Extracting Citation Metadata from Online Publication Lists Using BLAST

I-Ane Huang[1], Jan-Ming Ho[1], Hung-Yu Kao[1], Wen-Chang Lin[2]

[1]Institute of Information Science, Academia Sinica, Taiwan
[2]Institute of Biomedical Sciences, Academia Sinica, Taiwan

hia@iis.sinica.edu.tw, hoho@iis.sinica.edu.tw,
bobby@iis.sinica.edu.tw, wenlin@ibms.sinica.edu.tw

Abstract. Scientific research reports require a great deal of citation, therefore an automatic citation tool would be of great use. Due to numerous models, it is difficult to automatically transform semi-structured citation data into structured citations. Some digital library institutes, like ResearchIndex (CiteSeer) or Op-Cit, have attempted automatic citation parsing. In order to recognize citation metadata, we use gene sequence alignment tool to recognize citation data in our method. Known semi-structured citation data are transformed into protein sequences, and saved in a template database. To parse new semi-structured citation data, we can translate it into a protein sequence. We then use BLAST (Basic Local Alignment Search Tool), a sequence alignment tool, to find the most similar template to the protein sequence from the template database previously constructed. We can then parse metadata according to the template. Using the 2,500 templates generated by our template generating system as the template database and parsing all of the 2,500 citation using our parsing system, we obtain 89% precision rate. Using the same template database, ParaTools obtains 79% precision rate. ParaTools contains about 400 templates in the system. Using the default template database, ParaTools only obtains 30% precision rate.

1. Introduction

It is difficult for a computer to automatically parse citations because there are a lot of different citation formats. Citations always include author, title, and publication information. Publication information format varies according to publication type, e.g., books, journals, conference papers, research reports, and technical reports. Publication information can include publication name, volume, number, page number, year published, month published, and publisher's address. Citations can be presented in either structured or semi-structured form. Semi-structured citation form is more flexible, so bibliographies created by different people may have different citation forms. Metadata order may be different as well as their attributes. Bibliographies on the Internet are usually in semi-structured form. If we want to use their data, we must first transform the semi-structured bibliography into structured bibliography. We have to analyze the metadata of each citation, and build up an index for bibliography

searches and citing statistics. In this paper, we discuss how to transform semi-structured bibliographies into uniform structured data, the core problem of citation data processing.

CiteSeer [1][2][3][4], which use heuristics to extract certain subfields, can "find the titles and authors in citations roughly 80% of the time and page numbers roughly 40% of the time" [1]. Another system, ParaTools [5][6][7][8] (short for ParaCite Toolkit) is a collection of Perl modules used for reference parsing. It uses a template-based reference parser to extract metadata from references. Our approach is similar to ParaTools in that we also use a template-based reference parser, but we found a better alignment from BLAST [9][10].

There are about 30 billion nucleotides in a human genome, and for about every 1,000 base pairs, there will be a nucleotide difference in genomes. We can use BLAST to compare sequence to identify whether the sequences belong to the same person. We realized that we could use this method to identify citation. In our system, we use a form translation program to translate citations into a form that we can process more easily. The form we use is a protein sequence because we can use BLAST, a well-developed protein sequence matching program, to process it. BLAST needs a scoring table to search for most similar sequences in a protein sequence database. This database stores the templates of known citations that have been translated into protein form. After finding the most similar sequence template, we use a pattern extraction program to parse the citation metadata according to the template. Once the metadata are correctly parsed, we manually validate them and add them into our knowledge database.

The rest of the paper is organized as following. Section 2 is system architecture and design. Section 2.1 introduces the BLAST. Section 2.2 defines the tokens that we use. Section 2.3 tells how the tokens translated to amino acids. Section 2.4 and 2.5 are the template generating system and citation parsing system. In section 3, we do some experiment here. Section 4 is conclusion and future works.

2. System Architecture and Design

As shown figure 1, we can divide our system into two subsystems: a template generating system and a citation parsing system. For the template generating system, we manually crawl BibTeX files on the web to obtain our template generating data. From the BibTeX file, our system can find out the titles, and then use these titles automatically obtain semi-structured citations from the CiteSeer web site. Now we have both semi-structured data from CiteSeer and structured data from BibTeX files. Since the semi-structured and structured data are the same but presented in different forms, this system can use these two forms to automatically make a template database. After the template database is constructed, the system begins the parsing process. In the parsing system, BLAST is used to compare strings. We transform a citation into protein form, and use BLAST to search for the most similar sequence in the template database. We can then parse the citation according to the template that BLAST finds.

