteBase, RefLink, and ISI Web of Science are currently considered. INFOMAP is a hierarchical template-based reference metadata extraction method with an overall average accuracy level of 92.39% for the six major citation styles detailed in [11]. Rule-based methods are widely used in real-world applications. For example, CiteSeer [1-4] is a well-known search engine and digital library that uses heuristics to extract certain subfields. It identifies titles and author names in citations with roughly 80% accuracy and page numbers with roughly 40% accuracy [1].

We proposed a template-based citation parser that achieved approximately 80% of parsing precision, but it has a number of drawbacks. First, the template construction work relies on an author name database to identify possible author names, so the quality of the database greatly affects the parsing accuracy, and a high quality author name database is never easy to obtain in practice. Second, several heuristic rules are applied in previous work to transform training citation strings into related templates, but these rules only work for several special cases. It causes problems when extracting metadata from a citation string that does not follow those special cases. Third, during the matching process in previous work, there has high probability to mismatch a wrong template to a citation string because there are several templates having the same similarity score, and no other information could be used to distinguish them. We call this the "template conflict" problem. Generally, the larger the template database, the more serious the problem is.

In this paper, we propose a new citation parser, called BibPro, which retains the advantages of previous work (e.g., using protein sequences to represent citations and applying the Basic Local Alignment Search Tool (BLAST) to find similar templates), and resolves the weaknesses. Instead of relying on an author name database and heuristic rules, BibPro uses the order of punctuation marks in a citation string as a feature to represent the string's citation format. Furthermore, to find out the template with the highest similarity score, we use the Smit-Waterman algorithm [14] in conjunction with BLAST to extract metadata from citation strings and align the features (a protein sequence) with the templates in our template database. Based on these two modifications, BibPro does not need any heuristics, and thus overcomes the template conflict problem. BibPro can effectively and systematically extract the fields of author, title, journal, volume, number (issue), month, year, and page information from citations of different formats.

2 BibPro: Citation Parser

2.1 Basic Ideas

Our system is based on the following two ideas. First, a protein sequence is used to represent a citation string. We split a citation string into several tokens and use an amino acid symbol to represent each token. Figure 1 shows an example of a citation string transformed into a protein sequence "AAADTTTT DLLLLDYRPHS".

Second, when transforming a citation string to a protein sequence, only the order of the punctuation marks and reserved words of a citation string are transformed. Redundant information is then filtered out to simplify the problem and accelerate the parsing process. Here, a protein sequence is designed to capture some features of citation strings, and BibPro transform many real citation strings (training set in our system) and their styles to the protein sequences; here, we call these "citation templates" in the following content. When parsing a new citation, BibPro invokes a well developed protein sequence matching program, called BLAST, to search a suitable template and then partitions and extracts the desired metadata (fields) from the citation string.

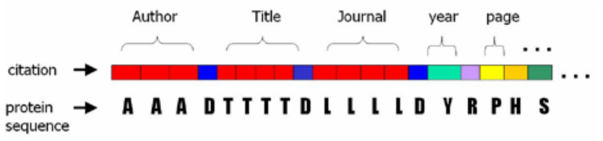


Figure 1. A Transformation from a citation string to a protein sequence

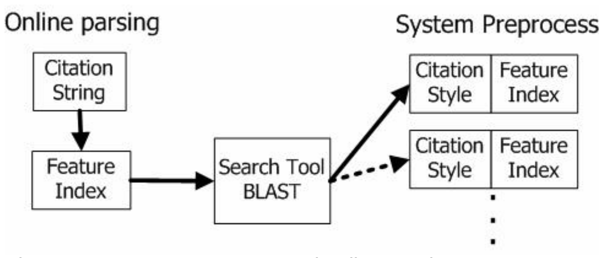


Figure 2. System preprocess and online parsing

Based on these two concepts, BibPro consists of two phases: a system preprocessing phase and an online parsing phase. The goal of the first phase is to generate feature indices for all training citation styles in advance, so that BLAST can find out a suitable citation styles for any given citation string in the second phase (see Figure 2.) During the online parsing phase, BibPro uses BLAST to find a citation style with a feature index similar to that of the citation string. BibPro then is able to extract metadata from this citation string.