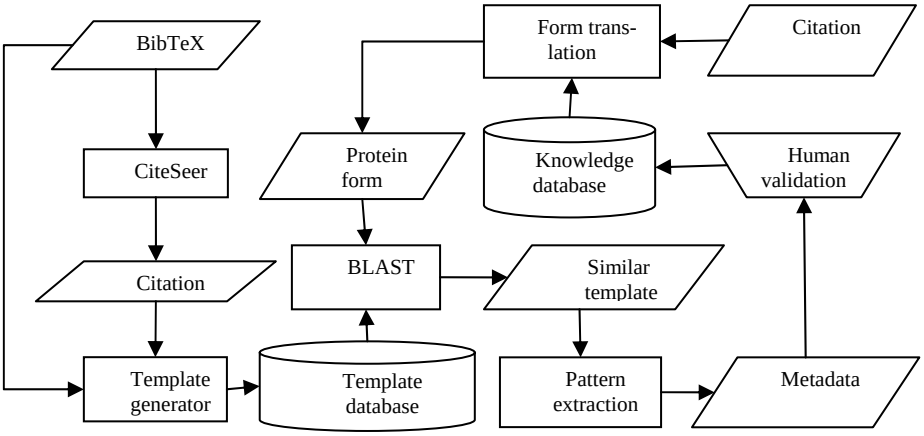

Fig. 1. The architecture of this system.

2.1 BLAST

BLAST is a similarity search tool developed by Altschul et al. (1990), it is based on dynamic programming. It is used to search for optimal local alignments between sequences. BLAST breaks the query and database sequences into fragments (words). It searches the matches for the word of length W that scores at least T. Matches are extended to generate an alignment with a score exceeding the threshold of S. The quality of each pair-wise alignment is represented as a score. Scoring matrices are used to calculate the score of the alignment amino acid by amino acid. The significance of each alignment is computed as a P value or an E value [10].

2.2 Tokens

Before explaining the architectures of the template generating and citation parsing systems, we have to identify the tokens they use. We use regular expression to precisely define the tokens. Tokens are classified into three types: numerical, general, and punctuation as follows:

- Numerical tokens: [0-9]+

  We reclassify the numerical token again as a year number, or general number. The regular expressions are as follows:

  - Year: [12][0-9][0-9][0-9]
  - General number: Otherwise
- General tokens: [a-zA-Z]+

We reclassify the general token to key words that often appear in citations, like page, number, volume, month, name or unknown. The regular expressions are as follows:

- Number:
  [Nn][Oo]
  |[Nn][Nn]
  | [Nn][Uu][Mm][Bb][Ee][Rr]
- Page:
  [Pp][Pp]
  | [Pp][Aa][Gg][Ee]([Ss])?
- Volume:
  [Vv][Oo]
  | [Vv][Oo][Ll]([Uu][Mm][Ee])?
- Month:
  [Jj][Aa][Nn]([Uu][Aa][Rr][Yy])?
  | [Ff][Ee][Bb]([Rr][Uu][Aa][Rr][Yy])?
  | [Mm][Aa][Rr]([Cc][Hh])?
  | [Aa][Pp][Rr]([Ii][Ll])?
  | [Mm][Aa][Yy]
  | [Jj][Uu][Nn]([Ee])?
  | [Jj][Uu][Ll]([Yy])?
  | [Aa][Uu][Gg]([Uu][Ss][Tt])?
  | [Ss][Ee][Pp][Tt]([Ee][Mm][Bb][Ee][Rr])?
  | [Oo][Cc][Tt]([Bb][Ee][Rr])?
  | [Nn][Oo][Vv]([Ee][Mm][Bb][Ee][Rr])?
  | [Dd][Ee][Cc]([Ee][Mm][Bb][Ee][Rr])?
- Name: We have a database that stores about 2,000 name tokens. If the token appears in the name database, it is identified as a name token.
- Unknown: If the general token is not classified above, it is classified as unknown.

- Punctuation tokens: [\"\'.\(),;\-!\?]