2.2 System Architecture

The BibPro system comprises two basic systems: a template generating system and a parsing system, as shown in Figure 3. In the template generating system, we developed programs to retrieve BibTeX files from the Web, and since BibTeX format is field-based, we can easily parse them to get the correct metadata for a citation string. After that, we then use the title field as a search query to search for a citation in CiteSeer or another search engine, e.g., Google. By using this method, we can obtain a lot of citation strings and corresponding metadata. However, the collected data for a citation string may be inconsistent with its metadata. Moreover, our token-based form translation may encounter problems if different fields share the same token. For above reasons, we designed a template filter to ensure that a template is consistent with its citation string. The template filter uses some simple rules (e.g., the author, title and journal fields cannot appear more than once in a citation string). After this process, BibPro can construct a large number of citation templates, and each of which includes a citation style and a feature index.

Once the template database has been compiled, BibPro can provide the citation parsing service on-the-fly. In the parsing system, when a queried citation string is inputted, BibPro transforms it into a protein sequence, and uses BLAST to search candidate citation templates from template database by matching their feature indices. BibPro then uses the Smith-Waterman algorithm to calculate the similarity between candidate citation templates and the queried citation string, and the most fit citation template is chosen. According to the final citation template, BibPro can extract metadata from the queried citation string.

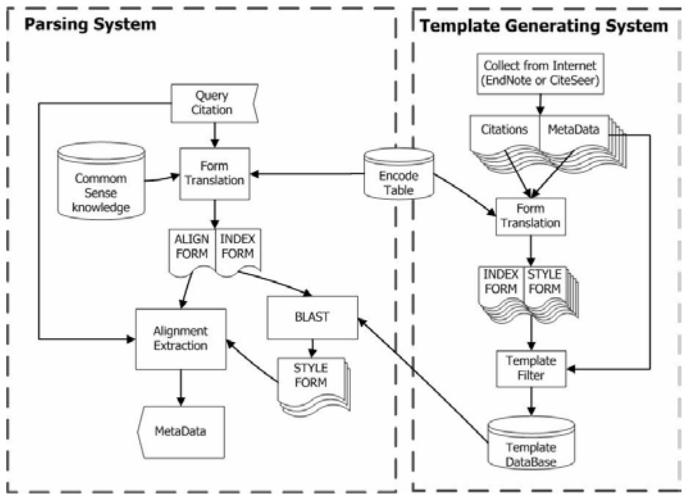


Figure 3. System architecture of BibPro

2.3 Form Translation

In order to use the BLAST to match similar citation templates, we need to transform incompatible citation strings into compatible protein sequences. Therefore, we need to consider the following questions:

- How many symbols can be used in a protein sequence?
- How many fields should be extracted from a citation string?
- How do we transform a citation string into a protein sequence and retain its citation style information?

Having considered the above questions, we created an encoding table, as shown in Table1, to define the relationships between the tokens in a citation string and the symbols in protein sequences.

Table 1. Encoding Table

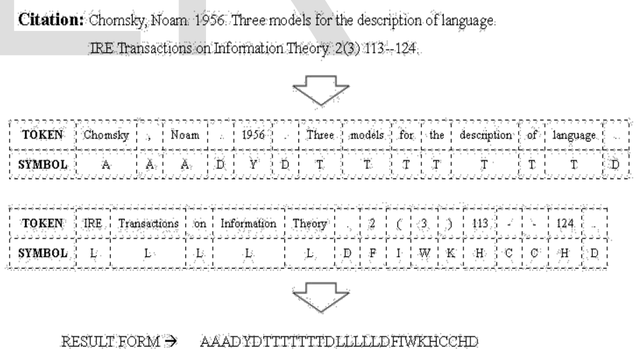
A: Author	N: numeral
T: Title	Q: @ # $ % ^ & * + = \ | ~ _ /
L: Journal	! ? 。
F: Volume value	I: ([{ < 「
W: Issue value	
H: Page value	K:)] } > 」
X: noise (unrecognized token)	D: .
M: Month	G: " “ ”
Y: Year (number: 1900-2010)	R: ,
S: Issue key. e.g. “no”, “No”	C: - :
P: Page key. e.g. “pp”, “page”	E: ' `
V: Volume key. e.g. “Vol”, “vo”	Z: ;
	B: blank (use one “B” to replace continuous “X”)

The design of the encoding table is based on the following observations:

- BLAST can only process sequences with 23 different symbols, so we use these 23 symbols to represent different fields, and use field separators to keep the citation style information in sequence.
- The most common fields in citation strings include: author, title, journal, volume, number, page, issue, month and year. We focus on extracting these fields from citation strings and assign a symbol to represent each field.
- The most common reserved words in citation strings include: "vo", "vol", "no", "NO", "pp", and "page". Since these words are also used to separate fields, we use a symbol to represent each kind of reserved words.
- The punctuation marks usually are used to separate fields, including: " , ", " . ", " ; ", " : ", " " " and " ' ". We also assign a symbol to represent each punctuation mark.
- Brackets and parentheses are synonymous in citation strings, so we use one symbol to represent both.