2.3 Protein Sequence Translation

In order to identify tokens, we use regular expression to describe the tokens in the last section. We can translate the token into a protein sequence according the following rules:

Y: Match to Year
N: Match to General number
S: Match to Number
P: Match to Page
V: Match to Volume
M: Match to Month
A: Match to Name
NULL: Match to Unknown

G: Match to Punctuation token “ or ‘
D: Match to Punctuation token . or ; or ?
I: Match to Punctuation token (
K: Match to Punctuation token)
R: Match to Punctuation token ,
Q: Match to Punctuation token : or – or !

When the tokens translate to the amino acids, the combinations of the amino acids become a protein sequence. We call this sequence a prototype protein sequences, and use it to represent the citation. We can use this sequence to search for the most similar template in the database. We also use it to construct the template database. The citation in Figure 2(a) can transform into Figure 2(b) by translating the tokens according to the rules described above.

2.4 Template generating system

In the template generating system, we construct a template database that contains the templates that represent most citation formats. Because BibTeX files are widely used in bibliographies, we retrieve BibTeX files from the Internet. BibTeX format was designed by Oren Patashnik and Leslie Lamport. It is field based, so we can parse the data of each field easily. We use the title field to search the citations in CiteSeer. We get the metadata of the citations found on the web by parsing the corresponding BibTeX files to construct the templates. We begin with the method described in section 2.3 to create the prototype protein sequence. Since we have found the metadata of each field, we can find the data in the CiteSeer citation and then modify the sequence by adding A (author), L (journal), and T (title) into the correct position in the sequence. We then change N to their corresponding amino acids. An amino acid N may become F (value of volume) or W (value of number). This is illustrated in Figure 2(c). It is now a completed protein sequence, and the template is finally constructed. We store this template in the template database.

2.5 Parsing System

After transforming the citation into a prototype protein sequence, we search for the most similar sequence in the template database by using BLAST. The parsing system is like a reverse process of the template generating system. After we find a template, we use it to extend the prototype protein sequence into a complete protein sequence. Now the metadata can be parsed. If we want to parse the citation shown in Figure 2(a), we must transform it into prototype protein sequence as figure 2(b). By entering this sequence into BLAST, we find the template AAAAAAAAARGTTTTTTTTTGRLLLLLLLRVDFRSDWRMDYD. In this template, author is in the first position. The field of author and the field of title are separated by RG. (punctuation marks), as are the title and journal fields. Modifying Figure 2(b) according this template, we get Figure 2(c). Then, by checking the original citation, we can parse out all the metadata correctly.

Fig. 2(a). The original citation we want to parse.

	Q		A		M	Q		A	R	G								G
R					D			R	V	D	N	R	S	D	N	R	M	D	Y	D

Fig. 2(b). The citation in the Fig. 2(a) transforms into its protein sequence as QAMQARGGRDRVDNRSDNRMDYD. Each blank in the table represents a token in the citation.

A	A	A	A	A	A	A	A	A	R	G	T	T	T	T	T	T	T	T	G
R	L	L	L	L	L	L	L	R	V	D	F	R	S	D	W	R	M	D	Y	D

Fig. 2(c). We transform the citation sequence into its protein sequence. The template of the citation is AAAAAAAAARGTTTTTTTTTGRLLLLLLLRVDFRSDWRMDYD

3. Experiment Results

Ideally, all of the metadata in the BibTeX should be consistent with the metadata in the citations searched from CiteSeer. Unfortunately, only a few data are consistent. We experimented with 2,500 article citations, but only 100 citations were consistent with BibTeX. To overcome this problem, we create a precision evaluation method to test whether the data is correctly parsed. We define the precision of each subfield as:

$$\text{Subfield precision} = \frac{\#[(Token_{Number}^{subfield} + Token_{word}^{subfield}) \cap Token_{BibTex}^{subfield}]}{\#[(Token_{Number}^{subfield} + Token_{word}^{subfield}) \cap Token_{BibTeX}]} \quad \textbf{(1)}$$

$Token_{Number}^{subfield}$: denotes the number tokens that appear in the parsed subfield.

$Token_{word}^{subfield}$: denotes the general word tokens that appear in the parsed subfield.

$Token_{BibTex}^{subfield}$: denotes the tokens that appear in the specific subfield in the BibTeX file.