Several kinds of punctuation marks appear in the title field, such as: " - ", " ! ", " ? ". However, we only use one symbol to represent all of them because these marks are useless.

Figures 4 and 5 show examples of citation strings transformed into protein sequences. Figure 4 shows that when the correct answer of a citation string's partitions is known, we can correctly label each token. We use the "RESULT FORM" to represent the correct encoded protein sequence. If we can correctly transform a citation string to its RESULT FORM, it is easy to extract metadata from the citation strings. However, it is impossible to know the correct answer when online parsing a citation string. As shown in Figure 5, we can only label each token based on its content due to no other information we could have. If some unrecognized tokens are found, we replace them with an "X". We call this protein sequence the "BASE FORM". Thus, the goal of parsing process is to transform a citation string from its BASE FORM to its RESULT FORM.



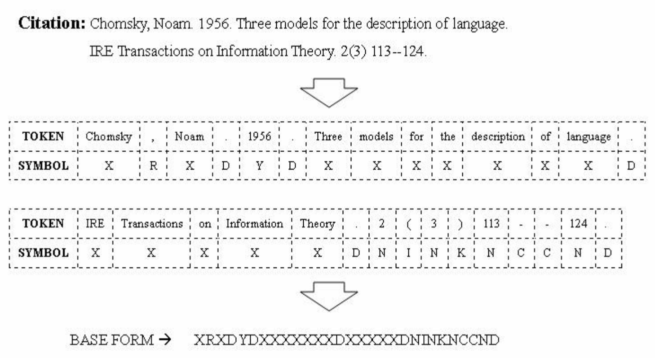
Citation: Chomsky, Noam. 1956. Three models for the description of language. IRE Transactions on Information Theory. 2(3) 113--124.

TOKEN	Chomsky	,	Noam	.	1956	.	Three	models	for	the	description	of	language	.
SYMBOL	A	A	A	D	Y	D	T	T	T	T	T	T	T	D

TOKEN	IRE	Transactions	on	Information	Theory	.	2	(	3	)	113	-	-	124	.
SYMBOL	L	L	L	L	L	D	F	I	W	K	H	C	C	H	D

RESULT FORM → AAADYDTTTTTTTDLLLLLDFIWKHCCHD

Figure 4. RESULT FORM of a citation string



Citation: Chomsky, Noam. 1956. Three models for the description of language. IRE Transactions on Information Theory. 2(3) 113--124.

TOKEN	Chomsky	,	Noam	.	1956	.	Three	models	for	the	description	of	language	.
SYMBOL	X	R	X	D	Y	D	X	X	X	X	X	X	X	D

TOKEN	IRE	Transactions	on	Information	Theory	.	2	(	3	)	113	-	-	124	.
SYMBOL	X	X	X	X	X	D	N	I	N	K	N	C	C	N	D

BASE FORM → XRXDYDXXXXXXXDXXXXXDNINKNCCND

Figure 5. BASE FORM of a citation string

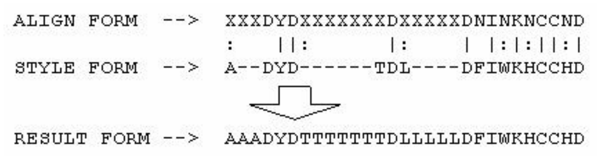
To transform a citation string from its BASE FORM into its RESULT FORM, we need to know its citation style. For this reason, we define several forms of a protein sequence for our mining process:

- STYLE FORM: To store citation style information. Although RESULT FORM can represent style information for specific citation strings, but it has a lot of excess information, such as the length of author, title, and journal fields. Thus we condense the redundant information in the RESULT FORM by using one of each symbol to represent the above fields. This sequence, called the "STYLE FORM", is used to represent a citation style.
- INDEX FORM: To recognize the style of a citation string, we use the order of punctuation marks in citation strings as the feature index. By removing all other unrecognized tokens. The INDEX FORM is the protein sequence that BLAST will try to match with similar sequences in the template database, so it is like an index used in a database.
- ALIGN FORM: Many fields of the citation like the author, title, and journal fields may contain punctuation marks. To extract these fields correctly, we have to remove punctuation marks inside the field. By some common sense knowledge, such as a dot always follow a name initial, we mark and group author and journal field tokens in citation strings during the online parsing phase. The processed sequence, called the "ALIGN FORM", is used to represent original citation string. Table 2 shows an example of each form.

Table2. Example of each form

BASE FORM	XRXDYDXXXXXXXDXXXXXDNINKNCCND
RESULT FORM	AAADYDTTTTTTT-DLLLLLDFIWKHCCHD
STYLE FORM	ADYDTDLDFIWKHCCHD
INDEX FORM	RDYDDDNINKNCCND
ALIGN FORM	XXXDYDXXXXXXXDXXXXXDNINKNCCND

When parsing a citation string, BibPro use the Smith-Waterman algorithm to perform global alignment between the STYLE FORM and the ALIGN FORM. With the alignment, BibPro is able get the RESULT FORM from the ALIGN FORM by adding "A" (author), "L" (journal), and "T" (title) in the correct positions and by changing "N" to its corresponding amino acid (e.g., an amino acid "N" may become F [volume], "W" [issue] or "H" [page]) as shown in figure 6. After that, by checking the original citation string and the RESULT FORM, BibPro can extract all the metadata



```
ALIGN FORM  -->  XXXDYDXXXXXXXDXXXXXDNINKNCCND
                 :   ||:      |:    | |:|:||:|
STYLE FORM  -->  A--DYD------TDL----DFIWKHCCHD

RESULT FORM -->  AAADYDTTTTTTTDLLLLLDFIWKHCCHD
```

Figure 6. Align STYLE FORM and ALIGN FORM to get RESULT FORM

3. Smith-Waterman Algorithm:

3.1. Initialization of Matrix

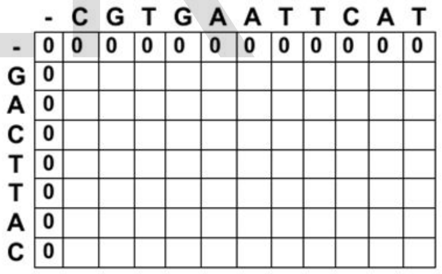
The basic steps for the algorithm are similar to that of Needleman-Wunsch algorithm. The steps are:

- Initialization of a matrix.
- Matrix filling with the appropriate scores.
- Trace back the sequences for a suitable alignment.

To study the Local sequence alignment consider the given below sequences.

CGTGAATTCAT(sequence#1orA)
GACTTAC (sequence #2 or B)

The two sequences are arranged in a matrix form with A+1columns and B+1rows. The values in the first row and first column are set to zero as shown in Figure 7.



	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0											
A	0											
C	0											
T	0											
T	0											
A	0											
C	0											

Figure 7: Initialization of Matrix

Variables used:

i,j describes row and columns.
M is the matrix value of the required cell (stated as $M_{i,j}$)
S is the score of the required cell ($S_{i,j}$)
W is the gap alignment

3.2 Matrix Filling

The second and crucial step of the algorithm is filling the entire matrix, so it is more important to know the neighbor values (diagonal, upper and left) of the current cell to fill each and every cell.

$$M_{i,j} = Maximum\ [M_{i-1,j-1} + S_{i,j},\ M_{i,j-1} + W,\ M_{i-1,j} + W, 0]$$

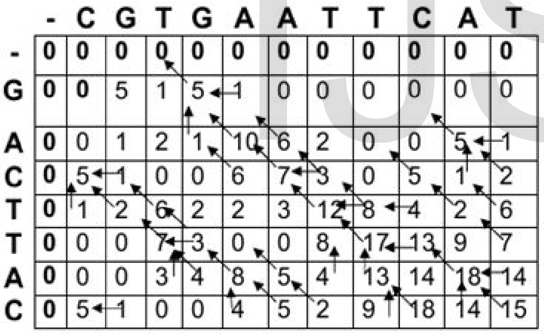
As per the assumptions stated earlier, fill the entire matrix using the assumed scoring schema and initial values. One can fill the 1st row and 1st column with the scoring matrix as follows.