$Token_{BibTeX}$: denotes all of the tokens that appear in the BibTeX file.

The denominator of subfield precision represents the number of the tokens which exist both in the citation file and in the BibTeX file. The numerator of the subfield precision represents the number of the tokens which are correctly parsed. We then define the total precision of a citation as the average of subfield precisions. Using the 2,500 templates generated by our template generating system as the template database and parsing all of the 2,500 citation using our parsing system, we obtain 89% precision rate. Using cross-validation to validate the precision of our system, we divide the citations into 10 subsets, and get an average precision rate of 84.8%. The precision distribution chart of the first 100 citation is shown in Figure 3. Parsing results are adequate because 85% of the citations achieve a precision rate of 70% or higher. Although the precision calculated here is not actually precision, the disparity between this precision here and actual precision is small. The actual precision of the parsed result is roughly 80%. BLAST needs a scoring table to evaluate the alignment result. In our system, we use a scoring shown in Figure 4. We had also tried a lot of different scoring table to parse the citation. The diagonals of all the various scoring matrices were always positive, and the variation in the percision of the parsing results because of a particular choice of scoring table was less than 3%. Both ParaTools and our system are template-based reference parsers. The completeness of the template database is an important factor to template-based parsing systems. We illustrate the effect of template completeness on precision in Figure 5. Our system performs better than ParaTools for all tested template completeness. In Table 1, we present the quality of parsing result. Our quality is better than ParaTools'. The parsing results will be used in the future. If the quality is not good, it is insignificant for reusing.

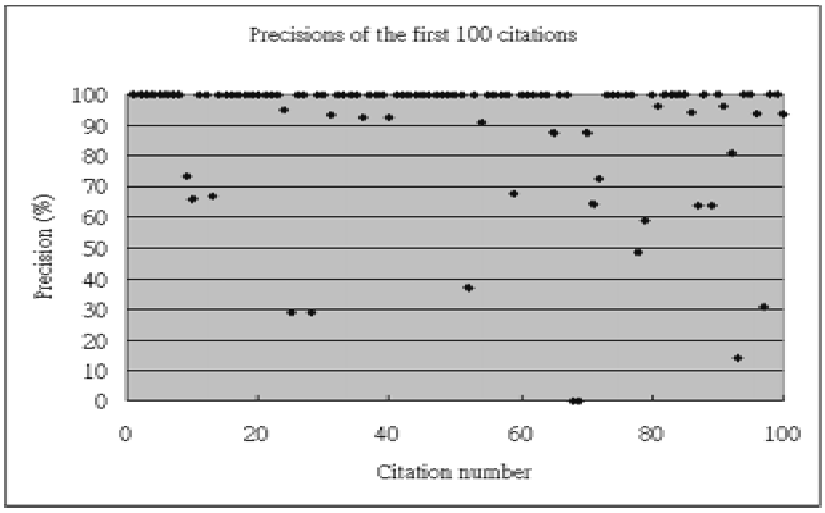

Fig. 3. Precision rates of the first 100 citations. The X axis is the assigned number of one citation, and the Y axis is the precision of that citation.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W
A	9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9	-9
R	-9	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
N	-9	-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
D	-9	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
C	-9	-1	-1	-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Q	-9	-1	-1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
E	-9	-1	-1	-1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
G	-9	-1	-1	-1	-1	-1	-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
H	-9	-1	-1	-1	-1	-1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
I	-9	-1	-1	-1	-1	-1	-1	-1	-1	9	-1	-1	-1	-1	-1	-1	-1	-1
L	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	4	-1	-1	-1	-1	-1	-1	-1
K	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	9	-1	-1	-1	-1	-1	-1
M	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-1	-1	-1	-1	-1
F	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-1	-1	-1	-1
P	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-1	-1	-1
S	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	6	-1	-1
T	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	9	-1
W	-9	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	4

Fig. 4. The scoring table we use this scoring table to evaluate the alignment score of the protein sequences