The first residue (nucleotides or amino acids) in both sequences is 'C' and 'G', the matching score or the mismatching score is going to be added the neighboring value which is diagonally located i.e. 0. The upper and left values are added to the gap penalty score from the matrix. So the scoring schema equation can be shown as follows.

$$\begin{aligned} M_{1,1} &= Maximum\ [M_{0,0} + S_{1,1},\ M_{1,0} + W,\ M_{0,1} + W,\ 0] \\ &= Maximum\ [0(-3), 0 + (-4), 0 + (-4), 0] \\ &= Maximum\ [-3, -4, -4, 0] \\ &= 0 \end{aligned}$$

From the above calculations the maximum value obtained is 0. Finding the maximum value for $M_{i,j}$ position, one can notice that there is no chance to see any negative values in the matrix, since we are taking 0 as lowest value.

After filling the matrix, keep the pointer back to the cell from where the maximum score has been determined. In the similar fashion fill all the values of the matrix of the cell.
For the example the matrix can be filled is shown in Figure 8.



	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	5	1	5	1	0	0	0	0	0	0
A	0	0	1	2	1	10	6	2	0	0	5	1
C	0	5	1	0	0	6	7	3	0	5	1	2
T	0	1	2	6	2	2	3	12	8	4	2	6
T	0	0	0	7	3	0	0	8	17	13	9	7
A	0	0	0	3	4	8	5	4	13	14	18	14
C	0	5	1	0	0	4	5	2	9	18	14	15

Figure 8: Matrix filling with back pointers

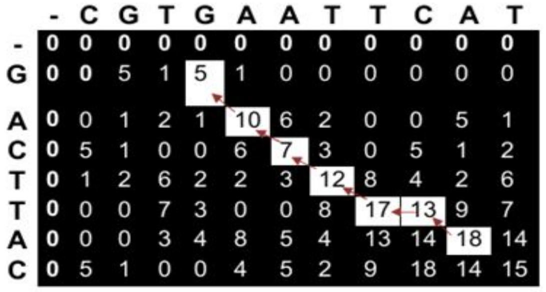
Each cell is back pointed by one or more pointers from where the maximum score has been obtained.

3.3Trace backing the sequences for an optimal alignment:

The final step for the appropriate alignment is trace backing, prior to that one needs to find out the maximum score obtained in the entire matrix for the local alignment of the sequences. It is possible that the maximum scores can be present in more than one cell, in that case there may be possibility of two or more alignments, and the best alignment by scoring it.

In this example we can see the maximum score in the matrix as 18, which is found in two positions that lead to multiple alignments, so the best alignment, has to be found.

So the trace back begins from the position which has the highest value, pointing back with the pointers, thus find out the possible predecessor, then move to next predecessor and continue until we reach the score 0 (Figure 9)



	-	C	G	T	G	A	A	T	T	C	A	T
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	5	1	5	1	0	0	0	0	0	0
A	0	0	1	2	1	10	6	2	0	0	5	1
C	0	5	1	0	0	6	7	3	0	5	1	2
T	0	1	2	6	2	2	3	12	8	4	2	6
T	0	0	0	7	3	0	0	8	17	13	9	7
A	0	0	0	3	4	8	5	4	13	14	18	14
C	0	5	1	0	0	4	5	2	9	18	14	15

Figure 9: Trace back of first possible alignment

It is possible to find two pointers pointing out from one cell, where both ways (alignments) can be considered, best one is found by scoring and finding maximum score among them.

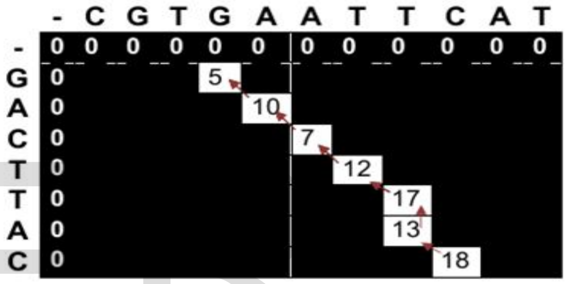
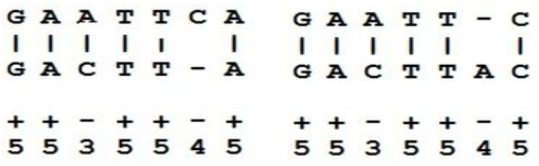


Figure 10: Trace back of second possible alignment

Thus a local alignment is obtained and one can see the possible alignments as in Figure 11.



```
G A A T T C A      G A A T T - C
| | | | |   |      | | | | |   |
G A C T T - A      G A C T T A C

+ + - + + - +      + + - + + - +
5 5 3 5 5 4 5      5 5 3 5 5 4 5
```

Figure 11: Scoring for best alignment

The two alignments can be given with a score, for matching as +5 , mismatch as -3 and gap penalty as -4, sum up all the individual scores and the alignment which has maximum score after this can be taken as the best alignment.

4. Conclusion and Future Work

Parsing citations is still a challenging problem due to the diverse nature of citation formats. In this paper, we proposed a template-based citation parsing system called "BibPro", which extends previous work by using the order of punctuation marks in a citation string to represent its format. When online parsing a citation string, BibPro transforms the citation string into a protein sequence and apply two sequence alignment techniques, BLAST and the Smith-Waterman algorithm, to find out the most similar template for exaction metadata from the citation. According to our experiments, BibPro performs very well and is scalable.

There are still several challenges when applying the BibPro into real world applications. One challenge is to obtain an accurate large-scale training dataset with all kinds of citation formats. The training dataset collected from the Web always contains a lot of errors, such as missing values, spelling errors, inconsistent abbreviations, and extraneous tokens [9]. Another challenge is that many publication formats include a lot of fields, and it is difficult to extract all the fields for all citation strings. Hence, we only concentrate on the most common information (fields) for all publication formats in this paper.

In the future, we focus on designing an automatic system to extract all publication information from researchers' publication lists by integrating the BibPro with our previous work [17].

5. References

[1] Giles, C. L., Bollacker, K. D., and Lawrence, S. "CiteSeer: an automatic citation indexing system," *Digital Libraries 98* Pittsburgh PA USA, 1998.

[2] Bollacker, K. D., Lawrence, S., and Giles, C. L., "CiteSeer: an autonous Web agent for automatic retrieval and identification of interesting publications," 1998.

[3] Lawrence, S., Giles, C. L., and Bollacker, K. D., "Autonomous citation matching," 1999.

[4] Lawrence, S., Giles, C. L., and Bollacker, K. D. "Digital Libraries and Autonomous Citation Indexing." *IEEE Computer*. Vol 32, 1999, pp. 67-71.

[5] F. Peng, A. McCallum, "Accurate information extraction from research papers using conditional random fields," *HLT-NAACL*, 2004, pp. 329-336.

[6] Hui Han, Giles, C.L., Manavoglu, E., Hongyuan Zha, Zhenyue Zhang, Fox, E.A. "Automatic document metadata extraction using support vector machines," *JCDL*, 2003.

[7] K. Seymore, A. McCallum, R. Rosenfeld, "Learning hiddenMarkov model structure for information extraction," *AAAI-99*, 1999, pp. 37-42.

[8] Takasu, A. "Bibliographic attribute extraction from erroneous references based on a statistical model," *JCDL*, 2003, pp. 49-60.

[9] Agichtein, E. and Ganti, V., "Mining reference tables for automatic text segmentation", *KDD'04*, 2004, pp. 20-29.

[10] I-Ane Huang, Jan-Ming Ho, Hung-Yu Kao, and Shian-Hua Lin, "Extracting citation metadata from online publication lists using BLAST," *In PAKDD*, 2004.

[11] Min-Yuh Day et al. "Reference metadata extraction using a hierarchical knowledge representation framework." *Decision Support Systems*, 2006.

[12] S. F. Altschul, W. Gish, W. Miller, E. Myers and D. Lipman. "A basic local alignment search tool." *J. Mol. Biol.*, 215, 1990, pp. 403-410.

[13] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/similarity.html

[14] Needleman, S. B. and Wunsch, C. D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *J. Mol. Biol.*, 48, 1970.

[15] http://paracite.eprints.org/

[16] http://www.cs.umass.edu/~mccallum/code-data.htm

[17] K.-H. Yang, J.-M. Chung and J.-M. Ho, "PLF: A Publication List Web Page Finder for Researchers", in the 2007 IEEE/WIC/ACM International Conference on Web Intelligence, 2007.