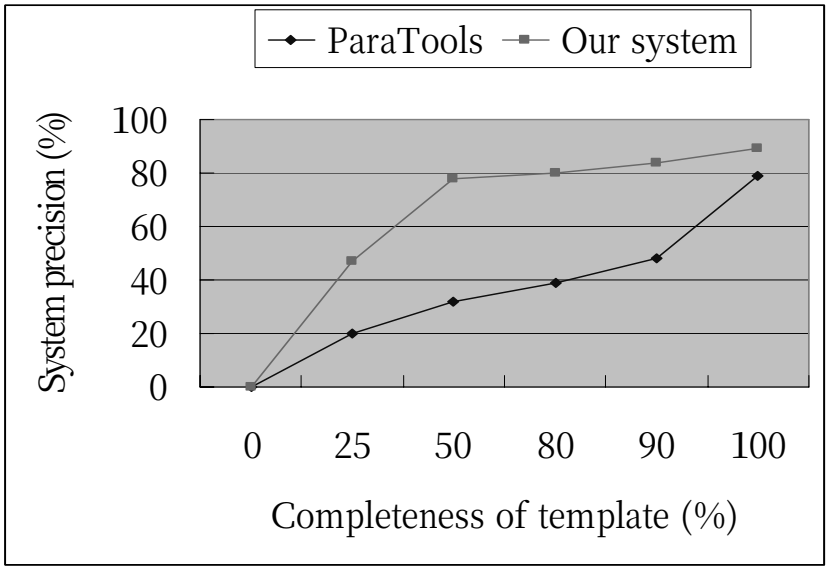

Fig. 5. For varied template database, different precision rates are achieved.

Table 1. Divide the data into k sets, and do cross-validation. The quality of the parsing result is defined good if the precision rate of the parsing result is better than 70%. The good results also contain perfect results which precisions rates are 100%.

Quality of the parsing result			
		Percent	
Value of k	Description	OpCit	Our system
10			
	Perfect	21	71
	Good (>70%)	26	85
	Not good(<70%)	74	15
5			
	Perfect	21	59
	Good(>70%)	26	76
	Not good(<70%)	74	24
2			
	Perfect	6	37
	Good(>70%)	6	52
	Not good(<70%)	94	48

4. Conclusion and Future Works

It is flexible for a template-based system to deal with citations. We not only can add new citation templates easily, but also can search the most similar template to rapidly parse the metadata. The precision of parsing result will be different for the completeness of template database. ParaTools contains about 400 templates in the system, but it does not fit the 2,500 data well. The precision rate for ParaTools to run the data is only 30%. Although ParaTools also use template-based method to parse metadata, our system performs better in the same template. We think that it is contributed by the string comparison tool, BLAST. In the future, we will create more templates to match most citation formats. We will use more Bioinformatics tools to resolve the problems happened in the computer category.

Acknowledgment

The comments of Shian-Hua Lin and Ming-Jing Hwang[2] during the preparation of this paper were very helpful. We thank for their help.

Reference

[1] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence, "CiteSeer: An Automatic Citation Indexing System," Digital Libraries 98 Pittsburgh PA USA.

[2] Kurt D. Bollacker, Steve Lawrence, and C. Lee Giles, "CiteSeer: An Autonomous Web Agent for Automatic Retrieval and Identification of Interesting Publications," 2nd International ACM Conference on Autonomous Agents, pp. 116-123, ACM Press, May, 1998.

[3] Steve Lawrence, C. Lee Giles and Kurt D. Bollacker, "Autonomous Citation Matching," Proceedings of the Third International Conference on Autonomous Agents, Seattle, Washington, May 1-5, ACM Press, New York, NY, 1999.

[4] Steve Lawrence, C. Lee Giles and Kurt D. Bollacker, "Digital Libraries and Autonomous Citation Indexing," IEEE Computer, Vol. 32, No. 6, pp. 67-71, 1999.

[5] Harnad, Stevan and Carr, Leslie, "Integrating, Navigating and Analyzing Eprint Archives Through Open Citation Linking (the OpCit Project)," Current Science (special issue honour of Eugene Garfield), Vol. 79, pp. 629-638, 2000.

[6] Donna Bergmark, "Automatic Extraction of Reference Linking Information from Online Documents," Cornell University Technical Report, TR 2000-1821, November, 2000.

[7] Donna Bergmark and Carl Lagoze, "An Architecture for Automatic Reference Linking," Cornell University Technical Report, TR 2000-1820, October, 2000.

[8] Mike Jewell, "ParaTools Reference Parsing Toolkit - Version 1.0 Released," D-Lib Magazine, Vol. 9, No.2, Feb., 2003.

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

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